11-12 1981

informatyka

WYDAWNICTWO NOT

Zmiany?

Ośrodki informatyczne dążą do usamodzielnienia się. Tendencja ta jest bardzo wyraźna w sieciach ZETO i ETOB, gdzie bezzasadność istnienia dotychczasowych administracji stała się dla większości pracowników oczywista. Podobnie jest w innych przedsiębiorstwach. Zasadniczych reorganizacji jednak nadal nie widać. A brak zdecydowanych posunięć może być w tym przypadku bardzo niekorzystny, może zanegować sens reformy gospodarczej.

INFORMATYKA nie jest w stanie zmusić władze do przejścia od zapowiedzi do czynów, niemniej może pomóc uniezależniającym się ośrodkom. Łamy pisma mogłyby ułatwić obieg „oddolnych" informacji — o możliwościach ośrodków oraz ich trudnościach, które dałoby się pokonać dzięki czyjemuś wsparciu, o nadwyżkach lub niedoborach fachowej kadry, o sprzęcie, który — na przykład — nie jest wykorzystywany, o systemach czy programach, o których nie wiedzą potencjalni użytkownicy, wreszcie — o samych informatycznych przeobrażeniach.

W przyszłym roku INFORMATYKA ma być jednak miesięcznikiem, wprawdzie o zmniejszonej objętości (32 kolumny), niemniej pismem bardziej niż dzisiaj aktualnym, być może nadążającym za szybko zmieniającą się rzeczywistością. Sądzimy, iż jest w tym pewna szansa dla ośrodków samodzielnych, zmuszonych do walki o swój byt. Chętnie nawiążemy z nimi bezpośredni kontakt. Przy okazji — przypominamy o konieczności pisania artykułów zwięzłych, możliwie krótkich, oddających sedno przedstawionych spraw.

Informatyka jest dziedziną szczególnie krytykowaną; ośrodkom obliczeniowym może więc zagrażać nawet bankructwo (mimo ich intelektualnego i technicznego potencjału). Ponieważ wiele argumentów przemawia za tym, iż byłoby to — ze społecznego punktu widzenia — niekorzystne, warto dążyć do utrzymania informatyczego narzędzia.

Opisanie czynów, przedstawienie sądów, nazwanie problemów — daje większą pewność, że nie będą popełnione stare błędy, że jednak nie skończy się na słowach. Spróbujmy zapisać stan obecny, dzisiejsze mankamenty i braki, poszukać i ujawnić ich genezę. Informatykom jest to dzisiaj wyjątkowo potrzebne.

Redakcja

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA SIGMA CZASOPISMA I KSIĄŻKI TECHNICZNE
ul. Świętokrzyska 14a
00-950 Warszawa
skrytka pocztowa 1004

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pokój 326, tel. 27-71-40, dyżury redakcji 10.00—12.00

Zakł. Graf. „Tamka". Zam. 319. Obj. 6,0 ark. druk. Nakład 6300 egz. L-114.

Cena egzemplarza zł 30.— INDEKS 36124 Prenumerata roczna zł 360.—

Gwiazda J.: Czy informatyce potrzebna jest reforma

INFORMATYKA 1981, nr 11—12, s. 4

Krytyczna ocena dotychczasowego rozwoju polskiej informatyki w warunkach istniejącego systemu zarządzania gospodarką narodową. Scharakteryzowano nowe zadania, główne kierunki zastosowań oraz sposoby oddziaływania informatyki w systemie zreformowanym.

Я. Гвязда: Нужна ли вычислительной технике реформа

ИНФОРМАТИКА 1981, № 11—12, стр. 4

Критическая оценка современного развития польской вычислительной техники в условиях существующей системы управления народным хозяйством. Определены новые задачи, основные направления применения, а также способы действия в преображенной системе.

Orłowski H.: Przemysł minikomputerowy a informatyka w Polsce

INFORMATYKA 1981, nr 11—12, s. 6

Szczegółowa charakterystyka obecnego stanu polskiego przemysłu minikomputerowego oraz ocena na tym tle rozwoju informatyki w Polsce ze wskazaniem przyszłościowych, zgodnych z tendencjami światowymi i potrzebami gospodarczymi kraju, kierunków zastosowań, a także odpowiedniego do ich realizacji sprzętu minikomputerowego.

Орловский Н.: Производство малых вычислительных машин а вычислительная техника в Польше

ИНФОРМАТИКА 1981, № 11—12, стр. 6

Подробная характеристика сегодняшнего состояния польского производства малых вычислительных машин и на этом фоне оценка развития вычислительной техники в Польше с указанием будущих, соответствующих мировым тенденциям и хозяйственным потребностям страны направлений применения, а также соответствующего для их реализации оборудования.

Zalewski J.: ADA — nowy język programowania

INFORMATYKA 1981, nr 11—12, s. 10

Pierwsza część prezentacji nowego języka programowania ADA. Podano charakterystykę możliwości funkcjonalnych tego języka, historię powstawania oraz aktualny stan jego realizacji.

Залевский Й.: АДА — новый язык программирования

ИНФОРМАТИКА 1981, № 11—12, стр. 10

Первая часть представления нового языка программирования АДА. Поданы характеристика функциональных возможностей этого языка, история появления и современное состояние его реализации.

Chmurzyński J., Liderman K.: Pomiary wydajności systemu komputerowego monitorem sprzętowym KL-80

INFORMATYKA 1981, nr 11—12, s. 13

Ogólna charakterystyka metod i zasad oceny działania systemów komputerowych w oparciu o stosowanie programowych i sprzętowych monitorów pomiarowych. Omówiono zalety i wady obu rodzajów monitorów. Bardziej szczegółowo omówiono monitor sprzętowy KL-80 skonstruowany w Wojskowej Akademii Technicznej w Warszawie, a także przykładowe wyniki przeprowadzonych przez autorów eksperymentów.

Хмужиньски Й., Лидерман К.: Измерения производительности системы автоматизированной техническим монитором KL-80

ИНФОРМАТИКА 1981, № 11—12, стр. 13

Общая характеристика методов и принципов оценки действования вычислительных систем на основе применяемых программных и технических измерительных мониторов. Более подробно обсуждается технический монитор KL-80 созданный в Военной Технической Академии в Варшаве, а также примерные результаты проведенных авторами экспериментов.

Chrobot S.: Statyczna i dynamiczna struktura w systemie operacyjnym SOM-5

INFORMATYKA 1981, nr 11—12, s. 18

Charakterystyka koncepcji projektowania strukturalnego programów współbieżnych, zastosowana w systemie operacyjnym SOM-5 dla minikomputera MERA 400. Bardziej szczegółowo omówiono pojęcie struktury dynamicznej programu oraz sposób jej definiowania w języku wyższego poziomu.

Хробот С.: Статическая и динамическая структура в операционной системе SOM-5

ИНФОРМАТИКА 1981, № 11—12, стр. 18

Характеристика концепции структурального проектирования согласованных программ, использованная в операционной системе SOM-5 для малой вычислительной машины MERA 400. Более подробно обсуждается понятие динамической струкутры программы с способ его определения в языке высшего уровня.

Bartyś M.: Język symboliczny i translator dla systemu CAMAC z procesorem 131

INFORMATYKA 1981, nr 11—12, s. 23

Charakterystyka języka ASCAM 131 oraz jego wykorzystywania do tworzenia oprogramowania zestawów aparatury pomiarowo-sterującej systemu CAMAC z procesorem autonomicznym typu 131. Bardziej szczegółowo omówiono strukturę i zasady działania procesora, translatora ASCAM 131 oraz zastosowanego systemu operacyjnego.

Бартыс М.: Символический язык и транслятор для системы CAMAC с процессором 131

ИНФОРМАТИКА 1981, № 11—12, стр. 23

Характеристика языка ASCAM 131 и его применения для создавания программного обеспечения измерительно-управляющей аппаратуры системы CAMAC с автономным процессором типа 131. Более подробно обсуждается структура и принципы действования процессора, транслятора ASCAM 131 и использованной операционной системы.

Gwiazda J.: Needs data processing a reform

INFORMATYKA 1981, No 11—12, p. 4

Critical evaluation of the polish data processing past development under circumstances of existing national economy management system. Characterized new tasks, main application directions and influence manner of data processing in the reformed system.

Gwiazda J.: Braucht die Datenverarbeitung einer Reform

INFORMATYKA 1981, Nr. 11—12, S. 4

Eine kritische Beurteilung der bisherigen Entwicklung polnischer Datenverarbeitung unter Bedingungen des bestehenden Nationalwirtschaftverwaltungsystems. Es wurden neue Aufgaben, grundsätzliche Anwendungsrichtungen und Beeinflussungsweise der Datenverarbeitung im reformierten System charakterisiert.

Orłowski H.: Minicomputer industry and data processing in Poland

INFORMATYKA 1981, No 11—12, p. 6

Detailed characteristics of actual state of the polish minicomputer industry and against this background the evaluation of data processing development in Poland pointing to future, consistent with world's trends and country's economic needs, application directions, as well as to suitable for their realization minicomputer hardware.

Orłowski H.: Die Kleinrechnerindustrie und die Datenverarbeitung in Polen

INFORMATYKA 1981, Nr. 11—12, S. 6

Eine ausführliche Charakteristik des heutigen Zustandes von der polnischen Kleinrechnerindustrie und auf diesem Hintergrund die Beurteilung von Datenverarbeitungentwicklung in Polen mit Hinweisung auf zukunftige, mit Welttendenzen und wirtschaftlichen Bedürfnissen des Landes übereinstimmende, Anwendungsrichtungen, sowie auf zu ihrer Realisierung angepasste Hardware.

Zalewski J.: ADA — a new programming language

INFORMATYKA 1981, No 11—12, p. 10

The new programming language ADA is presented. Functional possibilities of the langugae, the history of creation and present state of its realization are discussed.

Zalewski J.: ADA — eine neue Programmiersprache

INFORMATYKA 1981, Nr. 11—12, S. 10

Erster Teil einer Vorstellung der neuen Programmiersprache ADA. Es wurde eine Charakteristik der funktionellen Möglichkeiten dieser Sprache, die Geschichte ihrer Entstehung und jetziger Stand der Realisierung angegeben.

Chmurzyński J., Liderman K.: Efficiency measurement of computer system using hardware monitor KL-80

INFORMATYKA 1981, No 11—12, p. 13

General characteristics of methods and principles for computer system evaluation using software and hardware measuring monitors. Discussed advantages and disadvantages of both minitor kinds, and with more detail the KL-80 hardware monitor built on the Military Technical Academy in Warsaw, as well exemplary experiment results collected by the authors of the article.

Chmurzyński J., Liderman K.: Leistungsmessung der Rechnersysteme mit Hilfe des Hardwaremonitors KL-80

INFORMATYKA 1981, Nr. 11—12, S. 13

Eine Charakteristik der Methoden und Grundsätze für die Leistungsbeurteilung der Rechnersysteme mit Hilfe der Software- und Hardwaremessmonitoren. Es wurden die Vor- und Nachteile der beiden Monitorsarten besprochen. Mit grösseren Einzelheiten wurde der in der Technischen Militärakademie in Warszawa entwickelte Hardwaremonitor KL-80, sowie die Musterergebnisse der von den Autoren durchgeführten Experimente, besprochen.

Chrobot S.: Static and dynamic structure in the operating system SOM-5

INFORMATYKA 1981, No 11—12, p. 18

Characteristics of the concurrent programs structural design idea, which is applied in the operating system SOM-5 for the MERA 400 minicomputer. Discussed with more detail the idea of programs dynamic structure and the manner of its definition in high level language.

Chrobot S.: Statische und dynamische Struktur im Betriebssystem SOM-5

INFORMATYKA 1981, Nr. 11—12, S. 18

Eine Charakteristik der Konzeption von strukturellen Projektierung der gleichzeitig laufenden Programme, die im Betriebssystem SOM-5 für Kleinrechner MERA 400 angewendet wurde. Mit grösseren Einzelheiten wurden die dynamische Struktur des Programms und ihre Definierungsweise in höheren Programmiersprachen besprochen.

Bartyś M.: Symbolic language and translator for CAMAC system with 131 processor

INFORMATYKA 1981, No 11—12, p. 23

Characteristics of the ASCAM 131 language and its application for software creation of CAMAC system measuring and control device sets with autonomic 131 type processor. Discussed with more detail the structure and operating principles of the processor, the ASCAM 131 translator, as well of the applied operating system.

Bartyś M.: Symbolische Sprache und Übersetzer für das CAMAC System mit 131 Prozessor

INFORMATYKA 1981, Nr. 11—12, S. 23

Eine Charakteristik der ASCAM 131 Sprache und ihrer Ausnutzung für die Softwareherstellung des CAMAC Systems, mit den autononen Prozessor vom Typ 131. Mit grösseren Einzelheiten wurden die Struktur und die Wirkungsweise des Prozessors, des Übersetzers ASCAM 131 und des verwendeten Betriebssystem besprochen.

Informatyka

zastosowania w gospodarce, technice i nauce

Nr 11-12
MIESIĘCZNIK
1 9 8 1
ROK XVI
Listopad — grudzień

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

W NUMERZE: strona

Spis treści rocznika 1981 (wkładka)

Czy informatyce potrzebna jest reforma
Janusz Gwiazda 4

Przemysł minikomputerowy a informatyka w Polsce
Henryk Orłowski 6

ADA — nowy język programowania (1), Powstanie języka
Janusz Zalewski 10

Pomiary wydajności systemu komputerowego monitorem sprzętowym KL-80
Jerzy Chmurzyński, Krzysztof Liderman 13

Statyczna i dynamiczna struktura w systemie operacyjnym SOM-5
Stanisław Chrobot 18

Język symboliczny i translator dla systemu CAMAC z procesorem 131
Michał Bartyś 23

ALGORYTMY

Podręczna biblioteczka programisty
Andrzej Szałas, Zbigniew Świrski 27

Z KRAJU

Krajowy sprzęt dla informatyki na 53 MTP
Jacek Żebrowski 29

ZJEDNOCZENIE INFORMATYKI

Systemy informatyczne ZETO Poznań
Henryk Adamczewski 31

Metodyka tworzenia systemów informatycznych
Roman Ronkowski 34

ZE ŚWIATA

Najnowsze tendencje w dziedzinie komputerowych systemów sterowania
Janusz Zalewski 38

RECENZJE

Po(d)stępy bazorządctwa
Adam B. Empacher 40

TERMINOLOGIA

O pisowni nazw języków programowania
Janusz Zalewski 42

LISTY

Infologiczne problemy przetwarzania danych
Bogdan Stefanowicz 43

JANUSZ GWIAZDA
Sekretariat Komitetu Informatyki
Warszawa

Czy informatyce potrzebna jest reforma

Problemy reformy gospodarczej niezbyt często pojawiały się na łamach INFORMATYKI. Spróbujmy zatem, szkicowo — rzecz jasna — odpowiedzieć na trzy pytania:
— czy samej informatyce reforma jest potrzebna
— jakie są perspektywy usytuowania informatyki w zreformowanej gospodarce
— czy informatycy — ich umiejętności i narzędzia — mogą być przydatni podczas reformowania.

Próbując odpowiedzieć na pierwsze pytanie, warto podkreślić — być może niezbyt powszechnie zauważaną — istotną cechę sytemu nakazowo-rozdzielczego, która odcisnęła się szczególnym piętnem na informatyce. Spoiwem, które pozwala trwać silnie scentralizowanemu i dyrektywnemu sytemowi sterowania życiem państwa, warunkiem koniecznym takiego sposobu zarządzania jest ścisła reglamentacja informacji [1]). Reglamentacja rozumiana niemal dosłownie, a więc: precyzyjne określenie komu, ile i jakich informacji należy udzielić.

Z powyższej zasady wypływały wszystkie dalsze konsekwencje (organizacyjne, ekonomiczne i propagandowe) w informatyce. Niestety, także sami informatycy czynili starania, by wpasować komputerowe zastosowania w „piramidę rozdzielnictwa", licząc po cichu, że — przejąwszy kontrolę nad przetwarzaniem informacji — będą mogli stawiać warunki i rozwijać swoją dziedzinę. Przykłady niektórych systemów rządowych, o których można by powiedzieć, że rozkwitały na pustyni [2]), zdają się świadczyć o dobrym właśnie dopasowaniu.

Szczególnie charakterystycznym zjawiskiem, świadczącym za postawioną tezą, jest sytuacja systemów SPIS i SINTO. Systemy te również nosiły miano rządowych. Tu informatycy nie całkiem dopasowali się do modelu; roiła im się powszechność informowania, powszechny odbiór ich usług. SPIS-owi przyniosło to załamanie się możliwości finansiwych, a w efekcie — dalece niewystarczający sprzęt, oczywiście nie z kk, brak etatów, a nawet — próby przejmowania jego funkcji przez CENPLAN (obserwacja gospodarki) czy PESEL (obserwacja kadr: MAGISTER). System SINTO, który od początku deklarował otwartość, boć tylko wówczas miałby sens, na dobry ład nie doczekał się i narodzin. Wszystko co zdoałno własnym niemal sumptem zrobić (bo czymże jest niewiele ponad 100 mln w ciągu siedmiu lat takiego przedsięwzięcia) to w istocie dopiero przygotowania do budowy systemu.

Kończąc, dotknięty zaledwie problem, można z całą stanowczością stwierdzić, iż bez reformy — która, obalając system nakazowo-rozdzielczy, obali również zasadę bezwzględnego nadzoru przepływu informacji — zastosowania informatyki, powszechne, cenione i ekonomicznie uzasadnione, nie mają żadnych szans rozwoju. Wspomniane wyjątki lub zastosowania militarne nie mogą obalić tezy.

Drugim elemntem odpowiedzi na tytułowe pytanie jest problem przekształceń dotyczących nie otoczenia, w którym informatyka się rozwija, a przekształcenia jej samej — jej ekonomiki, organizacji, metod i technik. Być może poniższe stanowisko jest obarczone ograniczonym widzeniem przez pryzmat właśnie informatyki, wydaje się jednak, że:

- sytuacja w informatyce, gdyby ją z czymkolwiek porównywać, jest zachwaszczona w niezwykłym stopniu
- każde, choćby najbardziej radykalne i szybkie zmiany mogą jedynie poprawić jej stan, nic nie może zagrozić bardziej niż trwanie w obecnych układach i mechanizmach.

Dla poparcia pierwszego, mocnego zapewne stwierdzenia, wystarczy choćby przytoczyć znany szeroko (od roku) fakt ściąganej z zastosowań informatyki akumulacji, która równała się tej, jaką osiągał przemysł spirytusowy.

Sztywne, rozrośnięte, nieelastyczne struktury w informatyce (zjednoczenia, kombinaty, przedsiębiorstwa — tylko giganty), to antytezy dla działalności, w której sukces opiera się na elastyczności, dopasowaniu do stale zmieniających się warunków, pomysłowości ludzi i ich ruchliwości intelektualnej i — także — rozumianej dosłownie.

Uzasadnieniem zaś tego, że stwierdzenie drugie nie jest ekspresją nierozumnego optymizmu, będą uwagi na temat postawionego na wstępie pytania o perspektywy informatyki w reformie. W każdym jednak razie — totalne odejście od stosowania informatyki, nawet od tych jej nieimponujących początków w naszym kraju, nie jest możliwe; możliwe jest natomiast jej powolne obumieranie, gdyby proces zmian został zatrzymany.

●

Powszechne jest wśród informatyków przekonanie, że po reformie informatyka stanie wreszcie na twardym gruncie, że bez wymyślnych uzasadnień posiadać będzie prestiż i uznanie społeczne. Wtedy to pogardzone i wyśmiewane w publikatorach systemy obsługujące gospodarkę materiałową staną się w każdym przypadku potrzebne (da się nawet mówić o optymalizacji zapasów), każdemu zakładowi produkcyjnemu przydadzą się systemy rachunku kosztów, sterowania produkcją etc.

Rozwijanie tego tematu nie wydaje się celowe, jako że o kształcie naszej gospodarki po reformie można co najwyżej snuć przypuszczenia, niczego konkretnego bowiem (jeśli się nie jest wróżbitą) powiedzieć nie można.

Warto natomiast nieco uwagi poświęcić organizacyjnemu usytuowaniu informatyki i stosunkowi państwa (tu: administracji) do przyszłości zastosowań informatyki. Można zauważyć, że wszędzie tam, gdzie poziom gospodarczy jest wyższy niż w tzw. krajch rozwijających się, państwo wiele uwagi poświęca rozwojowi informatyki. Istnieje tam instytucjonalna forma obserwacji jej trendów, stymulowania jej rozwoju czy popierania niektórych ważnych — ze strategicznego [3]) punktu widzenia — kierunków. Dlatego też i w naszym kraju niezbędne jest istnienie instyucjonalnej formy wpływania przez rząd na kierunki zastosowań i wa-

[1]) Podobny wpływ daje się zaobserwować i w innych dziedzinach związanych z przekazywaniem informacji np. w łączności, informacji naukowo-technicznej, a nawet i słabości statystyki można doszukać się w tym samym źródle.
[2]) Parabola nie wydaje się przesadną, jeśli zważyć relacje między nasyceniem sprzętem i jego jakością tychże systemów a przeciętnym poziomem w kraju.

runkujące je sfery. Formy, treści i metody stymulacji muszą być inne niż sposoby dotychczasowe — hasła o koordynacji. Komitet Informatyki (nawet z najwyższym przewodniczącym na czele), przy braku egzekutywy choćby formalnej [4]) — nie miał żadnych możliwości wpływania na kierunki, jakość i zakres zastosowań.

Nowy sposób oddziaływania da się streścić następująco:

- wpływanie na całkowicie niezależnych wytrwóców dóbr informatycznych (sprzęt, opragramowanie(doradztwo itp.) poprzez mechanizmy ekonomiczne, informacyjne [5]), czy wreszcie — poprzez umowy rządowe
- oddziaływanie na pożądane społecznie (lub gospodarczo — w dalszej przyszłości) zastosowania informatyki znowu drogą stymulacji ekonomicznej lub dotacjami ze Skarbu Państwa, np. w oświacie i kształceniu, medycynie, kulturze.

Na zakończenie jeszcze garść problemów, w rozwiązaniu których informatycy mogliby pomóc, problemów, z którymi boryka się nasz kraj i z którymi będzie się podczas reformowania borykał. Rozpocząć chyba wypada od bodaj najważniejszych, których konsekwencje sięgają najdalej (budzi zresztą zdumienie fakt, iż dotychczas informatycy nie zostali tu zaangażowani).

Pierwszy — to samo opracowanie reformy i bardzo w tym przydatne — oprócz dyskusji — gry, symulacje, bilanse. Obszerna publikacja na ten temat, pełna pasji wypowiedź St. Albinowskiego (jeszcze wiosną br.) nie przyniosła niestety rezultatów. Zdumienie jest tym większe, że ani nie brakuje w Polsce ekonomistów i matematyków umiejących formułować zadania dla tego typu gier czy symulacji, ani też nie brak byłoby dla tego celu zdolnych informatyków (i sprzętu; tym bardziej), że choćby ośrodek obliczeniowy CENPLAN — delikatnie rzecz ujmując — nie narzeka na nadmiar pracy)

Drugim, równie (a może i bardziej) zasmucającym zjawiskiem jest nieobecność informatyki (znów np. w CENPLANIE czy rozsianych po Polsce ZETO) w działalności sztabów antykryzysowych. Trudno przyjąć, że dla ulżenia ciężkiej sytuacji kraju nie warto byłoby zmobilizować informatyków i zaangażować dostępne moce obliczeniowe. Nie można się wprawdzie łudzić, że w wyniku gier i bilansowania optymalnych oszczędności [6]) podniesie się gospodarkę z kryzysu, niemniej nie trzeba tu chyba uzasadniać, że jest to praktyka opłacalna.

Na marginesie powyższego nasuwa się refleksja: jakże głęboka jest nieufność władz do informatyki (czy może do informatyków). Nawet obecnie uważają one, iż lepiej mieć z informatyką jak najmniej do czynienia.

Ostatni problem, o którym będzie tu mowa to zadanie, które można rozwiązać bez oczekiwania na zew, które wymaga od ośrodków obliczeniowych jedynie inicjatywy, pomysłowości i rzetelności, od ośrodków, które uznałyby, że jest to dla nich szansa. Szansą tą jest oferowanie przedsiębiorstwom informacji o materiałach oraz możliwościach kooperacji i najkorzystniejszych zakupów. Już obecnie taka działalność może przynieść rozwój ośrodka, warunkiem jest jednak posiadanie właściwie wyposażonego komputera oraz elastyczne dostosowywanie się do bieżących potrzeb i wymagań użytkownika.

●

Na koniec drobne usprawiedliwienie faktu, że nie wspomniano tu o potrzebie uczestnictwa niezłych przecież informatyków w wypracowywaniu koncepcji reformy (choćby w pracach Komisji ds. Reformy Gospodarczej). Istotnie, o ile autorowi wiadomo — w pracach zespołu ani jeden informatyk nie uczestniczy. Nie znaczy to jednak, że środowisko informatyków zamarło w oczekiwaniu, wprost przeciwnie — przykładem choćby dziejące się już zmiany w ZETO, ETOBIE i wielu innych, aktywność w pracach nad kształtem modelu gospodarczego stowarzyszeń naukowych czy związków zawodowych. Wielkość tych działań usprawiedliwia rezygnację z próby ujęcia w jednym omówieniu całości zjawiska.

---

[3]) Nie należy rozumieć tego jedynie w sensie militarnym.

[4]) Komitet Informatyki w rzeczywistości nie ma żadnej władzy nad resortami, między które rozparcelowano odpowiedzialność za informatykę.

[5]) Dla przykładu — informowanie o zamierzonych preferencjach i ich perspektywach.

[6]) Optymalnych w znaczeniu przynoszących najmniejsze per soldo straty.

Pogody, uspokojenia i odrobiny dystansu do rzeczywistości, która nie bardzo daje się kochać i która tak łatwo nas od siebie uzależnia. Ciepła przy uboższym jeszcze Świątecznym stole. A ponadto - zdecydowania w potyczkach o lepszy przyszły rok

życzy Czytelnikom

REDAKCJA

HENRYK ORŁOWSKI
Instytut Maszyn Matematycznych
Warszawa

Przemysł minikomputerowy a informatyka w Polsce

Czyniąc zadość prośbie Redakcji INFORMATYKI dzielę się z Czytelnikami swoimi poglądami w sprawach stanu obecnego oraz pożądanych kierunków rozwoju przemysłu komputerowego i informatyki w naszym kraju. Wypowiedź ograniczam do tematyki minikomputerowej, jako bliskiej moim zainteresowaniom zarówno osobistym jak i zawodowym.

Podkreślam rozróżnienie pomiędzy przemysłem komputerowym a informatyką. Przemysł komputerowy to dostawca środków, narzędzi pracy dla informatyki — tak z zakresu sprzętu jak i oprogramowania. Natomiast informatyka to nauka i gałąź gospodarki zajmująca się wykorzystaniem komputerów, a często i zmianą otoczenia w celu efektywnego wykorzystania technik komputerowych (np. zmiana metod zarządzania lub struktur organizacyjnych).

Analogiczne rozróżnienie występuje np. pomiędzy przemysłem zajmującym się produkcją samochodów, a inżynierią ruchu drogowego. Jakkolwiek dziedziny te w wielu miejscach zazębiają się i nie pozostają bez oczywistego wpływu na siebie, to stosują różne metody w pracach badawczych i rozwojowych, a także w praktyce gospodarczej, wymagają pracowników o różnych kwalifikacjach i co jest bardzo istotne, różne mogą być cele ich optymalnego działania. Uważam że wiele nieporozumień, a nawet napięć powstających wewnątrz środowiska można by uniknąć, gdyby w wypowiedziach i działaniach świadomie rozróżniano przemysł komputerowy od informatyki.

Stan obecny przemysłu minikomputerowego

W połowie lat siedemdziesiątych ujawniły się dwa negatywne czynniki dla rozwoju i działania przemysłu komputerowego w naszym kraju. Po pierwsze — władze centralne głęboko rozczarowane efektami wdrażania szeregu wielkich systemów informatycznych (informatyka nie stała się kluczem do dobrobytu), a świadome kosztów (w tym dewizowych) poniesionych na te systemy — popadły w drugą skrajność. Uznały bowiem, że nie należy dalej popierać — także finansowo — informatyzacji gospodarki narodowej. Po drugie — uświadomiono sobie, że dynamiczny rozwój programu inwestycyjnego musi być zahamowany, że trzeba skierować środki na produkcję rynkową (dla zrównoważenia wypłat uczynionych w trakcie inwestowania) i produkcję proeksportową (dla spłaty zadłużenia). Manewr gospodarczy bynajmniej nie objął wszystkich dziedzin inwestowania, niewątpliwie jednak dotknął informatykę i przemysł komputerowy. Produkcję środków informatyki zaliczono do produkcji dóbr inwestycyjnych i ostro zalimitowano dostawy dla kraju.

Zgodnie z ideologią „manewru", w Ministerstwie Przemysłu Maszynowego popierano tylko dwa kierunki działania: produkcję rynkową i eksport, tolerowano tzw. produkcję kooperacyjną, natomiast inwestycyjną — eliminowano. W tej sytuacji przed przemysłem komputerowym stanęła alternatywa: albo przestawić się na eksport, albo przestać istnieć. Przemysł wybrał pierwsze wyjście.

Spośród trzech możliwych kierunków eksportu (systemy użytkowe, zestawy komputerowe oraz sprzedawane luzem urządzenia peryferyjne i pamięci) przemysł minikomputerowy z powodzeniem zrealizował trzeci. Systematycznie zwiększano produkcję, tak że w 1980 r. wyprodukowano w kraju ok. 19 tys. urządzeń peryferyjnych i pamięci, z czego wyeksportowano ok. 13 tys. Eksport do strefy dolarowej rozwinięto na tyle, że w roku 1980 sprzedano za większą kwotę niż wynosiło zapotrzebowanie dewizowe na zakup elementów i materiałów z II obszaru płatniczego dla całej produkcji Zjednoczenia MERA. Realizowany jest duży i bardzo opłacalny eksport do I obszaru płatniczego.

Sukces został tu osiągnięty zarówno dzięki licencjom (w szczególności drukarki mozaikowe i pamięci na dyskach elastycznych) oraz własnym opracowaniom (pamięci taśmowe kasetowe, ferrytowe pamięci operacyjne, stacje czytnik — perforator taśmy papierowej), jak i rozwojowi postlicencyjnemu (zwłaszcza monitory ekranowe). Zakłady MERA-BŁONIE stały się największą fabryką drukarek nie tylko w RWPG, ale i w Europie.

O ile zatem uważam za — sumarycznie biorąc — bardzo udane poczynania przemysłu minikomputerowego w produkcji urządzeń peryferyjnych i pamięci, to osiągnięcia w produkcji systemów użytkowych i zestawów minikomputerowych są znacznie mniejsze, mniejsze od naszych ambicji i potencjalnych możliwości. Nie jest to jednak — jak sądzą niektórzy — produkcja znikoma. W 1980 r. wyprodukowano ok. 800 zestawów mini- i mikrokomputerowych (wymieniając od największych do najmniejszych): MERA 400, SM 3, MERA 60, MERA 200, MERA 2500, MERA 100 i MERA 9150. Z tej produkcji odbiorcy krajowi otrzymali tylko ok. 300 zestawów, w tym najbardziej znanych MERA 400 — ok. 170. Małe dostawy dla kraju są zapewne przyczyną tego, że fakt dużej w rzeczywistości produkcji przemysłu komputerowego nie jest znany.

Co stanęło zatem na przeszkodzie znaczniejszemu rozwojowi produkcji mini- i mikrokomputerów na eksport? Podstawową przyczyną jest niedorozwój naszego przemysłu elektronicznego. Wobec fantastycznego wzrostu stopnia scalania elementów elektronicznych w ostatnich latach, przemysł komputerowy nie mający dostępu do nowoczesnych elementów jest skazany na brak rozwoju (nie chciałbym prorokować, że na zagładę).

Doc. dr inż. HENRYK ORŁOWSKI ukończył w 1958 r. studia na Wydziale Łączności Politechniki Wrocławskiej. Pracował w CBKO-Pruszków i w Instytucie Elektrotechniki w Międzylesiu, zajmując się sterowaniem numerycznym obrabiarek, a od 1962 r. wykorzystaniem komputera ELLIOTT 803. W latach 1966—1978 w Przemysłowym Instytucie Automatyki i Pomiarów zajmował się zastosowaniami komputerów w automatyce. Od 1979 r. jest dyrektorem Instytutu Maszyn Matematycznych w Warszawie i Głównym Konstruktorem Systemu Małych EMC w Polsce.

Dla osiągnięcia produkcji na wymienionym poziomie 800 zestawów rocznie, radzono sobie różnymi sposobami:

- w celu zrealizowania eksportu do II obszaru płatniczego, kupowano najnowsze elementy i z nich montowano mikrokomputery (MERA 2500, MERA 200, w ubiegłych latach MERA 100); z uwagi na duży wkład dewizowy minikomputery te nie trafiały na rynek krajowy
- w celu zrealizowania eksportu do I obszaru płatniczego, aby utrzymać się w sensownych relacjach cenowych, a także by zapewnić kompatybilność z rodziną SM — kupowano procesory ze Związku Radzieckiego i na ich bazie montowano systemy, które następnie eksportowano wraz z oprogramowaniem (SM 3, MERA 60)
- wykonywano systemy dla tych odbiorców krajowych, którzy mogli przekazać własne dewizy na zakup elementów (MERA 9150)
- pozostałe przypadki eksportu (poza unikalnymi) dotyczyły systemów użytkowych sprzedawanych w tak dużych konfiguracjach sprzętowych i programowych, że koszt jednostki centralnej przestawał ważyć w cenie całego zestawu (na przykład: MERA 400 dla sterowania rafinerii w ZSRR, ODRA 1325 dla CRPD [1]) w cukrowni na Węgrzech). Były również konkretne zamówienia, np. z Włoch na zakup większej liczby egzemplarzy MERA 400, ale cena za jaką Włosi (i trudno im się dziwić) byli skłonni kupować te zestawy wynosiła poniżej połowy naszych kosztów produkcji. Jest to jeszcze jedno potwierdzenie tezy, że bez dostępu do elementów o wysokim stopniu scalenia (w tym — pamięci półprzewodnikowych) o cenach i o jakości odpowiadających zwyczajom światowym, nie ma szans na produkcję opłacalnych jednostek centralnych mini i mikrokomputerów.

Drugą przyczyna ograniczającą nasz eksport zestawów minikomputerowych do krajów RWPG był fakt, że wszystkie te kraje uruchomiły u siebie produkcję minikomputerów i w efekcie ze zrozumiałych względów nie chcą ich kupować u nas. Jeżeli wymienione przyczyny można uznać za obiektywne (przynajmniej z punktu widzenia przemysłu komputerowego), to wystąpił także szereg przyczyn subiektywnych.

Przemysł komputerowy podjął brzemię produkcji i rozwoju MERY 400 jako owoc niesławnego zakończenia „sprawy K-202". W rezultacie:

- dzielono finanse i potencjał wykonawców na dwie linie oprogramowania (SM — dla umożliwienia eksportu, MERA 400 — aby nie sprzedawać zupełnie „gołego" następcy K-202)
- nie wykorzystano, na rzecz głównego minikomputera produkowanego w kraju (tj. MERY 400) oprogramowania, efektów szkolenia i literatury fachowej związanych z zakupem przez Polskę w latach siedemdziesiątych kilkudziesięciu zestawów PDP-11.
- dorobek krajowych użytkowników systemów minikomputerowych rozwijany na bazie MERY 400, nie procentował w eksporcie. Nabywcy z krajów RWPG reflektowali na systemy minikomputerowe w linii SM.

Ten ostatni rezultat oceniam jako szczególnie niekorzystny. Wiadomo bowiem, że bardziej opłacalny jest eksport systemów użytkowych niż zestawów, a tym bardziej urządzeń peryferyjnych. Tymczasem systemy minikomputerowe w kraju opracowywano i opracowuje się głównie w oparciu o MERĘ 400. Gdyby natomiast opracowywano je w oparciu o sprzęt SM, wtedy mielibyśmy szerszy wachlarz ofert na eksport do I obszaru płatniczego (obecnie tylko systemy CAMAC), jak również moglibyśmy też eksportować systemy do II obszaru płatniczego, zastępując w eksportowanych systemach jednostki centralne SM oryginalnymi PDP-11 i mając tym samym rozwiązane problemy ceny (o czym wspomniałem wyżej), niezawodności, części zamiennych i serwisu na odległych rynkach zbytu.

Inna przyczyna ograniczenia naszego eksportu wystąpiła w związku z MERA 100 i MERA 200. Bardzo zasłużone w produkcji drukarek Zakłady MERA-BŁONIE, przygotowując eksport tych mikrokomputerów do krajów kapitalistycznych, nie zadbały zawczasu, aby jednocześnie z wprowadzeniem na rynek sprzętu zaoferować bogate oprogramowanie. Ten błąd zmniejszył skuteczność oferty. Z uwagi na właściwości wymienionych mikrokomputerów potrzebne było przy tym nie tylko oprogramowanie podstawowe, ale i pakiety programów użytkowych oraz systemy skrośnej generacji programów.

Poza wymienionymi przyczynami, ograniczającymi skuteczność działania przemysłu jest jeszcze szereg innych, charakterystycznych dla całej naszej gospodarki w jej dotychczasowym trybie działania.

- Cała produkcja przemysłu była nastawiona na produkcję finalną gotowych wyrobów, z wyraźnym ograniczeniem produkcji części zamiennych. Z innej branży wiadomo, że do prac rolnych w roku 1981 nie wyjechało na pola z braku części zamiennych ponad 40 tys. ciągników. Nie znam liczb odnośnie zestawów minikomputerowych i urządzeń peryferyjnych, ale ich przestoje spowodowane brakiem części zamiennych są bardzo duże.
- Relikty XIX-wiecznych doktryn ekonomicznych ciążące na naszym życiu gospodarczym spowodowały, że prowadzone przez przemysł prace programistyczne, projektowe i serwisowe są traktowane jako „narzut na robociznę bezpośrednią w sferze wytwarzania", co powoduje ich stałe ograniczanie. Liczne wskaźniki obowiązujące przemysł (na przykład stosunek liczby pracowników bezpośrednio produkcyjnych do pośrednio produkcyjnych i zaplecza rozwojowego) zmuszały w konkretnym przypadku zestawów minikomputerowych do działania przemysłu przeciw interesom użytkowników, a więc i gospodarki narodowej traktowanej jako całość.
- Niedoskonałości mechanizmów obrotu handlowego w ramach RWPG (nad którymi z uwagi na profil tematyczny INFORMATYKI nie będę tutaj się rozwodził) powodują, że jak dotychczas nie można wyzyskać wszystkich potencjalnych korzyści, jakie daje jednolitość systemu minikomputerów produkowanych w krajach RWPG (np. swobodne konfigurowanie zestawów z urządzeń produkowanych w różnych krajach, obrót oprogramowaniem itp).
- Konieczność wykonywania planów „za wszelką cenę", co powoduje produkcję zestawów konfigurowanych według możliwości produkcyjnych, a nie potrzeb użytkowników (stąd m.in. na ogół zbyt szczupłe pamięci operacyjne) oraz obniżenie jakości poniżej możliwości konstrukcyjnych i opanowanych technologii wytwarzania. Wiąże się z tym obniżanie jakości wyrobów, którego główne przyczyny są następujące:

— deficyt materiałów do produkcji (w szczególności elementów elektronicznych) powodujący ograniczanie ich starzenia i selekcjonowania

— zakup elementów z zagranicy odbywa się nie od stałych, tych samych dostawców, lecz w sposób przypadkowy, w zależności od tego, w jakim kraju bank ma nadwyżkę dewizową lub otworzył „linię kredytową"

— brakujące elementy przychodzą często w ostatniej chwili, skutkiem tego zestawy są kończone w pośpiechu, a zwłaszcza nie są dotrzymywane reżimy starzenia i testowania; wyjaśnia to zjawisko, dlaczego niektóre zestawy minikomputerowe pracują u użytkowników bardzo dobrze, a inne bardzo źle.

Tym co napisałem wyżej nie chcę sugerować, że jestem zadowolony z obecnego stanu przemysłu minikomputerowego w kraju, a przyczyn „ewentualnych, przejściowych niedoskonałości" upatruję w czynnikach zewnętrznych. W każdej sytuacji można pracować lepiej lub gorzej, dokonywać lepszych lub gorszych wyborów. Uważam jednak, że stan obecny naszego przemysłu minikomputerowego jest znacznie lepszy niż sądzi wielu informatyków.

Perspektywy przemysłu minikomputerowego

Dla zachowania i rozwoju potencjału przemysłu minikomputerowego, należy — jak sądzę — utrzymać jego proeksportowy charakter. W tym zakresie należy rozwijać naszą produkcję urządzeń peryferyjnych i pamięci pomocniczych, z przeznaczeniem głównie na eksport do krajów RWPG. Nie uważam za możliwe — przy ograniczeniach inwestycyjnych — rozszerzenie asortymentu urządzeń, natomiast należy polepszać parametry już produ-

[1]) CRPD — centralna rejestracja i przetwarzanie danych

kowanych, zwiększać produkcję (w tym drogą zmian konstrukcyjnych i technologicznych), eliminować import materiałów i elementów, kompleksowo załatwiać problemy (np. należy w Polsce uruchomić produkcję dysketek i dysków kasetowych, skoro produkujemy pamięci je wykorzystujące). Należy też z odpowiednim wyprzedzeniem przygotowywać nowe urządzenia w asortymentach dotychczas produkowanych.

W celu rozwinięcia produkcji urządzeń peryferyjnych i pamięci, a w szczególności zestawów minikomputerowych, niezbędne jest opanowanie w kraju produkcji profesjonalnych elementów elektronicznych o wysokim i bardzo wysokim stopniu scalenia. Uważam, że w tym zakresie jest możliwe i wręcz niezbędne ścisłe współdziałanie tzw. zaplecza przemysłu elektronicznego i komputerowego. O ile bowiem przemysł elektroniczny dysponuje już technologiami, które umożliwiłyby wytwarzanie takich elementów, to w bardzo ograniczonym stopniu posiada możliwość projektowania i testowania. Stąd czas oczekiwania na uruchomienie produkcji nowych elementów jest bardzo długi i z przyczyn ekonomicznych nie ma możliwości produkowania elementów w krótkich seriach, to jest w takich ilościach, jakie są potrzebne przemysłowi komputerowemu.

Projektowanie i testowanie nowoczesnych elementów można rozwiązać tylko metodami komputerowymi, zatem jest rzeczą naturalną, aby zaplecze naukowo-badawcze przemysłu komputerowego przyczyniło się do rozwiązania tych problemów. Uważam zatem, że dla tego zaplecza udział w przygotowaniu wytwarzania nowoczesnych elementów elektronicznych jest — obok rozwoju peryferii i pamięci pomocniczych — najważniejszy. Dopiero mając możliwości projektowania i otrzymywania elementów zaspokajających własne potrzeby, można na serio opracowywać i produkować nowe jednostki centralne i zestawy minikomputerowe, które zastąpiłyby generację mini- i mikrokomputerów aktualnie produkowanych w Polsce.

Z punktu widzenia bieżących potrzeb naszej informatyki, poza wyposażeniem jej w nośniki informacji i inne materiały eksploatacyjne, uważam, że przemysł zrobiłby najwięcej dobrego dowartościowując zestawy już istniejące. To dowartościowanie powinno polegać między innymi na:

- remoncie wszystkich urządzeń, które wykazują MTBF [2]) poniżej dobrego
- zapewnieniu sprawnego serwisu wymieniającego pakiety lub całe podzespoły, naprawiane następnie w warunkach przemysłowych u producenta. Nie widzę powodów (oczywiście po zmianie obecnego systemu przepisów gospodarczych), dla których skuteczna naprawa miałaby nastąpić później niż nazajutrz po zgłoszeniu
- należy uzupełnić już istniejące zestawy w większe pamięci operacyjne, dodatkowe monitory i pamięci pomocnicze, a w przypadku silnych minikomputerów (jak MERA 400) należy upowszechnić pracę w trybie wieloprogramowym i wielodostępnym; postępując tą drogą osiągniemy z określonej ilości materiałów i dewiz znacznie większe efekty użytkowe niż produkując nowe zestawy; oddzielnym zagadnieniem jest konieczność wprowadzenia takich mechanizmów gospodarczych, aby postulowana działalność opłacała się samorządnym załogom przemysłu komputerowego
- ponieważ zainstalowane w kraju zestawy PDP-11 w większości przypadków występują także w konfiguracjach kadłubowych, zbyt skąpych jak na potrzeby użytkowników, to postuluję, aby skromne środki dewizowe jakimi użytkownicy będą dysponowali w najbliższych latach, poświęcić nie na zakup gotowych bloków za granicą, ale na wyprodukowanie dodatkowych urządzeń w kraju celem ich przyłączenia do zakupionych już PDP-11. Można to zrealizować, wykorzystując prace prowadzone nad minikomputerami SM w Zakładzie Doświadczalnym IMM
- należy polepszyć jakość oprogramowania oraz rozszerzyć zakres oprogramowania rozprowadzanego centralnie (odnosi się to w szczególności do MERY 400)

[2]) MTBF — Mean Time Between Failures (średni czas pomiędzy awariami)

- odnośnie produkcji i rozwoju systemów minikomputerowych — uważam, że aktualnie powinniśmy położyć główny nacisk na zestawy kompatybilne z PDP-11.

Mamy tu następujące możliwości:

— MERA 60 (SM 50/50-3) jest odpowiednikiem LSI-11/2; produkowanym przez MERA-STER w Katowicach z wykorzystaniem radzieckich jednostek centralnych

— SM 4 jest stosunkowo bliskim odpowiednikiem PDP-11/40. Może być kupowany w kompletnych zestawach ze Związku Radzieckiego. Jest także kompletowany przez MERA-CENTRUM w Warszawie z wykorzystaniem radzieckich procesorów

— SM 50/50-1 jest odpowiednikiem PDP-11/34. Prototyp został opracowany w Instytucie Maszyn Matematycznych w Warszawie przy udziale MERA-PIAP. Aktualnie przewidywany jest następujący obszar zastosowania tego minikomputera: w sieciach terminali bankowych produkowanych przez ELWRO, w nowym systemie zbierania danych z rozszerzonymi możliwościami lokalnego przetwarzania (następca MERY 9150) oraz dla szczególnie odpowiedzialnych zastosowań (sterowanie w czasie rzeczywistym), dla których parametry niezawodnościowe SM4 byłyby niewystarczające; dla tych ostatnich zastosowań zestawy będzie produkował Zakład Doświadczalny IMM.

Wszystkie trzy wymienione tu minikomputery są kompatybilne programowo, a ponieważ ponadto można na nich eksploatować oprogramowanie użytkowe opracowane na minikomputerach PDP-11, stwarza to dla użytkowników wiele korzyści.

W moim przekonaniu głęboko niesłuszny jest pogląd, że w przypadku produkowania sprzętu komputerowego kompatybilnego z wyrobami znanych firm, należy zaniechać własnych prac nad oprogramowaniem podstawowym, a ograniczyć się tylko do adaptacji istniejącego oprogramowania. Taki pogląd zrobił nam dużo szkody w zastosowaniach rodziny RIAD. Ponieważ parametry techniczne naszego sprzętu, posiadane kinfiguracje i relacje ekonomiczne różnią się u nas od oryginalnych wzorców, to oczywiste jest, że na przykład systemy operacyjne optymalne dla oryginalnych zestawów, wcale nie muszą być najlepsze dla naszych.

Po drugie, kto nastawia się na adaptowanie cudzych rozwiązań, ten z góry skazuje się na to, że będzie miał produkty opóźnione w. rozwoju, nienowoczesne. Dlatego uważam, że zarówno ośrodki programistyczne przemysłu jak i zespoły pracujące poza przemysłem powinny bezwzględnie pracować nad oryginalnym oprogramowaniem SM mając tą dodatkową korzyść, że udane produkty mogą zdobywać dodatkowe rynki zbytu w krajach RWPG na maszynach SM oraz w innych krajach na maszynach PDP-11 lub o tej samej liście rozkazów.

Odnośnie minikomputerów MERA 400, to należy wyciągnąć pragmatyczny wniosek z faktu, że jest to minikomputer o dużych możliwościach obliczeniowych, którego produkcja jest opanowana i który został już sprzedany w liczbie ponad 400 egzemplarzy. Uważam zatem, że poprzednie uwagi dotyczące dowartościowania istniejących zestawów, powinny w szczególności odnosić się do tego minikomputera. Natomiast nie zalecałbym opierania na nim nowych, ciekawych systemów użytkowych, ponieważ takie systemy powinny bazować na minikomputerach z linii SM, zapewniających szerokie możliwości eksportu.

Oddzielnym zagadnieniem jest sprawa minikomputera perspektywicznego, który mógłby stać się następcą minikomputerów aktualnie produkowanych w kraju lub importowanych. Odnośnie cech tego minikomputera to uważam za konieczne przyjęcie następujących podstawowych założeń:

- powinien mieć znacznie większe możliwości obliczeniowe od modeli aktualnie produkowanych, ponieważ za kilka lat miejsce obecnych minikomputerów zajmą mikrokomputery lub programowane kalkulatory
- powinien należeć do rodziny SM, co zapewni możliwość jego eksportu (chociażby w zestawach) w ramach krajów RWPG
- powinien akceptować oprogramowanie użytkowe zarówno dotychczasowych SM jak i MERY 400, aby nie pozostawić użytkowników „na lodzie"

• nie może być kopią lub analogiem kolejnego modelu firmy DEC, ponieważ w związku z nowymi rozwiązaniami w zakresie technologii elektronowej opracowanie i produkcja takich kopii w naszych warunkach byłaby ekonomicznie nieopłacalna, nie mówiąc o innych przeciwskazaniach.

Najbardziej dojrzałą propozycją w tym zakresie jest minikomputer, którego koncepcja została przedstawiona na seminarium w Instytucie Maszyn Matematycznych w Warszawie w dniu 24 kwietnia 1981 roku (patrz INFORMATYKA nr 7—8/81). Jest rzeczą oczywistą, że perspektywy jego produkcji zależą w zasadniczy sposób od rozwiązania problemu bazy elementowej. Muszą zostać także rozwiązane problemy dotychczas utrudniające lub uniemożliwiające prace użytkownikom, takie jak: zła jakość sprzętu, braki materiałów eksploatacyjnych i wyposażenia, serwis, uporządkowana dystrybucja oprogramowania.

Stan obecny i kierunki rozwoju informatyki

Mój pogląd „obserwatora z przemysłu" może być w tym zakresie nader subiektywny. Uwagi ograniczam znów tylko do mini- i mikrokomputerów.

Na pewno stan ilościowy zastosowań mini- i mikrokomputerów jest znacznie poniżej osiągniętego przez nasz kraj poziomu cywilizacyjnego i gospodarczego oraz poniżej ambicji i możliwości polskich informatyków. Za udane, niejako wzorcowe w naszym kraju, uważam zastosowania minikomputerów WANG w obliczeniach i przetwarzaniu oraz PDP-11 w sterowaniu.

Ogólnie biorąc — minikomputery nie odgrywają u nas jednak należnej roli z uwagi na ich znikomo małą liczbę w skali kraju oraz trudności, jakie napotykają użytkownicy w pracy, głównie z uwagi na brak nośników informacji (dyskotek, kaset dyskowych, ostatnia także papieru), kadłubowość konfiguracji, jak również nieoperatywność serwisu. Ewidentnym brakiem jest znikome wyposażenie w sprzęt mini- i mikrokomputerowy stanowisk pracy projektantów (brak urządzeń typu *„personal computers"*) oraz stanowisk pracy biurowej, magazynów, przychodni lekarskich, kas itp. Relatywnie mały udział minikomputerów w zastosowaniach wynika z ich wysokiej ceny, a jednocześnie niskiej jakości. Oba czynniki w głównej mierze są skutkiem stosowania przestrzałej bazy elementowej.

Jeszcze raz podkreślam — rozwój produkcji nowoczesnych elementów scalonych o wielkim stopniu integracji u nas uważam za zagadnienie kluczowe, zresztą nie tylko z uwagi na potrzeby informatyki. Jest to zagadnienie strategiczne dla przyszłości kraju. Podzielam pogląd tych specjalistów, którzy uważają, że kraje nie dysponujące własnymi możliwościami projektowania i produkowania elementów scalonych w ciągu kilkunastu lat spadną (jeżeli już się tam nie znajdują) do stanu krajów neokolonialnych.

Jeżeli założymy, że problem bazy elementowej zostanie rozwiązany, to w zakresie minikomputerów zadania informatyki są następujące:

• Wyposażenie wyższych uczelni i szeregu szkół średnich w minikomputery o nowoczesnej architekturze i oprogramowaniu. Jest absurdem pedagogicznym i ekonomicznym, jeżeli przyszłych użytkowników i decydentów kształci się na sprzęcie przestarzałym (często „złomowanym" w resortach gospodarczych), a następnie oczekuje się, że pracownicy gospodarki będą rozsądnie kupowali i efektywnie użytkowali sprzęt nowoczesny. W dotychczasowej, z gruntu błędnej, a w żadnym nowoczesnym kraju nie stosowanej koncepcji szkolenia informatyków upatruję jedną z istotnych przyczyn niepowodzeń szeregu zastosowań kosztownego sprzętu informatycznego zakupionego w latach siedemdziesiątych ze strefy dolarowej. Przypomnijmy jak sensownie, w ramach naszych możliwości, była zaprojektowana i oprogramowana ODRA 1204 przez pokolenie wykształcone na GIERZE i ELLIOTTA'ch, maszynach najnowoczesniejszych w swojej klasie na świecie w chwili zakupu dla naszych uniwersytetów.

• Biura projektowe, działy przygotowania produkcji w fabrykach, magazyny itp. powinny być wyposażone w „personal computers", zbudowany na bazie mini- i mikrokomputerów o słowie 16-bitowym i liście rozkazów PDP-11.

Zapewne większość zastosowań będzie realizowana na radzieckich procesorach ELEKTRONIKA-60 (tzn. minikomputery MERA 60 i ich pochodne), natomiast bardziej złożone — na krajowym minikomputerze SM 50/50-1. Oczywiście istotne jest prawidłowe skompletowanie urządzeń zewnętrznych i pamięci pomocniczych pod kątem potrzeb poszczególnych użytkowników, a także opracowanie parametryzowanych pakietów programów użytkowych. Ideałem byłoby, aby użytkownik otrzymał zestaw oprogramowany dla jego zastosowań, na którym mógłby wykonywać swoje zadania bez konieczności organizowania „ośrodka obliczeniowego" i zatrudnienia „nadwornych" programistów. W tym celu konieczne jest znaczne rozbudowanie, lub wręcz zorganizowanie działów obsługi klienta i oprogramowania użytkowego u dostawców minikomputerów. Należy też pomyśleć o nowych, bardziej wydajnych formach produkcji oprogramowania użytkowego oraz projektowania systemów. Za najbardziej wydajny sposób rozwiązania problemu uważam organizowanie kilku lub kilkunastoosobowych spółdzielni pracy, które zrzeszałyby projektantów systemów i programistów, realizujących zamówienia u siebie w domu, lub u klienta. Przedsięwzięcie takie byłoby bezinwestycyjne, z minimalnymi narzutami w kosztach bieżącej działalności. Użytkownicy opłacaliby programistów tylko wtedy, gdy są oni rzeczywiście potrzebni, a nie — jak to się praktykuje obecnie — od chwili zakupu komputera, aż po kres jego eksploatacji.

Jeszcze raz podkreślam, że uważam za konieczne, aby minikomputery dotarły do stanowisk pracy użytkowników, a nie stanowiły zalążek organizowania wydzielonych ośrodków obliczeniowych.

• Banki, kasy oszczędnościowe, biura sprzedaży i rezerwacji biletów, kasy w dużych domach towarowych powinny być wyposażone w terminale lub skomputeryzowane kasy, połączone przez koncentratory — zależnie od potrzeb — do minikomputerów lub dużych komputerów. Do pełnienia funkcji koncentratorów i minikomputerów dla tych zastosowań widzę SM 50/50-1, natomiast jako duże komputery — R 32 oraz jego następcę. Produkcja terminali bankowych jest obecnie uruchomiona w ELWRO.

• Z uwagi na niską jakość procesorów, jakie możemy kupować w krajach RWPG, szczególnie trudna jest sprawa rozwoju systemów czasu rzeczywistego o wysokim stopniu niezawodności. Bardzo proste systemy można oczywiście budować na odpowiednikach INTEL-a 8080, co jest robione w wielu ośrodkach w kraju. Odnośnie systemów bardziej złożonych widzę dwie sensowne drogi, a mianowicie:

— dla systemów przeznaczonych do obsługi obiektów charakteryzujących się wielkimi obrotami materiałów lub energii można wykonywać jednostkowo minikomputery SM 50/50-1 o podwyższonej jakości, stosując specjalne technologie starzenia i selekcji elementów oraz testowania podzespołów. Oczywiście takich zastosowań nie może być dużo, zarówno z uwagi na znaczne koszty budowanego w ten sposób minikomputera, jak i trudności z zakupem materiałów. Tą drogą można rozwiązać także problemy koniecznej wymiany i uzupełnień wielu systemów zakupionych w latach siedemdziesiątych w krajach kapitalistycznych dla różnych działów naszej gospodarki;

— dla pozostałych zastosowań (np. oddziałów intensywnej terapii w szpitalach, sterowania w rolnictwie) musimy poczekać na nowoczesne elementy o wysokim i bardzo wysokim stopniu scalenia i dobrej niezawodności. Bez tych elementów nie wyprodukujemy niezawodnych i tanich minikomputerów w ilości potrzebnej dla omawianych zastosowań.

Czytelnicy uznają zapewne niektóre poglądy tutaj zawarte za mało odkrywcze, ale sądzę, że mają prawo wiedzieć, z jakimi poglądami autor się identyfikuje.

JANUSZ ZALEWSKI
Instytut Badań Jądrowych
Warszawa

ADA – nowy język programowania (1)

Powstanie języka

Stosunkowo niedawno uświadomiono sobie, że przeważającą część nakładów związanych z wprowadzeniem i wykorzystaniem komputerów przeznacza się na oprogramowanie. Ponieważ procentowy udział kosztów oprogramowania nadal wzrasta, kraje przodujące technologicznie podejmują wiele działań mających na celu jego zmniejszenie. Jednym z najbardziej w ostatnim okresie znanych i interesujących przedsięwzięć tego rodzaju, zapoczątkowanym przez amerykański Departament Obrony, jest próba stworzenia uniwersalnego języka programowania ADA.

W kilku najbliższych numerach przedstawimy Czytelnikom przebieg procesu powstawania języka, opiszemy jego budowę, podamy przykłady programów oraz skomentujemy niektóre własności.

Badania przeprowadzone w latach 1973—1974 przez amerykański Departament Obrony wykazały, że koszty oprogramowania systemów komputerowych rosną zarówno sumarycznie jak i procentowo w stosunku do innych rodzajów nakładów. Odnosi się to szczególnie do tzw. wbudowanych systemów komputerowych (ang. *embedded computer systems*), stanowiących integralną część większych urządzeń wojskowych, instalacji przemysłowych lub innych obiektów technicznych. Według szacunkowych danych [1] oprogramowanie tych systemów pochłonęło w 1973 roku 56% ogólnych nakładów, w porównaniu do 19% przeznaczonych na oprogramowanie systemów przetwarzania danych i tylko 5% — na komputery wykorzystywane w badaniach naukowych. Należy dodać, że koszty związane z pielęgnacją oprogramowania mają tu znacznie większy udział niż koszty jego projektowania.

Jednym z wyników wspomnianej analizy było stwierdzenie, że przyczyną większości kosztów jest duża niejednorodność języków programowania. Tylko w systemach informatycznych na potrzeby wojsk lądowych doliczono się stosowania 450 różnych języków i wzajemnie niezgodnych ich odmian.

Oprócz niejednorodności inną przyczyną omawianego stanu była niedoskonałość oprogramowania. Systemy informatyczne czasu rzeczywistego [1]) wymagają oprogramowania o dużej niezawodności, a więc takiego, które będzie działać nawet w przypadku błędów sprzętu, danych wejściowych, operatora lub wreszcie — swoich własnych.

Duży wpływ na ten stan miało też nieprzystosowanie istniejącego oprogramowania do podstawowych wymagań systemów czasu rzeczywistego (krótkie czasy obsługi zgłoszeń zewnętrznych, łatwa komunikacja operatora z urządzeniem itp.).

Ponieważ język programowania stanowi centralny element oprogramowania komputera, jest zrozumiałe, że przez jego znormalizowanie można osiągnąć znaczne efekty ekonomiczne.

WIELOSTOPNIOWY PROCES DEFINIOWANIA JĘZYKA

Pierwszym krokiem w tym kierunku było sformułowanie programu zastosowania wspólnego języka programowania komputerów w siłach zbrojnych USA (styczeń 1975). W tym celu utworzono grupę badawczą HOLWG (skrót od High Order Language Working Group), której początkowe zadanie polegało na koordynowaniu działalności w zakresie ustalenia minimalnej liczby języków wysokiego poziomu do stosowania w projektowaniu i użytkowaniu systemów komputerowych w siłach zbrojnych. Jest charakterystyczne, że grupa ta nie była tzw. „komitetem", mającym za zadanie opracowanie nowego języka, jak to bywało w przeszłości (np. w przypadku ALGOL-u), lecz jedynie koordynatorem działań. Wspomniana grupa dość szybko stwierdziła, że żaden z istniejących języków nie jest odpowiedni do realizacji wszystkich rodzajów zastosowań, co spowodowało podjęcie decyzji opracowania całkowicie nowego. Wkrótce też stwierdzono, że wyrażenie w tym języku już istniejących programów będzie niepraktyczne i postanowiono, że będzie on użyty wyłącznie do opracowania nowych systemów informatycznych.

Pomijając mało istotną w cyklu powstawania języka próbę ograniczenia używanych języków do 7 najważniejszych, pierwszym ważnym efektem pracy grupy HOLWG było sformułowanie w raporcie STRAWMAN (kwiecień 1975) wymagań, mających charakter ograniczeń warunkujących późniejsze przyjęcie języka, lecz nie określających jego poszczególnych cech. Nie była to więc specyfikacja języka, lecz próba ścisłego określenia jego charakterystyki w taki sposób, aby można ją było poddać ocenie.

Było to zadanie o tyle trudne, że w rzeczywistości niełatwo oddzielić wymagania ogólne, takie jak: wydajność, czytelność, prostota, od cech języka, które tym wymaganiom odpowiadają. W rezultacie raport STRAWMAN miał jedynie charakter wstępny.

Krytyczna ocena przedstawionych wymagań, dokonana głównie przez jednostki sił zbrojnych USA, doprowadziła do powstania pełniejszego, lecz nadal uważanego za wstępne, opracowania WOODENMAN (sierpień 1975). Dokument ten, zawierający bardziej ścisłe, lecz jeszcze nie skwantyfikowane sformułowania, rozpowszechniono już w

[1]) Nie wydaje się potrzebne na użytek kilku artykułów tworzenie polskiego odpowiednika „embedded computer systems". Dlatego stosowane będzie nieco szersze określenie „systemy czasu rzeczywistego".

Dr inż. Janusz ZALEWSKI ukończył w 1973 r. studia na Wydziale Elektroniki Politechniki Warszawskiej, a pracę doktorską obronił na Wydziale Elektrycznym tejże uczelni w roku 1979. Pracuje na stanowisku adiunkta w Instytucie Badań Jądrowych, gdzie zajmuje się zagadnieniami automatyzacji eksperymentów. Jest członkiem kolegium redakcyjnego INFORMATYKI.

znacznie szerszym kręgu instytucji, a więc poza wojskiem — również w środowiskach uniwersyteckich, przemysłowych oraz organów administracji państwowej (także w Europie).

Po zebraniu uwag krytycznych powstał zbiór bardziej ścisłych, a jednocześnie zawężonych tematycznie wymagań TINMAN (styczeń 1976), ze szczególnym uzasadnieniem każdego z nich. Efektem końcowym tych prac było sympozjum [2], na którym przedyskutowano wyniki oraz przyjęto założenia i określono dalsze kierunki prac.

W kolejnych dokumentach IRONMAN (styczeń i lipiec 1977) określono w ścisłej formie wymagania zawarte w poprzednim raporcie. Tym razem był to dokument zawierający w zasadzie specyfikację języka, od której należało zacząć jego projektowanie.

Już w roku 1976, tzn. po opublikowaniu dokumentu TINMAN, 16 zaproszonych organizacji dokonało oceny dotychczas używanych 23 języków, pod względem spełnienia sformułowanych wymagań. Specjaliści ci stwierdzili, że żaden z istniejących języków nie spełnia tych wymagań w zadowalającym stopniu. Jedynie 3 języki (oraz ich pochodne) tj. ALGOL-68, PASCAL i PL/I uznano za zgodne z wymaganiami. Głównym wnioskiem tych prac było stwierdzenie, że w oparciu o wymienione języki należy stworzyć nowy, spełniający wszystkie pożądane wymagania.

W odpowiedzi na propozycję opracowania takiego języka (kwiecień 1977) powstało kilkanaście projektów, spośród których wybrano 4 najlepsze, pochodzące z 3 firm amerykańskich INTERMETRICS, SOFTECH, SRI-INTERNATIONAL oraz firmy francusko-amerykańskiej (CII-HONEYWELL BULL). W sierpniu 1977 r. rozpoczęły one formalne współzawodnictwo o najlepszą definicję języka.

Cztery dostarczone projekty (luty 1978) oznaczono, dla zachowania anonimowości, nazwami GREEN, RED, YELLOW i BLUE. Następnie poddano je ocenie międzynarodowych grup ekspertów. Wyniki tej oceny oraz zalecenia przedstawiono grupie HOLWG, która wybrała do dalszej rywalizacji (tzn. udoskonalenia definicji) tylko dwie propozycje: GREEN (CII-HONEYWELL BULL), oraz RED (INTERMETRICS).

Drugą fazę tworzenia języka [3], mającą na celu sformułowanie jego dokładnej definicji, rozpoczęto w kwietniu 1978 r. i zakończono — po dalszym uściśleniu wymagań (dokument STEELMAN, czerwiec 1978) — dostarczeniem i opublikowaniem (15 marzec 1979) dwóch pełnych dokumentów zawierających definicje języków GREEN i RED. Proces ich oceny został zakończony ogłoszeniem decyzji (2 maj 1979) o przyjęciu propozycji GREEN firmy CII-HONEYWELL BULL, którą opracowała grupa programistów kierowana przez J. Ichbiaha. Językowi temu nadano nazwę ADA od imienia pierwszej w historii programistki, córki lorda Byrona, wspomagającej Charlesa Babbage'a w pracy nad jego maszyną różnicową [4].

Po wyborze projektu GREEN rozpoczęto proces sprawdzania i oceny języka (ng. *test and evaluation phase*), mający na celu zbadanie jego przydatności do rzeczywistych zadań oraz dokonanie na tej podstawie niezbędnych poprawek. W tym celu różne grupy programistów przystąpiły do pisania w tym języku przykładowych programów a następnie ich uruchamiania za pomocą próbnego translatora udostępnionego w sieci komputerowej ARPA.

Rezultaty tych działań przedstawiono w 82 pracach na specjalnie zorganizowanym sympozjum [5]. Głównym wnioskiem z obrad było stwierdzenie konieczności dokonania dalszych poprawek. Okazało się również, że niedostateczne przeszkolenie programistów oraz interpretacyjny charakter translatora wpłynęły ujemnie na możliwości uzyskania dokładniejszej oceny języka.

OGÓLNA CHARAKTERYSTYKA JĘZYKA

Jednym z głównych celów realizatorów projektu była — jak stwierdził Jean Ichbiah — realizacja koncepcji składowych oprogramowania [6] (ang. *software components*). Zgodnie z tą koncepcją oprogramowanie powinno składać się z modułów, które można oddzielnie kompilować i łączyć w dowolny sposób, tak jednak, aby ich połączenia spełniały określone warunki zgodności. Ponieważ zasady zgodności, określane w każdym przypadku przez kompilator, byłyby analogiczne do reguł magistrali łączącej elementy systemu, całą koncepcję można porównać ze znaną ideą modularnych systemów cyfrowych, w których poszczególne bloki, dopasowane logicznie do magistrali, można konfigurować w dowolne zestawy użytkowe. Przeniesienie tej idei do dziedziny programowania oznacza, że moduły programowe można wybierać z biblioteki, a rolę magistrali spełnia kompilator.

Była to podstawowa reguła, jaką kierowali się autorzy przy określaniu struktury języka. Jego własności zostaną bardziej szczegółowo omówione w następnym numerze INFORMATYKI, obecnie zostanie podana jedynie ogólna ich charakterystyka.

Język ADA ma wiele cech PASCALA (w mniejszym stopniu — ALGOLU i PL/I), jednak jest od niego bardziej rozbudowany. Podobnie jak PASCAL, ADA ma duże możliwości, jeśli chodzi o typy danych, dopuszczając dodatkowo wprowadzanie nowych typów danych przez użytkownika.

Znaczenie typizacji danych łatwo zrozumieć, jeśli zachodzi potrzeba rozwiązania zagadnienia np. z zakresu regulacji ruchu ulicznego przy użyciu świateł. Jest oczywiste, że znacznie łatwiej napisać program posługując się abstrakcyjnymi danymi stanowiącymi nazwy kolorów świateł, niż ich reprezentacją numeryczną. Gdy istnieje możliwość tworzenia abstrakcyjnych typów danych, programista w ogóle nie troszczy się o ich przedstawienie w maszynie, pozostawiając zadanie to komputerowi. Oddzielenie tych dwóch spraw stwarza cenną możliwość tworzenia bibliotek programów we wszystkich dziedzinach, a nie, jak dotychczas, tylko w dziedzinie analizy numerycznej.

Inne podstawowe elementy konstrukcji ADY, to pakiety (ang. *packages*) i zadania (ang. *tasks*), stanowiące moduły włączane do biblioteki. Jedne z ważniejszych własności tego języka, odróżniające go od dotychczasowych, to spotkania (ang. *rendenzvouz*) i wypadki (ang. *exceptions*), stanowiące instrumenty synchronizacji i reakcji na sytuacje nieoczekiwane, a więc niezbędne przy programowaniu systemów czasu rzeczywistego.

ZAGADNIENIA OPROGRAMOWANIA PODSTAWOWEGO

Od chwili podjęcia prac zdawano sobie sprawę z konieczności jednoczesnego wytworzenia odpowiednich narzędzi programowych, a także — powstania organizacji, czuwającej nad kształtem języka oraz jakością kompilatorów. Niektóre z tych wymagań sformułowano już w raporcie STEELMAN, podejmując równocześnie wysiłki na rzecz skoordynowania działalności. Prace przygotowawcze, rozpoczęte w roku 1977 zaowocowały dostarczeniem wyników badań (wiosna 1978) i powstaniem dokumentu PEBBLEMAN (lipiec 1978), który — podobnie jak uprzednio STRAWMAN w odniesieniu do języka — miał zadanie zapoczątkować dyskusję nad własnościami oprogramowania podstawowego (środowiska programowego, ang. *software environment*).

Opublikowanie tego dokumentu zostało poprzedzone konferencją [7], na której przedyskutowano wstępną wersję raportu. Do najważniejszych wniosków tego sympozjum należy stwierdzenie, że podstawowe trudności w procesie oprogramowania związane są z fazami projektowania i pielęgnacji programów, przy czym koszty pielęgnacji sięgają niekiedy do 95% nakładów całkowitych. Potwierdza to bezwzględną konieczność normalizacji środowiska programowego dla ADY. Krytycznie oceniono sposób realizacji tak ważnego zadania, jak weryfikacja i testosowanie kompilatorów. Dotąd nie wypracowano wspólnej metodyki, choć prace są prowadzone w wielu ośrodkach badawczych. Jednym ze środków zaradczych byłoby utworzenie odpowiedniej organizacji użytkowników.

Druga wersja dokumentu PEBBLEMAN (styczeń 1979) powstała na podstawie uwag i opinii zgłoszonych do pierwszej publikacji. Mówiąc w największym skrócie, efektem prac było wyróżnienie trzech głównych składników oprogramowania podstawowego (czyli tzw. środowiska programowego), niezależnych od szczegółów realizacji kompilatora, a mianowicie:

- sprzęgu użytkowego (ang. *user interface*)
- bazy danych (ang. *data base*)
- narzędzi programowych (ang. *tools set*).

Wymienione składniki są rozumiane bardzo szeroko [8]. Przykładowo, sprzęg użytkowy mogą tworzyć tylko najprostsze karty sterujące (ang. *job control cards*) służące do konwencjonalnego wywołania kompilatora, lub może to być skomplikowany program interakcyjny, koordynujący dostęp do bazy danych i mający wiele innych możliwości. Podobnie, bazę danych mogą tworzyć tylko podręczniki użytkowania, albo też może ona być złożonym systemem bibliotek. Natomiast narzędziem programowania może być nie tylko ołówek z kartką papieru, lecz również zautomatyzowany system z możliwością testowania programów i generowaniem — zależnie od problemu — danych do tych testów.

Dyskusja nad raportem PEBBLEMAN (i wcześniejszym — SANDMAN) ujawniła, że utworzenie właściwego środowiska programowego wymaga znacznie większych nakładów niż to się powszechnie sądzi. Ponadto stwierdzono, że wymagania formułowane dotąd raczej w teorii, należy odnieść do rzeczywistych instalacji. W efekcie powstały dwa dalsze opracowania [9, 10], w których zawarto wiele cennych spostrzeżeń dotyczących m.in. roli warunków społecznych i organizacyjnych, a więc tzw. metaśrodowiska, w jakim projektuje się oprogramowanie oraz z niego korzysta. Ze względu na ogromną różnorodność i nierówny poziom istniejącego oprogramowania, oprogramowanie nowe i jednolite — najbezpieczniej byłoby wprowadzać stopniowo. Należy przy tym pamiętać, że środowisko, w którym trzeba będzie pielęgnować to oprogramowanie, może znacznie się różnić od środowiska, w którym je opracowano. Jednym z ciekawszych spostrzeżeń było stwierdzenie użyteczności asemblerów jako sprzęgów wewnętrznych, łączących różne elementy środowiska programowego.

W wyniku tych dyskusji przystąpiono do równoległych prac nad normalizacją środowiska programowego języka (tj. oprogramowania podstawowego nie obejmującego kompilatora), a więc: bibliotek, sprzęgów użytkowych, narzędzi automatycznego projektowania, uruchamiania i pielęgnacji programów itp., a także — nad realizacją i uwiarygodnieniem kompilatorów (ang. compiler validation), uwiarygodnienie należy rozumieć tu jako potwierdzenie zgodności z normą.

POWSTAWANIE OPROGRAMOWANIA

Obecnie trwają intensywne prace nad kompilatorami języka oraz powstałym oprogramowaniem podstawowym. Dotychczas opracowano jedynie wersje eksperymentalne kompilatorów, nie nadające się jeszcze do rozpowszechniania. Jednakże wiele instytucji ma już gotowe programy w języku ADA i czeka tylko na powstanie i udostępnienie kompilatorów.

Oficjalne zlecenie na napisanie kompilatora dla komputerów. VAX-aa/780, PDP-11/70 i GYK-12 (model wojskowy) otrzymała firma SOFTECH Inc. Przewiduje się, że będzie on gotowy w całości w roku 1982 i wtedy — poddany uwiarygodnieniu. Kontrakt zakłada, że kompilator będzie współdziałał z istniejącym oprogramowaniem podstawowym, tzn. z systemami operacyjnymi VMS, UNIX itp.

Firmy SOFTECH i HONEYWELL rozpoczęły także prace nad tzw. systemem ALS (ADA Language System) zawierającym kompilator, optymalizator i generator kodu oraz program ładujący. Prace te stanowią część przedsięwzięcia zwanego MAPSE (Minimal ADA Programming Support Environment) mającego na celu stworzenie pełnego oprogramowania podstawowego, zgodnego z wymaganiami zawartymi w dokumencie STONEMAN. Jesienią 1982 roku przewidziana jest weryfikacja i uprawomocnienie systemu ALS (Teledyne Brown Engineering), a wiosną 1983 roku — jego wprowadzenie do użytkowania.

Inny projekt przewiduje opracowanie drogą współzawodnictwa, pełnego przenośnego oprogramowania podstawowego w trzech etapach.

W pierwszym etapie zawarto wstępne kontrakty na opracowanie kompilatora dla komputerów IBM 370 (pod nadzorem systemu operacyjnego VM) z firmami TEXAS INSTRUMENTS, INTERMETRICS, Inc. i COMPUTER SCIENCES Corp., które wygrały ogłoszony konkurs (brały w nim udział również firmy SOFTECH, TRW i UNIVAC). Po dokonaniu oceny (marzec 1981) przewidziano realizację najlepszego kompilatora (lipiec 1981) i zademonstrowanie jego przenośności (na komputer INTERDATA 8/32, pod nadzorem sytemu operacyjnego OS32).

W trzecim etapie przewidziane jest opracowanie oprogramowania podstawowego (tzw. MAPSE) wg zaleceń zawartych w dokumencie STONEMAN. Wymagane narzędzia programowe obejmują: kompilator, program redagujący i łączący, program uruchamiania oraz generator systemu operacyjnego. W podobny sposób określono pożądane własności bazy danych i sprzężenia użytkowego.

Jest prawdopodobne, jednak, że pierwsze kompilatory powstaną dla mikroprocesorów takich firm, jak: MOTOROLA, ZILOG i INTEL. Firma INTEL opracowała specjalny układ mikroprocesora 32-bitowego iAPX-432, którego architektura odpowiada strukturze języka [11]. Niektóre konstrukcje języka mają swoje odpowiedniki bezpośrednio w elementach architektury mikroprocesora. W pozostałych przypadkach, dzięki elastyczności języka istnieje możliwość ścisłego dopasowania struktur programowych do układowych, co pozwala wyeliminować używanie języka symbolicznego. Tak więc mikroprocesor iAPX-432 będzie oprogramowany całkowicie w języku ADA.

Powstają również kompilatory dla wielu innych komputerów i, co ciekawe, co najmniej tyle samo translatorów ADY na inne języki, jak FORTRAN lub PASCAL. Jednak, niektóre firmy w USA zachowują powściągliwość nie włączając się narazie do prac i co najwyżej prowadząc szkolenie programistów. W Europie, kompilatory pełnej wersji języka ADA powstają m.in. w Karlsruhe, Monachium, Zurichu, i Budapeszcie.

STAN AKTUALNY

Firma CII-HONEYWELL BULL dostarczyła kontrahentowi formalną definicję języka ADA 30 września 1980 r., po wprowadzeniu uzupełnień do poprzedniej wersji. W ciągu 3 lat nad projektem języka pracowało jednocześnie 4—5 osób, a łącznie — 11. Poprawki, które ostatnio wprowadzono, dotyczyły w różnym stopniu wszystkich elementów języka: od deklaracji, przez podprogramy i pakiety, do zadań i wypadków. Jednak dokument jest krytykowany z bardziej ogólnego powodu, a mianowicie, za niezbyt precyzyjny sposób opisu, dopuszczający niejednoznaczną interpretację.

Do przyjęcia ADY jako międzynarodowego standardu mogą upłynąć jeszcze 2—3 lata. Najpierw bowiem amerykański Departament Obrony powinien zebrać opinie od potencjalnych użytkowników i dostarczyć je wraz z definicją języka organizacjom normalizacyjnym, tj. ANSI (American National Standards Institute) oraz ISO (International Organization for Standardization). Następnie, wymienione organizacje muszą powtórzyć długotrwały proces zbierania opinii — przed podjęciem ostatecznej decyzji w drodze głosowania.

Zainteresowanie językiem jest jednak tak duże, że z pewnością stanie się on standardowym na długo przed oficjalnym zatwierdzeniem normy. Już obecnie istnieją organizacje zrzeszające użytkowników w USA, w Europie, a nawet w poszczególnych krajach. Organizacja ADA Europe powstała w lutym 1980 r., a przewodniczy jej jeden z twórców języka B. W. Wichmann (Wielka Brytania). Spotkania odbywają się kilka razy do roku, a ich głównym celem jest koordynacja prac nad definicją języka i realizacją oprogramowania. Działalność poszczególnych sekcji tematycznych dotyczy m.in. dokonywania zmian w definicji, oprogramowania podstawowego, realizacji i uprawomocnienia kompilatorów, przenośności oprogramowania, rozpowszechniania informacji itp.

Ostatnio zaproponowano m.in. powołanie europejskiego centrum uprawomocnienia kompilatorów, afiliowanego przy jednej z zaangażowanych w projekt instytucji. Jest to o tyle istotne, że instytucja finansująca większość prac, tj. amerykański Departament Obrony, zamierza zastrzec używanie nazwy ADA jedynie dla tych kompilatorów, które zostaną poddane uprawomocnieniu. Proces uprawomocnienia ma trwać ok. 2 miesiące i kosztować ok. 10 000 dolarów.

LITERATURA

[1] Fisher D. A., DOD's Common Programming Language Effort, COMPUTER, vol. 11, No 3, 1978, p. 24

[2] Williams J. H., Fisher D. A., eds., Design and Implementation of Programming Languages, Proc. DOD Workshop, October 1976, Lecture Notes in Computer Science, vol. 54, Springer-Verlag, Berlin, 1977

[3] Wichmann B. A., The Development of ADA, The DOD Language, pp. 52-63, GI-9. Jahres tagung, Bonn, K. H. Boehling, P. P. Spies, eds., Springer Verlag, Berlin, 1979

[4] Bytniewski M., Ada Byron — romantyczna prekursorka programowania maszyn cyfrowych. INFORMATYKA, nr 9, 1977, str. 36

[5] ADA Test and Evaluation Worksnop, Boston, MA, USA, 23—26 October, 1979

[6] Zalewski S., Uniwersalny język komputerowy, Przegląd Techniczny nr 16, 1981, str. 35

[7] Standish T. A., ed., Proc. Irvine Workshop Alternatives for the Environment, Certification and Control of the DOD Common High Order Language, Irvine, CA, USA, June 1978

[8] Elzer P. F., The Evolution of the Requirements for the Software Environment for ADA, pp. 397—401, Real-Time Data Handling and Process Control, Proc. 1st European Symposium, Berlin West 23—25 October, 1979, H. Meyer, ed., North-Holland, Amsterdam, 1980

[9] Fisher D. A., Standish T. A., Initial Thoughts on the PEBBLEMAN Process. Raport IDA P-1392, Institute of Defense Analysis, Arlington, VA, USA, June 1979

[10] Elzer P., Some Observations Concerning Existing Software Environments, pp. 227—260, Pt. 1, Minutes 7th Annual Meeting International Purdue Workshop Industrial Computer Systems, West Lafayette, IN, USA, 8—11 October 1979

[11] Rattner J., Lattin W. W., ADA Defermines Architecture of 32-bit Microprocessor, Electronics, Vol. 54, No. 3, p. 119 12 February 1981

JERZY CHMURZYŃSKI
KRZYSZTOF LIDERMAN

Wojskowa Akademia Techniczna
Warszawa

Pomiary wydajności systemu komputerowego monitorem sprzętowym KL-80

Na obecnym etapie rozwoju techniki komputerowej powstała konieczność stosowania specjalnych metod oceny systemów komputerowych. Oceny oparte na pomiarach należą do kategorii metod bezpośrednich i mają duże walory użytkowe. Zasadniczym powodem dokonywania oceny lub pomiaru jest potrzeba określenia najlepszego sposobu projektowania, instalowania lub ulepszania działania systemu komputerowego.

Powyższe potrzeby stały się bodźcem do rozwoju nowego działu techniki komputerowej — miernictwa komputerowego [1, 3, 4, 8].

Miernictwo komputerowe wypracowało już własne metody pozyskiwania danych o zachowaniu się systemów komputerowych, interpretowania i uogólniania tych danych oraz wykorzystywania ich podczas projektowania i eksploatacji systemu.

W niniejszym artykule przedstawione zostały metody i wyniki pomiarów przeprowadzonych monitorem sprzętowym KL-80, zbudowanym w Wojskowej Akademii Technicznej.

MONITORY POMIAROWE

Podstawowymi środkami śledzenia zachowania się systemów komputerowych są monitory. Monitorowanie jest metodą zbierania danych o pracy systemów komputerowych. W zależności od tego:

- jakie dane trzeba uzyskać
- gdzie można je znaleźć
- jak je uzyskać i zarejestrować

do pozyskiwania danych stosowane są monitory programowe i sprzętowe [3].

Monitory programowe

Monitor programowy jest zbiorem progamów do badania pracujących systemów komputerowych. W zbiorze tym można wyróżnić dwie grupy programów:

- programy pomiarowe, zbierające dane do analizy
- programy analizująco-redukujące, opracowujące zebrane dane pomiarowe (programy takie mogą być wykonywane po zakończeniu pomiarów).

Dr inż. Jerzy CHMURZYŃSKI ukończył studia na Wydziale Cybernetyki WAT. od 1972 r. pracuje w WAT. Zajmuje się projektowaniem systemów informatycznych.

Mgr inż. Krzysztof LIDERMAN ukończył w 1979 r. studia na Wydziale Cybernetyki WAT. Obecnie odbywa praktykę w WAT.

Programy pomiarowe zbierają dane o pracy systemu, badając i rejestrując zawartości określonych rejestrów i/lub odpowiednich obszarów PAO, z częstotliwością ustaloną przez użytkownika lub system komputerowy.

Można wyróżnić następujące dwie metody pomiaru:

- metoda przejmowania sterowania programowego, polegająca na tym, że w węzłowych punktach systemu operacyjnego i/lub programach użytkowych wprowadza się dodatkowe rozkazy przekazujące sterowanie odpowiedniemu programowemu elementowi wykonawczemu
- metoda próbkowania, polegająca na tym, że wykorzystuje się specjalne podprogramy generujące przerwania w odpowiednich odstępach czasu, umożliwiające elementowi wykonawczemu zapis stanu systemu w momencie przerwania.

Pozostałe programy wchodzące w skład monitora programowego, tzw. programy analizująco-redukujące opracowują dane uzyskane z programów pomiarowych do postaci dogodnej dla użytkownika.

Więcej wiadomości na temat monitorów programowych można znaleźć w publikacjach [3, 7, 8].

Monitory sprzętowe

Podstawowe funkcje sprzętowego monitora pomiarowego przedstawia rys. 1. Monitor jest jednostką autonomiczną, łączoną z badanym systemem komputerowym zespołem sond pomiarowych. Sondy rejestrują zmiany stanów w wybranych miejscach układów elektronicznych. Zebrane sygnały elementarne mogą być wzmacniane i kształtowane do postaci dogodnej do dalszego użycia.

Rys. 1. Podstawowe funkcje monitora sprzętowego

Zebrane sygnały elementarne wprowadzane są na sieć logiczną złożoną z funktorów logicznych. Z pomocą sieci logicznej dokonuje się kombinacji i selekcji sygnałów elementarnych w celu sformułowania, istotnych dla danego pomiaru, sygnałów zdarzeń pomiarowych. Przykładowo, sygnały elementarne ze wskaźnika EXM, linia $\overline{R}$ i linia $\overline{B}$ złącza standardowego komputera ODRA 1305 podane na sieć logiczną tworzą sygnał zdarzenia pomiarowego „równoczesna praca użytkowa procesora i transmisja danych w kanale".

Struktura sieci logicznej związana jest z rodzajem przeprowadzanego pomiaru. Zmianę struktury sieci dokonuje się dwoma sposobami. Pierwszy z nich pozwala ustalić strukturę sieci za pomocą tablicy krosowej. Przed każdym więc pomiarem należy ustalić odpowiednią strukturę sieci. Drugi sposób zakłada istnienie zbioru wymiennych pakietów, w których „zaszyto" na stałe strukturę sieci logicznej. Poprawnie przygotowane pakiety pozwalają na szybkie przystosowanie monitora sprzętowego do przeprowadzania odmiennego typu pomiaru i pozwalają unikać błędów podczas zmian struktury sieci logicznej.

Zliczanie liczby zdarzeń pomiarowych i czasu ich trwania dokonywane jest w zespole liczników. Do pomiaru czasu trwania zdarzeń pomiarowych konieczny jest układ zegarowy, którego impulsy są sumowane w wybranym liczniku wtedy, kiedy tylko pojawi się określone zdarzenie. Mówiąc inaczej, wejściem na iloczyn logiczny sygnał zdarzenia pomiarowego warunkuje skierowanie impulsów z układu zegarowego do licznika.

Zebrane przez monitor sprzętowy dane pomiarowe często są nieprzydatne do bezpośredniego wykorzystania. Stąd też wyprowadzanie danych pomiarowych odbywa się na taśmę magnetyczną lub pamięć dyskową, z przeznaczeniem do dalszego przetworzenia. Formy przedstawiania wyników monitorowania mogą być następujące:

- histogramy aktywności składowych systemu
- tabele zbiorcze pomiarów
- tabele statystyczne pomiarów itd.

Monitor sprzętowy jest narzędziem umożliwiającym pomiar szeregu współczynników charakteryzujących pracę systemu komputerowego. Współczynniki te mogą dotyczyć np.:

- wykorzystania procesora (aktywny/nieaktywny, dekodowanie operacji itp.)
- wykorzystania pamięci taśmowych (aktywna-nieaktywna, liczba operacji „pisz"/„czytaj" itp.)
- wykorzystania pamięci dyskowych (częstość dostępów, czas szukania, liczba przesunięć głowic itp.)
- wykorzystania terminali
- zrównoleglenia pracy kanałów
- zrównoleglenia pracy procesora z innymi elementami składowymi systemu
- rodzaju i typu przerwań we/wy.

W nowoczesnych rozwiązaniach monitor sprzętowy współpracuje z minikomputerem, który steruje zbieraniem danych i przesyła do swych pamięci masowych wyniki pomiarów. Pamięć operacyjna minikomputerów wykorzystywana jest jako pamięć buforowa, pośrednicząca w przekazywaniu danych pomiarowych.

Porównanie monitorów sprzętowych i programowych

Nie należy uważać za konkurencyjne metod pozyskiwania danych za pomocą monitorów programowych i sprzętowych, gdyż mają one swoje wady i zalety, a w wielu przypadkach uzupełniają się wzajemnie. Poniżej przedstawiono porównanie podstawowych własności obu metod pomiarowych.

- **Koszty.** Koszty bezpośrednie monitorowania sprzętowego są zwykle wyższe od kosztów monitorowania programowego. Wyższe są również związane z nim koszty dodatkowe, takie jak: szkolenie, obsługa i przygotowanie eksperymentu.

- **Obciążenie badanego systemu.** Monitory sprzętowe nie wprowadzają praktycznie żadnych zakłóceń ani obciążeń dodatkowych do badanego systemu. Monitory programowe wymagają natomiast, podczas zbierania danych, dodatkowego obszaru pamięci oraz czasu procesora. Obciążenie to zmienia się w zależności od częstości dokonywania pomiarów. Doświadczenia wykazują, że obciążenie procesora monitorem programowym może wynosić 5—10% obciążenia systemu komputerowego, co należy mieć na uwadze przy interpretacji wyników.

- **Zależność od systemu komputerowego.** Monitory sprzętowe są całkowicie niezależne od sprzętu i oprogramowania badanego systemu komputerowego. Monitory programowe są opracowywane specjalnie dla określonego oprogramowania i przeznaczenia systemu.

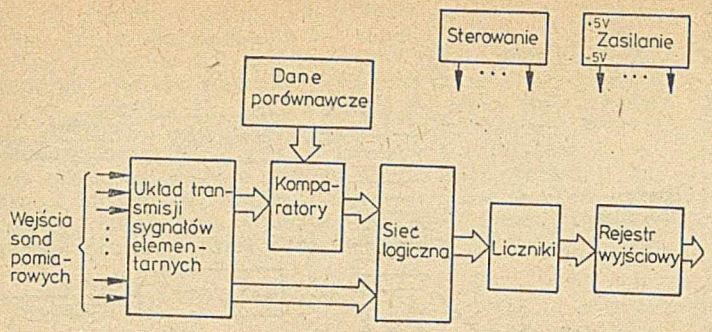
- **Dokładność pomiaru.** W monitorze sprzętowym dokładność zależy od budowy i szybkości działania np. rejestrów buforowych czy układów zapisujących dane w pamięci. Dokładność pomiaru monitorów programowych jest zależna od funkcji i możliwości badanego systemu komputerowego, np. dokładności odmierzania czasu systemowego, zasady obsługi przerwań, możliwości włączenia dodatkowych procedur do oprogramowania systemowego.

- **Dostęp do danych.** Niektóre dane, np. dane o długościach kolejek w systemie, czasie oczekiwania zgłoszeń w kolejkach itp. są trudno dostępne dla monitorów sprzętowych. Dostęp ten może być ułatwiony przez wprowadzenie do systemu pomocniczych procedur wspomagających pracę monitora sprzętowego.

Z innych, godnych uwagi własności monitorów sprzętowych należy wymienić uniwersalność. Monitory te mogą być stosowane również poza systemami komputerowymi wszędzie tam, gdzie występują sygnały elektryczne mierzalne przez monitor.

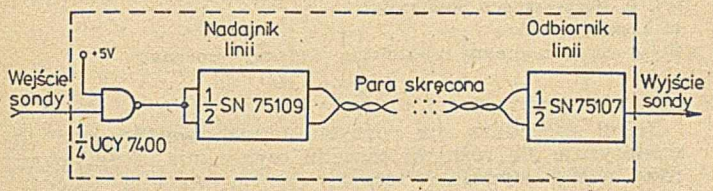
POMIAROWY MONITOR SPRZĘTOWY KL-80

Monitor sprzętowy KL-80 jest zbudowany z układów TTL małej i średniej skali integracji, zmontowanych na 16 pakietach [2, 6]. Monitor przewidziany jest do współpracy z minikomputerem, w którego pamięciach masowych przechowywane są dane pomiarowe. Schemat blokowy monitora KL-80 przedstawiono na rys. 2.



Rys. 2. Schemat blokowy monitora KL-80

Sygnały elementarne podawane są na wejścia sond pomiarowych. Sonda pomiarowa (rys. 3) stanowi obciążenie prądowe badanego systemu komputerowego, równe jednostce obciążenia układów scalonych TTL. Monitor KL-80 ma możliwość zbierania do 72 sygnałów elementarnych.



Rys. 3. Sonda pomiarowa

Częstość sygnałów elementarnych przekazywana jest do układu komparatorów. Komparatory pozwalają porównywać zawartości wybranych rejestrów badanego systemu z wartością danych porównawczych, wprowadzanych za pomocą klawiatury pulpitu technicznego monitora. Monitor wyposażony jest w 6 komparatorów 8-bitowych, które mogą być łączone w komparatory 16- i 24-bitowe. Sygnały na wyjściu komparatorów, reprezentujące wyniki porównania, są wprowadzane do sieci logicznej.

Sieć logiczna monitora KL-80 znajduje się na pakiecie wymiennym. Struktura sieci logicznej dla przeprowadzonych eksperymentów pomiarowych zostanie opisana w końcowej części artykułu.

Pomiar czasu trwania i zliczanie zdarzeń pomiarowych dokonywane są w układzie liczników. Monitor wyposażony jest w 2 liczniki 24-bitowe, 8 liczników 16-bitowych i 10 liczników 8-bitowych. Zawartości liczników wyprowadzane są z monitora poprzez układ rejestru wyjściowego. Kolejne bajty danych z liczników przekazywane są według kolejności ustalonej przez przełącznicę sterującą informacją wyjściową. Dane pomiarowe przekazywane są do minikomputera MERA 306 na liniach *BE0* — *BE7* złącza standardowego. Również na tych liniach przesyłana jest zawartość rejestru stanu monitora.

Warunkiem pojawienia się na liniach BE_i bajtu danych lub zawartości rejestru stanu monitora jest pojawienie się odpowiednich sygnałów sterujących z układu sterowania. Układ sterowania generuje wszystkie inne sygnały niezbędne do działania monitora. Sygnałem wykorzystywanym do synchronizowania pracy monitora jest sygnał *T* zegara, przekazywany z komputera ODRA 1305 poprzez jedną z sond. Wydzieloną częścią układu sterowania jest 24-bitowy licznik czasu. Jego zadaniem jest odmierzanie cyklu pomiarowego monitora. Cykl ten jest regulowany z pulpitu technicznego w granicach od 60 ms do 20 s.

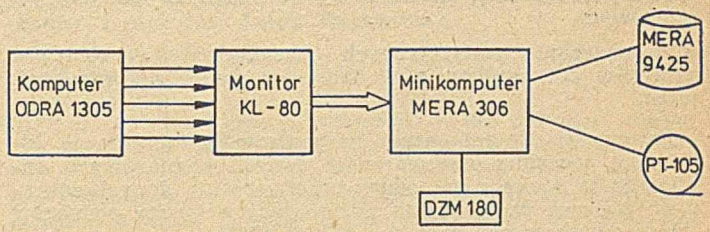
Na wejście licznika czasu podawane są impulsy zegarowe. W momencie wyzerowania licznika generowany jest sygnał końca cyklu pomiarowego, który inicjuje operację przekazywania zawartości liczników poprzez rejestr wyjściowy do minikomputera.

W każdym cyklu pomiarowym monitor przesyła 40 bajtów danych do minikomputera. Struktura przekazywanych danych przedstawiona została na rys. 4. Podczas przekazywania danych pomiarowych zablokowany jest dostęp do liczników i monitor nie reaguje na sygnały zdarzeń pomiarowych. Czas ten wynosi ok. 4 ms. Po przesłaniu danych pomiarowych zerowane są wszystkie liczniki i rozpoczyna się kolejny cykl pomiarowy.

Rys. 4. Struktura danych przekazywanych w każdym cyklu pomiarowym z monitora KL-80 do minikomputera MERA 306

ZESTAW POMIAROWY

W trakcie prowadzenia pomiarów monitor sprzętowy KL-80 współpracuje z minikomputerem MERA 306. Na rys. 5 przedstawiony został zestaw pomiarowy w całości.



Rys. 5. Zestaw pomiarowy

Drukarka DZM 180 służy do wprowadzania dyrektyw inicjujących i kończących pomiary. Wraz z dyrektywą inicjującą wprowadzane są takie parametry pomiaru jak: data, typ i numer kolejny pomiaru, długość cyklu pomiarowego. Parametry te wprowadzane są do etykiety początku podzbioru danych pomiarowych, przechowywanego w pamięci dyskowej MERA 9425.

Dane pomiarowe przekazywane do minikomputera są wstępnie porządkowane, przy czym oddzielnie buforowane są dane pochodzące z komparatorów oraz dane z liczników. Wydzielono trzy obszary buforowe po 192 bajty, odpowiadające wielkości podstawowej jednostki transmisji danych do pamięci dyskowej. Do pierwszego bufora składowane są dane z komparatorów (blok A struktury z rys. 4), do drugiego — liczniki z bloków *B* i *C*, natomiast do trzeciego — liczniki z bloków *D* i *E* danych pomiarowych. Po zapełnieniu, zawartość buforów przesyłana jest do pamięci dyskowej.

Na zakończenie pomiarów w podzbiorze danych pomiarowych zapisywana jest etykieta końca podzbioru. Dane pomiarowe mogą być przetwarzane bezpośrednio w minikomputerze MERA 306. Dla dużej ilości danych i złożonych procedur przetwarzania minikomputer może okazać się jednak urządzeniem zbyt wolnym. Z tego względu przyjęto założenie, że dane pomiarowe będą przetwarzane w komputerze ODRA 1305. Przeniesienie danych pomiarowych odbywa się za pomocą pamięci taśmowej PT-105. Dane te w oddzielnym przebiegu są przepisywane z dysku na taśmę. Ponieważ program przepisujący zawartość zbiorów dyskowych na taśmę magnetyczną może zapisywać w standarcie ICL, taśma ta jest bezpośrednio dostępna standardowym programom stosowanym na komputerze ODRA 1305.

Wykorzystywane podczas prowadzenia pomiarów w omawianym zestawie oprogramowanie obejmuje dwie grupy procedur. Do pierwszej grupy zaliczane są procedury obsługi monitora sprzętowego w minikomputerze MERA 306. Procedury te zaprogramowane zostały w języku PLAN-M i skompilowane w komputerze ODRA 1305 kompilatorem skrośnym X397 [6]. Procedury te spełniają następujące funkcje:

- przyjmowanie dyrektyw wprowadzanych poprzez klawiaturę drukarki DZM 180
- zapisywanie etykiet początku i końca podzbiorów danych pomiarowych w pamięci dyskowej
- wyprowadzanie na drukarkę komunikatów o przebiegu pomiaru i sytuacjach awaryjnych
- wczytywanie i buforowanie danych pomiarowych z monitora
- przesyłanie danych pomiarowych do pamięci dyskowej
- kopiowanie do pamięci taśmowej zbiorów danych pomiarowych zapisanych w pamięci dyskowej.

Druga grupa procedur przeznáczona jest do przetwarzania danych pomiarowych w komputerze ODRA 1305. Napisane one zostały w językach FORTRAN i PLAN. Procedury te spełniają następujące funkcje:

- odczyt i rozpakowanie etykiety początku taśmowego podzbioru danych pomiarowych
- odczyt pojedynczego bloku informacji z pamięci taśmowej
- odczyt bloku informacji i upakowanie bajtów z zawartością określonych liczników monitora sprzętowego
- wyprowadzanie na drukarce zawartości liczników monitora sprzętowego w postaci histogramów
- wstępna selekcja i uogólnianie zebranych danych pomiarowych
- wyznaczanie statystycznych charakterystyk przebiegów procesów obliczeniowych w badanym systemie komputerowym.

Procedury drugiej grupy zorganizowane zostały w bibliotekę procedur przetwarzania danych pomiarowych. Biblioteka ta zawiera procedury przetwarzania danych zebranych podczas prowadzenia następujących typów pomiarów dwukomputerowego systemu ODRA 1305:

- pomiaru czasu wykonywania wybranych procedur
- pomiaru obciążenia systemu obsługą urządzeń zewnętrznych
- pomiaru czasu biegu jałowego komputerów
- pomiaru czasu pracy komputerów: w stanie sterowania (EXM = 1) oraz w stanie użytkowym (EXM = 0)
- pomiaru czasu trwania wybranych transmisji do/z komputerów
- pomiaru obciążenia transmisjami adaptera międzykomputerowego.

EKSPERYMENTY POMIAROWE

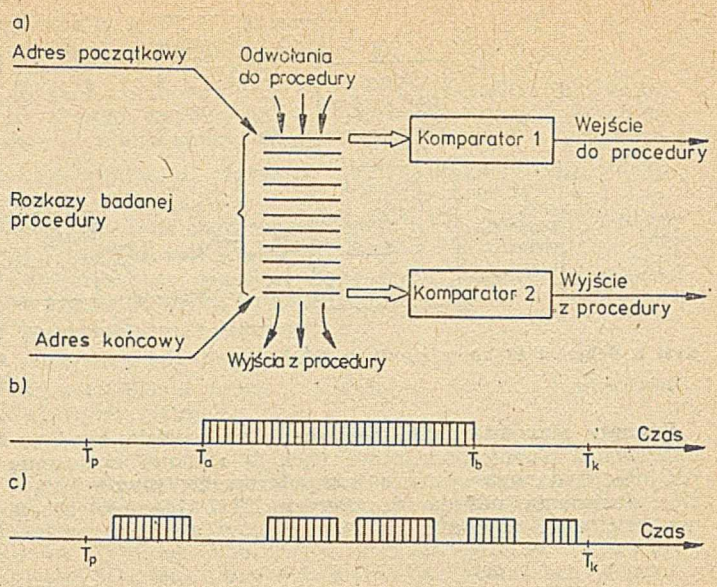
Poniżej zostaną omówione wyniki następujących eksperymentów pomiarowych:

- pomiaru obciążenia komputera procedurą wykonywaną wielokrotnie
- pomiaru procentowego udziału czasu pracy komputera w stanie sterowania.

Pomiar czasu wykonywania wybranej procedury polega na zmierzeniu odcinka czasu pomiędzy następującymi dwoma zdarzeniami:

- wykonaniem pierwszego rozkazu procedury
- wykonaniem ostatniego rozkazu procedury.

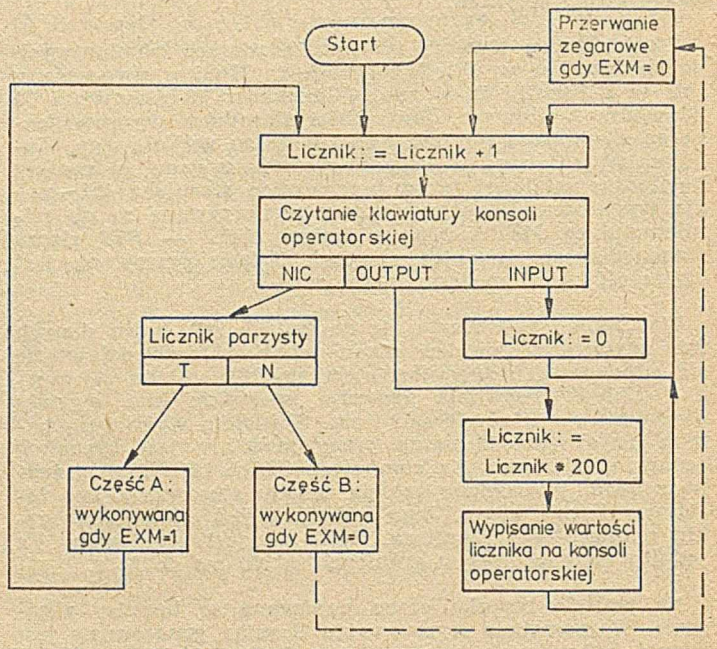
Zasadę tego pomiaru przedstawia rys. 6a. Zdarzenie pierwsze wykrywane jest za pomocą komparatora 1, który porównuje zawartość licznika rozkazów komputera z daną porównawczą, równą adresowi początkowemu procedury. Komparator 2 w podobny sposób wykrywa drugie zdarzenie. Sygnały WEJSCIE DO PROCEDURY i WYJSCIE Z PROCEDURY przekazywane są do sieci logicznej. Na rys. 6b przedstawiono przykład harmonogramu trwania procedury (przedział T_a—T_b) podczas trwania eksperymentu pomiarowego (przedział T_p—T_k).



Rys. 6. Eksperyment pomiarowy
a) zasada pomiaru
b) c) harmonogramy wykonania badanej procedury

Jeżeli procedura dla różnych danych wejściowych charakteryzuje się różnymi czasami wykonania, drogą pomiarów można wyznaczyć średni czas wykonania procedury oraz procentowe obciążenie komputera wykonywaniem tej procedury.

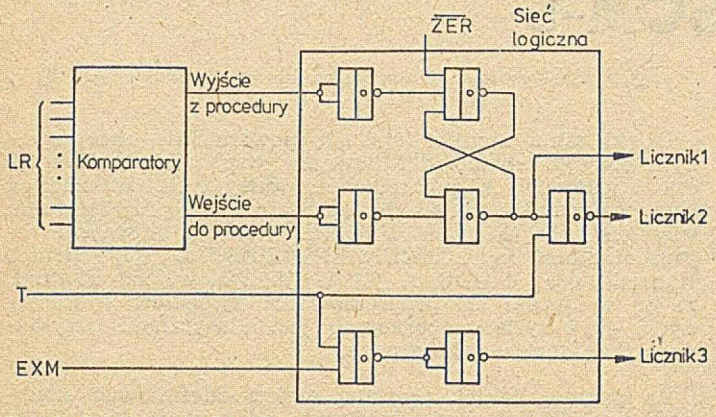
Zasada tego pomiaru jest podobna do przedstawionej powyżej. Komparatory wykrywają każde wejście i wyjście licznika rozkazów z obszaru adresowanego badanej procedury i sterują sumowaniem czasu pobytu w tym obszarze. Na rys. 6c przedstawiono przykład harmonogramu kilku wykonań procedury. Sumaryczny czas wykonywania procedury podzielony przez liczbę skierowanych do niej odwołań daje średnią wielkość czasu wykonania procedury. Ten sam sumaryczny czas wykonywania procedury w stosunku do czasu trwania eksperymentu pomiarowego T_k—T_p pozwala określić procentowe obciążenie komputera badaną procedurą.



Rys. 7. Algorytm badanej procedury

Algorytm badanej procedury obrazuje rys. 7. Procedura została zapisana w języku wewnętrznym i wykonywana jest przemiennie co 200 ms w stanie sterowania komputera (część A) i co 200 ms w stanie użytkowym (część B).

Rys. 8 przedstawia schemat wykorzystanej — w przeprowadzonych eksperymentach — sieci logicznej. Sygnały z licznika rozkazów *LR* przekazywane są do komparatorów, z których sygnały zdarzeń pomiarowych przekazywane są do sieci logicznej. Do sieci tej doprowadzany jest z badanego komputera sygnał zegarowy *T*, odmierzający kwanty czasowe o długości 1,2 μs. Do licznika *1* przesyłane są sygnały zliczające kolejne wejścia do badanej procedury. Licznik *2* przyjmuje sygnały zegarowe odmierzające czas kolejnego wykonania procedury. Do sieci logicznej doprowadzono sondą również stan wskaźnika *EXM* komputera.



Rys. 8. Schemat sieci logicznej

Czas pracy komputera w stanie sterowania zliczany jest w liczniku *3*.

W oparciu o przedstawioną sieć logiczną dokonano dwóch rodzajów pomiarów omówionej procedury. W pomiarach pierwszego rodzaju komparatory użyte zostały do kontrolowania wejść i wyjść z części *A* badanej procedury. Jednocześnie czas pobytu komputera w stanie sterowania zliczany był z pomocą wskaźnika *EXM*. Wyniki dwóch przykładowych pomiarów tego rodzaju przedstawia tab. 1.

Tabela 1.

Numer kolejny pomiaru	Obciążenie komputera częścią A (%)	Praca komputera w stanie sterowania (%)	Średni czas wykonania części A (ms)
1	50,34	50,33	202
2	50,37	50,36	203

Czas trwania pomiarów: 160 s
Długość cyklu pomiarowego: 1 s
Czas wykonania części A: 200 ms

W pomiarach drugiego rodzaju komparatory użyte zostały do kontrolowania wejść i wyjść z części *B* badanej procedury. Jednocześnie czas pobytu komputera w stanie sterowania zliczany był z pomocą wskaźnika *EXM*. Wyniki dwóch przykładowych pomiarów tego rodzaju przedstawia tab. 2.

Tabela 2.

Numer kolejny pomiaru	Obciążenie komputera częścią A (%)	Praca komputera w stanie sterowania (%)	Średni czas wykonania części A (ms)
1	49,92	50,32	199
2	49,67	50,32	197

Czas trwania pomiarów: 160 s
Długość cyklu pomiarowego: 1 s
Czas wykonania części B: 200 ms

* * *

Przedstawione założenia i wyniki eksperymentów pozwalają ocenić możliwości zestawu pomiarowego z monitorem sprzętowym KL-80. Zestaw ten przekazany został do eksploatacji użytkowej. Jednocześnie rozpoczęto prace nad projektem nowego zestawu pomiarowego w wersji przenośnej. Autorzy są zainteresowani w nawiązaniu współpracy z potencjalnymi użytkownikami tego rodzaju sprzętu.

W uruchomieniu zestawu pomiarowego z monitorem sprzętowym KL-80, jego oprogramowaniu i przeprowadzeniu pomiarów eksperymentalnych oprócz autorów niniejszego artykułu uczestniczyli: **mgr Małgorzata Rudnicka, mgr inż. Wiesław Barcikowski, mgr inż. Stanisław Piotrowski, mgr inż. Krzysztof Politowski, mgr inż. Paweł Romaniec i mgr inż. Jacek Wiśniewski.**

LITERATURA

[1] Computer Performance Evaluation — the European Computing Conference on Computer Performance Evaluation. London, 1976

[2] Dokumentacja techniczno-ruchowa pomiarowego monitora sprzętowego KL-80. Wydział Cybernetyki, WAT, Warszawa 1980

[3] Drummond Jr. E.: Analiza i ocena eksploatacji systemów komputerowych. WNT, Warszawa 1979

[4] Duch J.: Metody oceny systemów liczących. INFORMATYKA nr 1/1974

[5] Grzywacz J.: Minikomputer MOMIK-8b/1000. Oprogramowanie. Opis języka PLAN-M. WAT, Warszawa 1977

[6] Kuczborski W.: Zastosowanie monitorów sprzętowych do oceny systemów liczących. Materiały Sympozjum „Organizacja Maszyn Cyfrowych i Mikroprogramowanie", tom I. PWN, Warszawa 1976

[7] Małecki K.: Wykorzystanie monitora programowego do pomiarów i oceny systemów komputerowych. INFORMATYKA nr 4/1976

[8] Terplan K.: Leistungsfähigkeit der Datenverarbeitung. VD München 1977.

STANISŁAW CHROBOT
Przedsiębiorstwo Systemów
Komputerowych MERA SYSTEM
Warszawa

Statyczna i dynamiczna struktura w systemie operacyjnym SOM-5

Zasadniczą innowację, jaką wniosły wysokopoziomowe języki programowania ALGOL, PASCAL itp., była możliwość wyodrębniania w programach dobrze zdefiniowanych modułów. Pojawienie się w ALGOL-u pojęć bloku i procedury pozwoliło łączyć instrukcje w większe całości. Z formalnego punktu widzenia można je traktować jako instrukcje nowe, które mogą być używane do definiowania bloków i procedur wyższego rzędu. Wprowadzenie do PASCAL-a pojęć tablicy i rekordu pozwoliło na podobnych zasadach wyodrębnić moduły danych.

Zwróćmy jednak uwagę, że blok i procedura pozwalają agregować wyłącznie instrukcje (lokalne struktury danych nie są widoczne na zewnątrz bloku i procedury), zaś tablica i rekord pozwalają agregować wyłącznie dane.

Na tym tle przełomowego znaczenia nabiera wprowadzenie do języka SIMULA-67 pojęcia klasy, które pozwala agregować jednocześnie dane i instrukcje.

Klasę można potraktować jako odmianę algolopodobnego bloku, którego zasadniczą własnością jest to, że jego struktura danych nie zanika jeśli sterowanie opuszcza blok. Mówiąc obrazowo, blok taki może być wykonywany „na raty", gdyż przy każdym nowym wejściu do bloku w zmiennych lokalnych zastajemy stan z momentu ostatniego opuszczenia bloku.

Rozwój tego typu konstrukcji językowych doprowadził do wykrystalizowania się metodologii strukturalnego projektowania programów sekwencyjnych. Metodologia ta polega na wprowadzeniu do tekstu programu dobrze zdefiniowanej struktury, czyli na wyodrębnieniu w nim modułów oraz relacji w zbiorze tych modułów. Należy podkreślić, że wprowadzenie do programu struktury nie było celem samym w sobie, lecz jedynie środkiem do celu, jakim jest zredukowanie pozornej złożoności programu do poziomu możliwości intelektualnych człowieka.

Rozwijająca się równolegle teoria programowania współbieżnego doprowadziła do określenia takich pojęć, jak: współbieżne procesy sekwencyjne, własne i wspólne dane tych procesów, regiony krytyczne i wzajemne wykluczanie. Teoria programowania współbieżnego rozwijała się stosunkowo wolno, gdyż dotyczyła obiektów dużo bardziej złożonych niż programy sekwencyjne. Programy współbieżne są bowiem nieodtwarzalne w tym sensie, że kolejne ich wykonania mogą produkować różne zbiory danych wyjściowych dla identycznego zbioru danych wejściowych. Zbiór danych wyjściowych programu współbieżnego zależny jest bowiem nie tylko od zbioru danych wejściowych, ale również od szybkości pojawiania się elementów zbioru wyjściowego, szybkości wykonywania poszczególnych fragmentów programu współbieżnego itp.

Weryfikacja programu współbieżnego nie może się więc sprowadzać do stwierdzenia niezmienności wartości elementów zbioru wyjściowego dla danego zbioru wejściowego, jak się to dzieje w przypadku programów sekwencyjnych, lecz na stwierdzaniu niezmienności relacji między wartościami elementów we wszystkich produkowanych przez program współbieżny zbiorach danych wyjściowych. Mówiąc bardziej obrazowo jest to poszukiwanie niezależnych od czasu relacji w chaosie zależnych od czasu wartości danych wyjściowych. Dla opanowania złożoności programu współbieżnego owocnym stało się przejęcie na potrzeby programowania współbieżnego dotychczasowych osiągnięć sekwencyjnego programowania strukturalnego. Wynikiem tego było wykrystalizowanie się pojęcia monitora jako klasy, która oprócz własności agregowania danych i operacji ma również własność wzajemnego wykluczania.

Dzięki temu monitor stał się powszechnie akceptowanym narzędziem do definiowania wspólnych zmiennych dla współbieżnych procesów sekwencyjnych. W ślad za tym pojawiły się języki dla strukturalnego projektowania współbieżnych, których w chwili bieżącej głównymi reprezentantami są CONCURRENT PASCAL [3] Brinch Hansena oraz MODULA [4] Wirtha. Mówi się również, że są to języki do programowania systemów operacyjnych, gdyż system operacyjny jest klasycznym przykładem programu współbieżnego.

Za pomocą takich języków można zdefiniować strukturę w programie współbieżnym, przy czym kompilator może skontrolować poprawność tego zdefiniowania. Produktem kompilacji jest jednak program w języku maszynowym, w którym nie znajduje żadnego odbicia struktura programu źródłowego. Dlatego też mówi się, że struktura taka jest strukturą statyczną, albo inaczej — strukturą etapu kompilacji, a nie strukturą etapu działania programu.

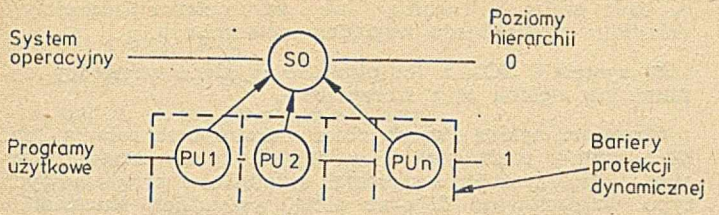
Kontrolowanie struktury statycznej jest z teoretycznego punktu widzenia wystarczające, gdy program traktowany jest jako zamknięty i niezmienny w czasie. W odniesieniu do systemów operacyjnych jest to jednak niewystarczające z następujących powodów.

- Struktura systemu operacyjnego, podobnie jak struktura każdego programu, może zostać zakłócona w wyniku awarii komputera

- System operacyjny nie jest programem zamkniętym w tym sensie, że tworzy on jedynie środowisko dla programów. Program użytkowy może oczywiście uzupełniać system operacyjny do postaci programu zamkniętego, jeśli programowany jest w tym samym języku co system operacyjny i w trakcie kompilacji tworzy z nim nierozłączną całość. Podejście takie jest jednak niepraktyczne, czego najlepszym dowodem jest fakt, że system operacyjny traktuje się raczej jako nieodłączną część sprzętu niż programów użytkowych. Podejście to wymagałoby ponadto stosowania dla programów użytkowych tylko jednego języka programowania, mianowicie tego, w którym zaprogramowany jest system operacyjny. Jeśli jednak zgodzimy się, że to ostatnie ograniczenie jest nie do przyjęcia (przede wszystkim ze względów praktycznych), to musimy stwierdzić, że programy użytkowe pisane w innych językach programowania nie będą tworzyć wraz z systemem operacyjnym zintegrowanej i kontrolowanej na etapie kompilacji struktury.

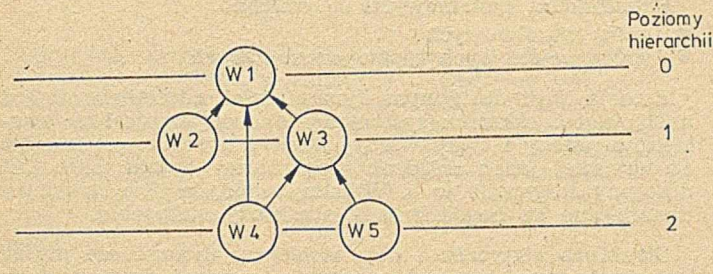
Stąd też bardzo wcześnie wprowadzony został do komputerów układowy aparat ochrony systemu operacyjnego. Aparat ten polega na tym, że system operacyjny rezyduje w niedostępnym dla programu użytkowego obszarze pamięci, a dla wzajemnej komunikacji służą specjalne rozkazy zwane ekstrakodami. Rozpatrując natomiast ten aparat z punktu widzenia programowania strukturalnego, możemy stwierdzić, że służy on do kontrolowania dynamicznej struktury oprogramowania komputera, dynamicznej w tym sensie, że integralność modułów i relacje między nimi są kontrolowane w czasie działania systemu. Strukturę dynamiczną oprogramowania klasycznego wieloprogramowego systemu przetwarzania wsadowego przedstawia rysunek 1. W strukturze tej modułami są: system operacyjny i programy użytkowe, zaś relacja dostępności powoduje, że w każdym programie użytkowym dostępny jest jedynie system operacyjny. Oprogramowanie takie ma zawsze strukturę dwupoziomową, a jego moduły są

dość pojemne. System operacyjny można tu traktować jako jeden olbrzymi monitor, a jego ekstrakody jako procedury wejściowe.



Rys. 1. Struktura dynamiczna systemu wieloprogramowego

Zasadniczą innowacją w systemie operacyjnym SOM-5 [1]) jest taka rozbudowa środkami programowymi układowego aparatu protekcji, aby możliwe było ustanawianie bardziej elastycznych i wielopoziomowych struktur dynamicznych oprogramowania, takich na przykład, jak to przedstawiono na rysunku 2.



Rys. 2. Przykładowa struktura SOM-5

System operacyjny został zaprojektowany strukturalnie w języku wysokiego poziomu, inaczej mówiąc — została zdefiniowana jego struktura statyczna. Ponadto język ten został wyposażony w konstrukcje pozwalające określać, które moduły statyczne tworzyć będą moduły dynamiczne i jakie będą prawa dostępu do tych ostatnich.

Dla celów projektowania systemu operacyjnego SOM-5 zdefiniowano język PROSO. Swój rodowód bierze on z języka CONCURRENT PASCAL. Zasadniczymi innowacjami w języku PROSO są:

- warstwa (konstrukcja do definiowania dynamicznych składników programowych)
- typ wejściowy (dodatkowy atrybut składników programowych)
- koprocedura, ang. coprocedure (zmodyfikowana forma opisu procesów współbieżnych)
- operacje „delay" i „resume" (zminimalizowany aparat synchronizacji procesów w monitorach).

KONSTRUKCJE JĘZYKOWE DO DEFINIOWANIA STRUKTURY STATYCZNEJ I DYNAMICZNEJ

Skupimy obecnie uwagę wyłącznie na konstrukcjach językowych służących do definiowania struktury statycznej i dynamicznej. Omówimy kolejno **klasę** i **monitor** z języka CONCURRENT PASCAL oraz typ **wejściowy** i **warstwę** z języka PROSO, jako dalsze rozwinięcia pojęcia klasy.

Klasę tworzy nazwany blok programowy ze zbiorem wyróżnionych procedur, zwanych dalej procedurami wyjściowymi. Po wykonaniu instrukcji tego bloku, jego struktura danych dostępna jest z zewnątrz za pomocą procedur wejściowych. Z tego powodu klasa traktowana jest jako definicja typu zmiennych.

[1]) Charakterystykę tego sytemu można znaleźć w nr 12/80 INFORMATYKI.

Ogólna postać klasy jest następująca:

```
type C = class (f1:F1;...;fk:Fk);
var m1:M1;...;ml:ML;
proc entry P1(...);begin...end;
...
proc entry Pn(...);begin...end;
begin SC end
```

Atrybuty zdefiniowane na najbardziej zewnętrznym poziomie klasy związane są jej atrybutami lokalnymi. W powyższej definicji są nimi:

- **parametry formalne** *f1,...,fk* typów *F1,...,Fk*, które w momencie inicjowania zmiennej typu C zastępowane są parametrami aktualnymi
- **zmienne lokalne** *m1,...,ml* typów *M1,...ML*, definiujące bieżący stan zmiennej typu C
- **procedury wejściowe** *P1,...,Pn*, które mogą być wywoływane z zewnątrz klasy dla oddziaływania na jej zmienne lokalne i parametry aktualne
- **zdanie inicjujące** *SC*, które nadaje wartości początkowe zmiennym lokalnym klasy.

Zadeklarowanie zmiennej *c* typu *C* wymaga notacji:

```
var c : C
```

Zdanie

```
init c (a1,...,ak);
```

przesyła do zmiennej *c* parametry *a1,...,ak* i powoduje wykonanie zdania inicjującego *SC*. Wywołanie procedury *P1* klasy *c* jest oznaczane przez *c.P1(...);*

Zmienne zdefiniowane za pomocą klas nazywane są składnikami programowymi. Obowiązują dla nich następujące reguły zakresu zmiennych:

- w składniku programowym dostępne są wyłącznie zmienne lokalne i parametry aktualne składnika
- zmienne lokalne składnika programowego dostępne są wyłącznie w tym składniku.

Powyższe reguły zakresu powodują, że klasa jest względnie wyodrębnionym fragmentem programu — w tym sensie, że wyszczególniony jest w jej tekście zbiór wszystkich dostępnych w niej zmiennych.

Oznaczmy przez *Sc* przestrzeń stanów zmiennych lokalnych składnika typu C

```
Sc = M1x...xMl;
```

gdzie: M1,...,Ml są typami zmiennych lokalnych m1,...,ml.

Z faktu, że zmienne te dostępne są z zewnątrz składnika wyłącznie za pomocą jego procedur wejściowych może wynikać, że nie wszystkie punkty tej przestrzeni zostaną osiągnięte, nawet przy dowolnej kolejności oraz dowolnej liczbie wywołań tych procedur.

Innymi słowy definicja klasy wyznacza w przestrzeni swoich zmiennych lokalnych *Sc* pewien podzbiór *Sć*

```
Sć ⊂ Sc
```

W tym sensie klasę można traktować jako definicję abstrakcyjnego typu danych, którego znaczeniem jest zbiór *Sć*. Zdanie logiczne wyznaczające zbiór *Sć* w przestrzeni *Sc* nazywać będziemy **niezmiennikiem klasy C.** Za pomocą klas zdefiniować więc można precyzyjnie strukturę programu, gdyż z samego tekstu klasy wynika znaczenie składników, zaś ze sposobu ich zainicjowania — relacja dostępności w zbiorze tych składników.

Język PROSO dopuszcza definiowanie dodatkowych atrybutów lokalnych klas, zwanych typami wejściowymi. Są nimi klasy, które zostały oznaczone słowem kluczowym *entry*.

Notacja:

```
type C = class(...);
  var m1:M1,...,Ml:Ml;
  type entry D1 = class(...);...end;
  ...
  type entry Di = class(...);...end;
  ...
  begin SC end
```

rozszerza poprzednią definicję klasy *C* o dodatkowe atrybuty, którymi są:

- **typy wejściowe** D1,...,Di, służące do definiowania na zewnątrz składnika typu *C* składników, w których dostępne są zmienne lokalne *m1,...,ml* i parametry aktualne tego składnika.

Deklarowanie, inicjowanie i wywoływanie składników typu *C* pozostaje bez zmian.

Zadeklarowanie zmiennej *d1* typu wejściowego *D1* klasy *C* wymaga notacji:

```
var d1:C.D1;
```

Zdanie *init c.d1(...);* ustanawia zmienną *c* typu *C* **zmienną nadrzędną** składnika *d1*, przesyła do *d1* parametry aktualne i powoduje wykonanie jego zadania inicjującego, a zmienna *d1* staje się automatycznie **zmienną podrzędną** dla *c*. Typy wejściowe mogą być z kolei klasami z typami wejściowymi — tak, że dany składnik może mieć kilka zmiennych nadrzędnych i kilka zmiennych podrzędnych.

Dla składników definiowanych za pomocą tak rozszerzonych klas obowiązują następujące reguły zakresu:

- w składniku programowym dostępne są wyłącznie zmienne lokalne oraz parametry aktualne tego składnika i wszystkich jego składników nadrzędnych
- zmienne lokalne składnika programowego dostępne są wyłącznie w tym składniku oraz we wszystkich jego składnikach porzędnych.

Znaczenie klasy z typami wejściowymi definiuje się za pomocą niezmienników dla klasy obejmującej i wszystkich jej klas wejściowych. Jest to możliwe, gdyż — podobnie jak dla klasy prostej — reguły zakresu określają zbiór wszystkich składników, w których dostępne są zmienne lokalne dowolnego składnika.

Niektóre klasy nie będące typami wejściowymi mogą być wyróżnione za pomocą słowa kluczowego *layer*. Są one wtedy nazywane **warstwami.** Niezależnie od statycznych reguł zakresu, dla warstw obowiązują dodatkowo dynamiczne reguły zakresu, pozwalające definiować struktury dynamiczne: ogólną i szczegółową, które zostaną opisane w dalszej części artykułu.

Niezależnie od tych własności, z warstwą można związać opcjonalnie własności wzajemnego wykluczania się w czasie operacji wejściowych warstwy i jej typów wejściowych. Przez analogię do sposobu rozróżniania w języku CONCURRENT PASCAL klasy i monitora, w języku PROSO rozróżniać można warstwę monitorową i niemonitorową. Na przykład notacje:

```
typeA = layer class(...);...end,
type B = layer monitor(...);...end
```

definiują niemonitorową warstwę A oraz monitorową warstwę B.

STATYCZNA I DYNAMICZNA STRUKTURA SOM-5

Strukturę statyczną operacyjnego SOM-5 tworzy zbiór składników programowych oraz relacja „być dostępnym statycznie", zdefiniowana w zbiorze tych składników.

Składnikami programowymi są zmienne:
- typów warstwowych: monitorowych i niemonitorowych
- typów wejściowych typów warstwowych
- typów wewnętrznych, zdefiniowanych za pomocą klas i typów wejściowych tych klas.

Relacja „być dostępnym statycznie" jest relacją dwuargumentową w zbiorze składników programowych, która definiowana jest następująco: składnik *a* dostępny jest statycznie w składniku *b* wtedy i tylko wtedy, gdy *a* jest zmienną lokalną lub parametrem aktualnym składnika *b*, bądź zmienną lokalną lub parametrem aktualnym składnika nadrzędnego względem składnika *b*.

W systemie SOM-5 definiowane są dwie struktury dynamiczne: ogólna oraz szczegółowa.

Strukturę ogólną tworzy zbiór warstw oraz relacja „być dostępnym", zdefiniowana w zbiorze tych warstw. Warstwę *L* tworzy zbiór wszystkich składników programowych typu warstwowego *L* oraz zbiór wszystkich składników typów wejściowych typu *L*. Relacja „być dostępnym" jest relacją dwuargumentową w zbiorze warstw i definiowana jest następująco: warstwa *L1* dostępna jest w warstwie *L2* wtedy i tylko wtedy, gdy w warstwie *L2* istnieje co najmniej jeden składnik programowy *a*, w którym dostępny jest statycznie składnik programowy *b* wchodzący w skład warstwy *L1*.

Strukturę szczegółową tworzy zbiór składników systemowych, zbiór warstw oraz relacja „być dostępnym dynamicznie", zdefiniowana w produkcie kartezjańskim tych zbiorów.

Składnikami systemowymi są zmienne:
- typów warstwowych
- typów wejściowych typów warstwowych.

Relacja „być dostępnym dynamicznie" zdefiniowana jest następująco: składnik systemowy *a* dostępny jest w warstwie *L* wtedy i tylko wtedy, jeżeli w warstwie *L* istnieje co najmniej jeden składnik programowy *b* taki, że *a* jest dostępny statycznie w *b*. Widzimy więc, że obie struktury dynamiczne są uogólnieniami struktury statycznej.

Struktura statyczna i obie struktury dynamiczne muszą być strukturami hierarchicznymi, co oznacza, że w grafach tych struktur nie mogą występować pętle. W konsekwencji we wszystkich strukturach można wyróżnić poziomy hierarchii, co z kolei pozwala na stosowanie przy projektowaniu systemu metod „od góry do dołu" (ang. *top-down*) lub „od dołu do góry" (ang. *bottom-up*).

Struktura ogólna wyznacza poziomy hierachii w zbiorze warstw, a co za tym idzie maksymalną liczbę zagnieżdżeń procedur wejściowych składników systemowych w procesach współbieżnych. Dzięki temu możliwe jest wyznaczanie maksymalnej zajętości pamięci przez stos tzw. rekordów aktywacji, tworzony przy zagnieżdżaniu procedur wejściowych składników systemowych. Upraszcza to w znaczny sposób problem dynamicznego przydziału pamięci dla tych rekordów.

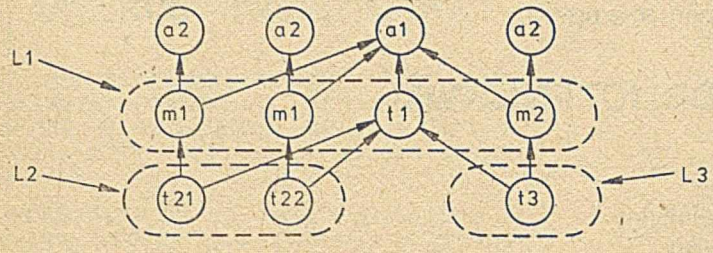
Struktura szczegółowa pozwala wyznaczyć w zbiorze składników wchodzących w skład warstwy *L* podzbiory składników dostępnych w warstwach poziomu wyższego niż poziom warstwy *L*. Unika się w ten sposób szkodliwej interferacji dynamicznej warstw wyższego poziomu za pomocą składników systemowych zdefiniowanych w warstwie *L*.

Przykład:

Rozważmy schemat programu zawierający definicję trzech typów warstwowych: *L1*, *L2* i *L3*. Typ *L1* zawiera definicję wewnętrznej klasy *A* i wejściowej klasy *M*. Zmienne *a1* i *a2* typu *A* są odpowiednio zmiennymi lokalnymi typu warstwowego *L* oraz typu wejściowego *M*. Każdy z typów warstwowych *L2* i *L3* posiada jako parametr zmienną typu *L*, co umożliwia zadeklarowanie w tych typach zmiennych lokalnych m1 i m2 typu wejściowego M. W zdaniu inicjującym zadeklarowano po jednej zmiennej typu L1 i L3 oraz dwie zmienne typu L2:

```
type L1 = layer monitor
  type A = class...end;
  var a1:A;
  type entry M = class
  var a2:A;...;
  begin init a1,...end;
  begin init a1,...end;
type L2 = layer class (t1:L1);
```

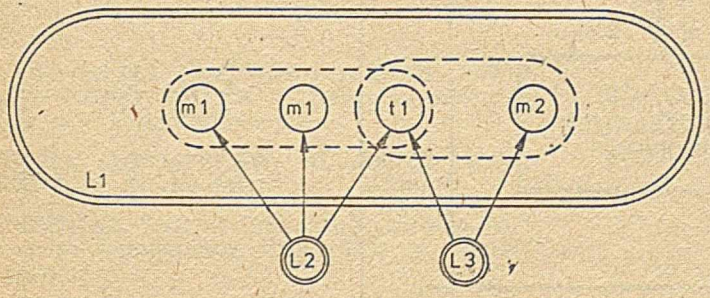
```
var m1:L1.M;...;
begin init t1.m1;...end;
type L3 = layer class (t1:L1);
var m2:L1.M;...;
begin init t1.m2;...end;
var t1:L1, t12, t22:L2; t3:L3;
begin init t1, t12 (t1), t22 (t1), t3 (t1);...end;
```



Rys. 3. Struktura statyczna przykładowego programu

Statyczną strukturę tego programu przedstawia rysunek 3. Dla przykładu, w zmiennej *t3* dostępna jest *t1* jako parametr aktualny oraz *m2* jako zmienna lokalna. Z kolei w *m2* dostępna jest jej zmienna lokalna *a1* oraz zmienna lokalna *a1* zmiennej *t2*, gdyż *t1* jest zmienną nadrzędną *m1*. Na rysunku 3 liniami przerywanymi objęto składniki systemowe warstwy *L1*, *L2* i *L3*, dzięki czemu łatwo wywnioskować dynamiczną strukturę ogólną systemu.

Szczegółową strukturę dynamiczną systemu przedstawia rysunek 4, z którego widać, że warstwa *L2* nie posiada dynamicznego dostępu do zmiennej lokalnej *m2* składnika wchodzącego w skład warstwy *L3*.



Rys. 4. Szczegółowa struktura dynamiczna przykładowego programu

BIBLIOTEKA WARSTW I GENERATOR SYSTEMU

W ślad za językiem CONCURRENT PASCAL wprowadzono do języka PROSO dalsze ograniczenia syntaktyczne, które pozwalają w sposób istotny uprościć proces translacji programu. Przyjęto więc, że definicje typów warstwowych nie mogą być zagnieżdżane. Mogą się więc one pojawiać jedynie w najbardziej zewnętrznym bloku, zwanym systemem. Dzięki temu żadne typy ani procedury wewnętrzne jednej warstwy nie są dostępne w innych warstwach. Przyjęto również zasadę, że pamięć dla składnika systemowego przydzielana jest na etapie inicjowania w tej warstwie, w której zdefiniowano jego typ. W tej sytuacji środowisko, jakie tworzy dana warstwa na potrzeby translacji innych warstw, zawiera wyłącznie zestaw identyfikatorów jej artybutów wejściowych oraz typów parametrów tych atrybutów.

Ograniczenia te pozwalają w stosunkowo prosty sposób zorganizować niezależną translację warstw oraz prosty aparat konsolidacji takich warstw. Z kolei ze względu na dynamiczną protekcję warstw mogą być one programowane w różnych językach programowania, również takich, które nie kontrolują struktury statycznej, np. w ASSEMBLERZE.

Dynamiczna protekcja warstw i łatwa ich konsolidacja daje olbrzymie ułatwienia przy organizowaniu pracy nad systemem i pozwala patrzeć na system w zupełnie nowym świetle. Użytkownik nie dostaje bowiem systemu operacyjnego jako monolitycznego tworu dostosowanego przez producenta do jego konfiguracji, lecz bibliotekę warstw z prostym w obsłudze generatorem systemu, za pomocą którego może on z warstw bibliotecznych składać różne wersje systemu.

* * *

W chwili obecnej trudno jeszcze o generalną ocenę przyjętej metody projektowania. Uruchomiono bowiem pierwszą wersję systemu operacyjnego SOM-5, która przechodzi wstępną eksploatację na trzech instalacjach. Wersja ta, zaprojektowana w języku PROSO, została (z powodu braku kompilatora) zaprogramowana w makroasemblerze. Ale przy takim właśnie sposobie uruchamiania systemu struktura dynamiczna okazuje się najbardziej potrzebna. Dynamiczna protekcja warstw chroni bowiem wcześniej uruchomione warstwy przed zniszczeniem przez bieżąco uruchamiane warstwy. Należy również podkreślić, że struktura statyczna, pozostająca tylko w stadium projektu, oddaje duże usługi projektantowi, chroniąc go przed popełnieniem wielu błędów. Szacuję, że przy projektowaniu SOM-5 popełniono o 60—80% mniej błędów zależnych od czasu, niż w projektowanym tradycyjnie 12 lat temu systemie operacyjnym TRAN.

Inne wnioski dotyczą architektury maszyny, na której struktura dynamiczna jest realizowana. Otóż programowa nadbudowa aparatu dynamicznej ochrony warstw jest czaso- i pamięciochłonna, powinna więc być w znacznym stopniu zrealizowana sprzętowo. W przypadku MERY 400 postulat ten dotyczy nie tylko rozbudowy, ale nawet i zmiany istniejącego obecnie układu ochrony pamięci. Jest on bowiem w sposób skrajny przystosowany do wieloprogramowego systemu przetwarzania wsadowego.

LITERATURA

[1] Chrobot Stanisław: LAYER — a language construction for concurrent structural program design, Information Processing Letters, vol. 4, nr 5, February 1976

[2] Chrobot Stanisław: Analiza podstawowej warstwy systemu operacyjnego ze względu na własności programowania procesów współbieżnych. Rozprawa doktorska WAT, Warszawa, 1976

[3] Hansen Per Brinch: CONCURRENT PASCAL, a programming language for operating system design. Information Science Technical Report No 10, 1974

[4] Wirth Niklaus: MODULA: a language for modular multiprogramming. ETH Zurich.

Na prośbę Profesora Stefana Paszkowskiego uprzejmie zawiadamiamy, że był On opiniodawca przekładu książki C. J. Date'a „Wprowadzenie do baz danych", wydanej nakładem WNT w r. 1981.

Redakcja Informatyki
Wydawnictw Naukowo-Technicznych

CENTRUM KOMPUTEROWYCH SYSTEMÓW AUTOMATYKI I POMIARÓW

MERA-ELWRO

PRACOWNIA PROJEKTOWANIA SYSTEMÓW

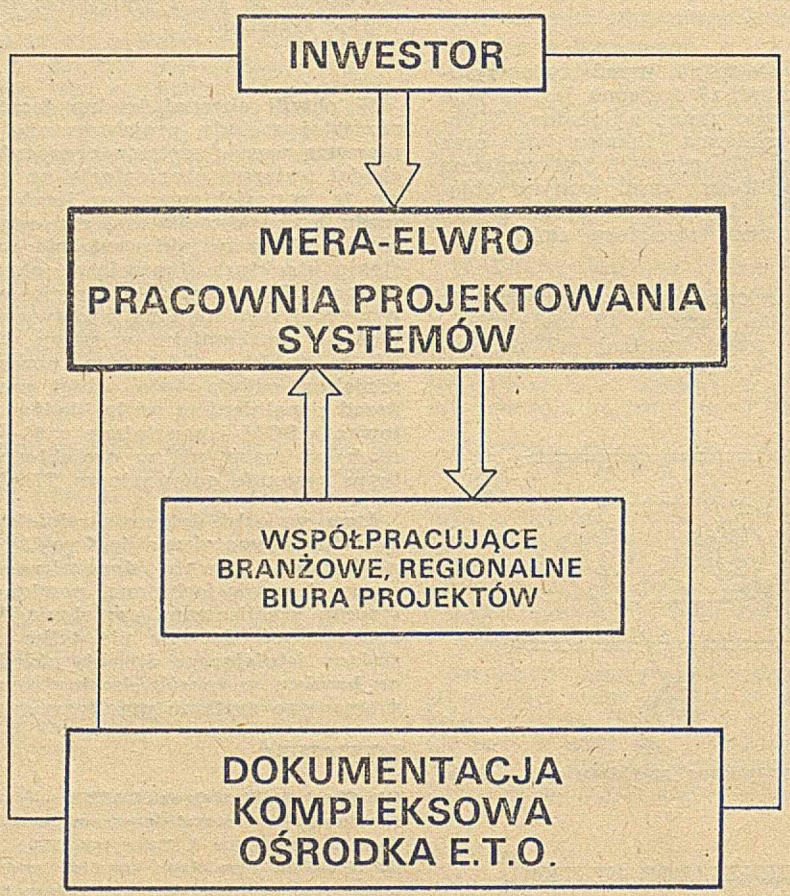
ul. Obornicka 66/68
51-114 Wrocław

Telefon: 25 31 47
Telex: 0712789 ppe pl

OFERUJE: kompleksową, wielobranżową, dokumentację nowych obiektów, adaptacje istniejących

OŚRODKÓW TECHNIKI OBLICZENIOWEJ

- założenia techniczno-ekonomiczne
- projekt techniczny technologiczny
- projekt techniczny budowlano-instalacyjny
- nadzór autorski nad realizacją ośrodka
- szkolenie obsługi



DOKUMENTACJA TECHNOLOGICZNA

- SPRZĘT KOMPUTEROWY
- OPROGRAMOWANIE
- ORGANIZACJA
- TECHNOLOGIA
- WYPOSAŻENIE TECHNICZNE
- OPRZYRZĄDOWANIE

DOKUMENTACJA BUDOWLANO-INSTALACYJNA

- ARCHITEKTONICZNA
- BUDOWLANA
- WOD. KANALIZACYJNA
- C.O.
- KLIMATYZACJA
- P. POŻAROWA

WIELOLETNIE DOŚWIADCZENIE
BEZPOŚREDNI KONTAKT Z PRODUCENTEM, DOSTAWCĄ KOMPUTERÓW
KRÓTKIE TERMINY WYKONANIA DOKUMENTACJI

EO/685/K/81

Spis treści rocznika 1981

ARTYKUŁY PROBLEMOWE

nr str.

BAŃKOWSKI JACEK, DOBOSZ JAROSŁAW, ROMAŃSKI STANISŁAW, SZYMAŃSKI BOLESŁAW, ZABŻA-TARKA EWA — Oprogramowanie relacyjnego dostępu do baz pakietu CDS ISIS 1 17

BARTYŚ MICHAŁ — Język symboliczny i translator dla systemu CAMAC z procesorem 131 11—12 23

BELCZAK STANISŁAW, CZAJA JANUSZ — Język BASIC dla mikrokomputera INTEL 8080 3 11

BIELENINIK EDWARD — Nowe zastosowania teleinformatyki 7—8 8

BRZOZOWSKA-REITER DANUTA, WIŚNIWSKI ZDZISŁAW — SYWIN — generacyjny system wyszukiwania i wydruku informacji 2 12

CARLSON PIOTR K. — Zastosowanie języka CDL 2 do konstrukcji oprogramowania dla MERY 400 7—8 4

CHMURZYŃSKI JERZY, LIDERMAN KRZYSZTOF — Pomiary wydajności systemu komputerowego monitorem sprzętowym KL-80 11—12 13

CHOMICKI JAN — Abstrakcyjne typy danych 3 13
CHROBOT STANISŁAW — Statyczna i dynamiczna struktura w systemie operacyjnym SOM-5 11—12 18

DAWIDOWSKI JAN, OWCZARCZAK PIOTR, WIŚNIEWSKI MAREK — Badanie sprawności systemów komputerowych 4 15

DOBIJA MIECZYSŁAW, KOLARZYK KAZIMIERZ — Koszty informacji sytemowej 3 8

DREWNIAK WIT — Organizacja serwisu technicznego komputerów firmy ICL w Wielkiej Brytanii 1 10

GACKOWSKI ZBIGNIEW — Perspektywy komputeryzacji badania i projektowania systemów (uogólniony analizator) 1 6

GWIAZDA JANUSZ — Czy informatyce potrzebna jest reforma 11—12 4

GOVENLOCK JACEK, KARWAT ANDRZEJ, MARCIŃSKI WŁODZIMIERZ, PSIURSKI WACŁAW — Pakiet Programów Przetwarzania Zbiorów Dyskowych dla MERY 400 9—10 23

HOŁYŃSKI MAREK — Symulacja procesu nauczania w systemie PLATO 2 4

JAEGERMANN TADEUSZ — Ochrona danych obowiązuje 7—8 16

KOTT RYSZARD K., SAPIECHA KRZYSZTOF — Rola sieci działań w nauczaniu programowania 1 4

KULIKOWSKI JULIUSZ LECH — Czy informatyka polska musi tkwić w impasie? 5—6 4

LEWICKI WŁODZIMIERZ — DOORS — system wyszukiwania informacji w języku naturalnym
Część 1 4 10
Część 2 5—6 9

nr str.

LEWOC JÓZEF B. — O zawodzie inżyniera informatyka 3 17

LEWOC JÓZEF B., NAWROT JERZY, URBANEK ADAM — Jak kształcić inżynierów informatyków? 4 18

LIBURA LUDWIK — Minisystem wyszukiwania informacji o materiałach 3 6

MACIASZEK LESZEK A. — Słownik-skorowidz. w procesie powstawania i utrzymywania baz danych 7—8 11

MAŁECKI KRZYSZTOF — Komputerowe rozpoznawanie mowy 4 4

MATWIEJCZUK JERZY — ODRA 1300 jako system pośredniczący w programowaniu mikroprocesorów 7—8 13

MERCIK ANDRZEJ — Człowiek w kontakcie z komputerem 2 18

ORŁOWSKI HENRYK — Przemysł minikomputerowy a informatyka w Polsce 11—12 6

OSTASIEWICZ WALENTY — Komputery w dydaktyce 7—8 19

PŁUC BOGUSŁAWA, RUTA RYSZARD — Zautomatyzowany wywiad lekarski 1 14

Projektowanie języka użytkownika (oprac. **KLEPACZ WŁADYSŁAW)** 3 19

RAPORT

Ocena polskiego przemysłu komputerowego w latach 1971—1980 oraz stan zaspokojenia potrzeb informatyki przez ten przemysł 9—10 4

Ocena przedsięwzięcia K-202 9—10 8

RYZNAR ZYGMUNT — Projektowanie strukturalne 1 21

SEKUŁA ZOFIA — Wdrażanie systemów informatycznych 2 15

SKUPIŃSKI MARIAN, WIESNOWSKI ANTONI — Oprogramowanie systemu wielomaszynowego komputerów JS jako elementu sieci komputerowej 5—6 12

STABROWSKI MAREK — Projektowanie podstawowego oprogramowania systemów mikrokomputerowych 2 9

SYC CZESŁAW — Teleks w służbie informatyki 9—10 18

ŚLUSAREK MACIEJ — Realizacja programów współbieżnych w systemie ODRA 1305 3 4

WIESNOWSKI ANTONI — Technologiczne wersje systemów operacyjnych komputerów Jednolitego Systemu 4 13

ZALEWSKI JANUSZ — ADA — nowy język programowania (1), Powstanie języka 11—12 10

INFORMATYKA WĘGIERSKA

ARATÓ MATYAS — Badania naukowe w dziedzinie zastosowań komputerów 5—6 25

GAL FERENC — Państwowe Przedsiębiorstwo Techniki Komputerowej 5—6 28

	nr	str.
KOCSIS ANDRAS — Centralny ośrodek szkolenia specjalistów techniki obliczeniowej	5—6	23
KONDRICZ JÓZSEF — Sieć usług informatycznych	5—6	19
KOVACS GYOZO — Towarzystwo im. Jánosa Neumanna organizacją zawodową informatyków węgierskich	5—6	33
MATÓK GYORGY — Obieg informacji o informatyce	5—6	31
NEMETH LÓRANT — Informatyka na Węgrzech	5—6	16
PARIS GYORGY — Nauczanie informatyki	5—6	21
VAMOS TIBOR — Instytut Techniki Obliczeniowej i Automatyzacji	5—6	27

ALGORYTMY

Podręczna biblioteczka programisty — **Szałas Andrzej, Świrski Zbigniew**	11—12	27

Z KRAJU

Co nowego w diagnostyce? — **Budka Marian, Hławiczko Andrzej, Łent Bogdan, Piecha Jan**	2	20
O mikroprocesorach w Polsce — **Zalewski Janusz**	2	23
INFOGRYF'80 — Korzyści i nie spełnione nadzieje — **Bernatowicz Krystyn**	2	27
SPIS'80 — Źródła danych w centralnych systemach informatycznych — **Dąbrowski Henryk**	5—6	35
Podsumowanie INFOGRYFU'80	5—6	37
O komputerach w automatyce — **Zalewski Janusz**	7—8	21
Jaki powinien być nowy polski minikomputer? — **Zalewski Janusz**	7—8	23
Racje SPIS-u — Oprac. **Gluza Z.**	9—10	31
Systemy operacyjne JS — **Leonarski Dariusz**	9—10	35
Krajowy sprzęt dla informatyki na 53 MTP — **Żebrowski Jacek**	11—12	29

ZJEDNOCZENIE INFORMATYKI

O czym zapomnieć, co zapamiętać — **Bernatowicz Krystyn**	1	25
Zjednoczenie Informatyki — 15 lat działalności — **Pawlak Tomasz**	1	29
SYMLEK — System oceny hodowlanej krów — **Illukowicz Barbara, Stramska Zofia**	2	29
Systemy Zarządzania Bazą Danych — **Pasula J.**	2	30
System rezerwacji miejsc sypialnych i kuszetek — **Świtalska-Jeleńkowska Małgorzata**	3	25
RURY — system obliczania samokompensacji rurociągów — **Gawryś Lech, Wiszniewski Jan**	3	29
SYMES — analiza statyczna układów powłokowo-prętowych — **Wyrzykowski Grzegorz**	4	26
Oferta Działowego Ośrodka Informacji CPiZI	5—6	15
PARYS — pakiet procedur rysunkowych — **Wyrzykowski Grzegorz**	5—6	49
ICES-SIGMA — podsystem graficznej prezentacji obliczeń wytrzymałościowych konstrukcji — **Stefankiewicz Alfred**	5—6	50
BAZY DANYCH — konieczność czy sztuka dla sztuki? — **Piasecki Józef**	7—8	29
System dla przedsiębiorstw handlu zagranicznego — **Żmudzka Kazimiera**	9—10	39
Miernictwo informatyczne — możliwości i potrzeby — **Peterseil Adam**	9—10	41
Systemy informatyczne ZETO Poznań — **Adamczewski Henryk**	11—12	31
Metodyka tworzenia systemów informatycznych — **Ronkowski Roman**	11—12	34

POLSKIE TOWARZYSTWO INFORMATYCZNE

Powstaje Stowarzyszenie informatyków polskich **(J. P.)**	1	33
Organizacji ciąg dalszy **(M. H.)**	2	26
Przygotowanie do Zjazdu	3	24
Zjazd założycielski — **Gluza Z.**	5—6	39
— Uchwała	5—6	41
— Wybrane władze PTI	5—6	41
Z referatu programowego: — Struktura i program	5—6	39
— Członkowie i organa	5—6	40
— Obecny stan informatyki	7—8	26
— Do czego dążyć w polskiej informatyce?	7—8	27
— Cele Polskiego Towarzystwa Informatycznego	7—8	27
Statut Polskiego Towarzystwa Informatycznego	9—10	27
Bieżące informacje **(Z.G)**	9—10	43

ZWIĄZKI ZAWODOWE

„Solidarność" o taryfikatorze — **Jacek Doliński**	1	33
Biuletyn NSZZ PI **(Z.G)**	1	34
„Solidarność" wśród informatyków — **Boni Piotr, Doliński Jacek**	2	28
„Solidarność" w ZETO — **Lipiński Włodzimierz**	3	31
NSZZPI o problemach płacowych — **Fiutowski Bogdan**	3	31
Prace nad Układem Zbiorowym — Głos „Solidarności" — **Mrozik Włodzimierz**	3	32
— Głos NSZZPI — **Madejski Wojciech**	3	32
Aktualne problemy NSZZPI	4	23

	nr	str.
Dla naszego dobra — **Fiutowski Bogdan**	4	24
Walka o samodzielność i samorządność — **Młynarski Mariusz**	4	24
Horyzont związkowy — oprac. **Gluza Z.**	5—6	42
Uchwała reprezentatywnego przedstawicielstwa załóg sieci ETOB	5—6	48
Komunikat NSZZ „Solidarność" sieci informatycznych GUS, ZETO, ETOB, CEKAR	5—6	48
Zreformowane przedsiębiorstwo informatyczne — podstawowe kierunki przemian — **Dąbrowski Piotr**	7—8	24
Komisje Koordynacyjno-Porozumiewawcze „Solidarności" w sieciach informatycznych — **Lipiński Włodzimierz**	9—10	37
Czekać dłużej? — **Fiutowski Bogdan**	9—10	38

ZE ŚWIATA

	nr	str.
Komputeryzacja a zatrudnienie pracowników umysłowych w RFN (oprac. **W. Klepacz**)	1	35
Trendy na europejskim rynku komputerów (oprac. **T.J.**)	2	32
Bezpośredni dostęp do 150 tys. dokumentów prawnych (oprac. **W.K.**)	2	34
Bank danych krwi (oprac. **W.K.**)	2	34
Zdalny serwis (oprac. **W.K.**)	2	34
COMPSTAT 1980 — **Bartkowiak Anna**	2	35
Oprogramowanie dla międzynarodowych operacji bankowych (oprac. **W.K.**)	2	35
Bezprzewodowe wprowadzanie danych (oprac. **W.K.**)	2	36
Mikroelektronika — rewolucja nie dokonana (oprac. **A.R.**)	3	33
UNIVAC zwiększa produkcję półprzewodników (oprac. **W.K.**)	3	34
Współpraca UNIVAC z ChRL (oprac. **W.K.**)	3	34
Zaskakujące wyniki badań ankietowych — **Klepacz Władysław**	4	28
Automatyzacja drukarni — **Klepacz Władysław**	4	29
Jubileusz serii UNIVAC 1100 — **Klepacz Władysław**	4	30
Koncern Springera intensyfikuje komputeryzację (oprac. **W.K.**)	4	30
Ogrzać się komputerem (oprac. **E.B.**)	5—6	51
MEDICAL INFORMATICS EUROPE'81 — **Jasiński Piotr J.**	7—8	31
Algorytmy'81 — **Bazewicz Mieczysław**	7—8	32
Nowy numeryczny układ scalony INTEL 8087 (oprac. **K.M.**)	7—8	33
Rozwój sprzętu coraz bardziej podporządkowany człowiekowi — oprac. **Klepacz Władysław**	9—10	43
COMNET'81 — **Kulikowski Juliusz L.**	9—10	46
Najnowsze tendencje w dziedzinie komputerowych systemów sterowania — **Zalewski Janusz**	11—12	38

RECENZJE

	nr	str.
Wiarygodność informacji — **Gogołek Włodzimierz**	1	39
O realizacji komputerowych układów automatyki — **Zalewski Janusz**	3	35
„Informatyk to jak hydraulik..." — **Kulikowski Juliusz L.**	4	31
„Informatyka nauką nie jest" — **Turski Władysław M.**	4	33
Matematyczne ozdobniki — **Blike Andrzej, Mazurkiewicz Antoni**	4	34
Historia w karykaturze — **Łukaszewicz Leon**	4	35
Wykład o SIMULI — **Winkowski Józef**	5—6	52
Nieporozumienie — **Empacher Adam B.**	7—8	33
Aktualizacja sprzed lat — **Stępniewski Jan**	7—8	35
Czy komputer może decydować? — **Zawadzki Stefan W.**	7—8	35
Encyklopedie języków wysokiego poziomu — **Bonkowicz-Sittauer Stanisława**	7—8	36
Wykład o LISPIE — **Loska Jerzy**	9—10	47
Po(d)stępy bazorządctwa — **Empacher Adam B.**	11—12	40

LISTY

	nr	str.
Kaleki system dla spółdzielni mieszkaniowych — **Rogowska Alicja**	1	38
O pracy niektórych rodzimych producentów oprogramowania **(H.Z)**	2	39
Spółdzielczość mieszkaniowa o systemie — **Gotfalski Zbigniew**	2	40
Jak koordynować? — **Kierczuk Elżbieta**	3	36
Sugestie praktyka — **Drobiszewski Jerzy**	3	37
Co dalej ze skomputeryzowaną rachunkowością? — **Chądzyński Zdzisław**	3	38
Wykorzystać informatykę — **Jacek Ewald**	4	36
Moje postulaty — **Grzesiak Ryszard**	4	37
Odnowa INFORMATYKI? — **Żebrowska Krystyna**	7—8	III okł.
W sprawie napaści czworga profesorów na mnie... — **Targowski Andrzej**	9—10	50
Uczone kłamstwa — **Targowski Andrzej**	9—10	53
Bariery komputeryzacji w przedsiębiorstwie przemysłowym — **Błaszczak Bolesław**	9—10	55
Infologiczne problemy przetwarzania danych — **Stefanowicz Bogdan**	11—12	43

TERMINOLOGIA

	nr	str.
Adresowanie — **Zalewski Janusz**	1	37
Mikroprocesory — **Zalewski Janusz**	2	37
Adresowanie — **Zalewski Janusz**	3	39
Teleinformatyka — **Zalewski Janusz**	4	38
Teleinformatyka — **Zalewski Janusz**	5—6	54
Teleinformatyka — **Zalewski Janusz**	7—8	37
Szyna na magistrali — **Zalewski Janusz**	9—10	48
O pisowni nazw języków programowania — **Zalewski Janusz**	11—12	42

KONFERENCJE

	nr	str.
Pierwsza Brytyjska Konferencja nt. Baz Danych	1	9
Bureautique AFCET — SICOB'81	1	13
Kongres Rozpoznawania Obrazów i Sztucznej Inteligencji	2	17
PORTEX'81 w Hamburgu	2	36
COMNET'81	3	16
Wielkie bazy danych	3	23
SOFSEM'80 — **Klaczak Jerzy J.**	3	24
Systemy operacyjne komputerów Jednolitego Systemu — **Breczko Aleksy**	3	24
XI Sympozjum Sekcji Cybernetyki Medycznej TIP	7—8	7
CAMAC w zastosowaniach przemysłowych	7—8	20
SPIS'81	7—8	38
INFRA'82	9—10	17
SOFTENG II — **Winiarski Maciej S.**	9—10	36

POGLĄDY

	nr	str.
Apel informatyków niedocenionych — **Gluza Zbigniew**	4	III okł.

System CAMAC stanowi zbiór norm określających zasady współpracy komputerów ze środowiskiem przy pomiarach i sterowaniu. Przydatność systemu została w ciągu kilkunastu lat potwierdzona (także w Polsce) licznymi zastosowaniami. W roku ubiegłym opublikowaliśmy dwa artykuły stanowiące wprowadzenie do niniejszej tematyki: S. Koślacza i K. Rzymowskiego „CAMAC — blokowy system elektroniczny do automatyzacji pomiarów i sterowania" (nr 2/1980) oraz J. Zalewskiego „I Konferencja CAMAC'80" (nr 6/1980). Obecnie rozpoczynamy przegląd zagadnień bardziej szczegółowych, zachęcając jednocześnie projektantów i użytkowników systemu do nadsyłania następnych wypowiedzi. **(Red.)**

MICHAŁ BARTYŚ
Instytut Automatyki Przemysłowej
Politechnika Warszawska

Język symboliczny i translator dla systemu CAMAC z procesorem 131

W artykule przedstawiono syntetyczny opis języka adresów symbolicznych ASCAM 131 i translatora przeznaczonego dla zestawów aparatury pomiarowo-sterującej systemu CAMAC z procesorem autonomicznym typu 131 oraz pamięcią operacyjną o minimalnej pojemności 1 K słów 24-bitowych. Scharakteryzowano także wykorzystywany system operacyjny.

Wielu użytkowników aparatury pomiarowo-sterującej systemu CAMAC [4], zawierającej procesor autonomiczny 131 [5] oraz pamięć operacyjną (1 K słów 24-bitowych) typu 201, nie posiada do chwili obecnej odpowiednich narzędzi programowych które umożliwiłyby efektywne wykorzystanie tego sprzętu. „Firmowe" oprogramowanie wymaga zastosowania modułu pamięci operacyjnej o minimalnej pojemności 2 K słów 24-bitowych [3]. Dotychczasowe trudności technologiczne krajowej produkcji modułów takiej pamięci sprawiły, iż przydatność oferowanego oprogramowania dla wielu użytkowników sprzętu CAMAC jest w chwili obecnej niewielka.

W tej sytuacji w Instytucie Automatyki Przemysłowej Politechniki Warszawskiej opracowano podstawowy zestaw środków programowych, umożliwiających efektywne i w miarę wygodne wykonanie oprogramowania użytkowego na zestawach autonomicznych CAMAC z procesorem 131 i pamięcią typu 201. Oprogramowanie to obejmuje translator języka adresów symbolicznych ASCAM 131 [1] oraz konwersacyjny program redagujący EDCAM 131 [2].

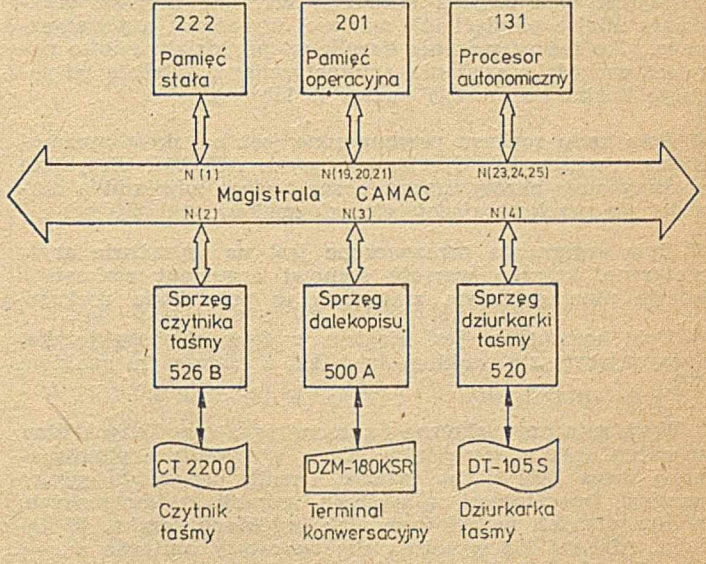
Mgr inż. Michał BARTYŚ ukończył w 1973 r. Wydział Mechaniki Precyzyjnej Politechniki Warszawskiej. Pracuje w Instytucie Automatyki Przemysłowej Politechniki Warszawskiej na stanowisku starszego asystenta. Zajmuje się problematyką systemów pomiarowo-informacyjnych.

TYPOWE ŚRODOWISKO DZIAŁANIA TRANSLATORA

Translator języka ASCAM 131 działa pod kontrolą systemu operacyjnego zapisanego na taśmie dziurkowanej. System ten, złożony z odpowiednich modułów obsługi urządzeń zewnętrznych, umożliwia — przy współudziale programu zarządzającego — sprawne wykonywanie wszystkich czynności związanych z translacją. Wykorzystywane są następujące urządzenia zewnętrzne:

- **drukarka znakowo-mozaikowa z klawiaturą,** przeznaczoną do konwersacji z systemem, redagowania tekstu programu oraz wydruku wyników działania programów użytkowych
- **czytnik taśmy papierowej,** służący do wprowadzania zawartości taśm programu źródłowego lub ładowania zawartości taśm kodu maszynowego programu do pamięci operacyjnej
- **dziurkarka taśmy papierowej,** służąca do wyprowadzania taśm kodu maszynowego lub taśm programów źródłowych.

Typową konfigurację przedstawiono na rysunku 1.



Rys. 1. Konfiguracja sprzętu, zapewniająca działanie translatora języków adresów symbolicznych ASCAM 131 (wszystkie bloki funkcjonalne CAMAC produkcji ZAE ZZUJ POLON, Warszawa)

CHARAKTERYSTYKA PROCESORA AUTONOMICZNEGO 131 CAMAC

Procesor autonomiczny typu 131 jest programowalnym sterownikiem kasety systemu pomiarowo-sterującego CAMAC. Stanowi jednostkę centralną modularnego minikomputera 24-bitowego, zawierającą generator rozkazów CAMAC. Układ tego generatora spełnia wszystkie wymagania normy [4], dotyczące obsługi magistrali CAMAC. Pracą generatora steruje jednostka centralna, realizująca program ulokowany w pamięci zewnętrznej.

Jednostka centralna jest typową maszyną jednoadresową o przetwarzaniu równoległym, wykonaną w technologii cyfrowych układów scalonych o małym i średnim stopniu scalenia. Lista rozkazów jednostki obejmuje ogółem 183 pozycje, a wśród nich: rozkazy arytmetyczno-logiczne, rozkazy skoków warunkowych i bezwarunkowych, rozkazy operacji na rejestrach procesora, rozkazy obsługi zgłoszeń, rozkazy generacji funkcji CAMAC itp. Programy wykonywane przez jednostkę centralną mogą być umieszczone w modułach pamięci stałej lub pamięci o dostępie swobodnym.

Wśród modułów pamięci o dostępie swobodnym wyróżnia się (uwzględniając specyficzną konstrukcję jednostki centralnej) moduły pamięci operacyjnej oraz moduły pamięci danych. W obrębie modułu pamięci operacyjnej możliwa jest realizacja pełnej listy rozkazów jednostki, zaś w obrębie modułu pamięci danych możliwe jest wykonywanie tylko części rozkazów jednostki, a w szczególności — rozkazów zapisu, odczytu i modyfikacji zawartości pamięci. Dla procesora autonomicznego 131 maksymalna pojemność pamięci operacyjnej wynosi 4 K, zaś pamięci danych — 16 K słów 24-bitowych.

SYNTETYCZNY OPIS SKŁADNI JĘZYKA ASCAM 131

- Program źródłowy w języku adresów symbolicznych ASCAM 131 składa się z ciągu wierszy, zakończonego wierszem końca programu.
- Wiersz programu stanowi ciąg znaków alfabetu języka, zakończony znakiem końca wiersza *(CR)*.
- Wiersz programu zaczynający się od znaku # jest wierszem końca programu.

Każdy wiersz programu podzielony jest umownie na cztery części (pola), oddzielone określonymi znakami końcowymi. Kolejność pól jest następująca: pole nazw, pole kodu rozkazu, pole argumentu, pole komentarza. Wykorzystanie niektórych pól (np. pola komentarza) jest nieobowiązkowe. W szczególności wiersz programu nie zawierający żadnego z pól traktowany jest jako wiersz pusty o zerowej wartości semantycznej.

- Pole nazw przeznaczone jest na: przypisanie nazwie występującej w tym polu aktualnej wartości licznika przydziału miejsc pamięci lub wartości wyrażenia arytmetycznego, albo też przypisanie licznikowi przydziału miejsc pamięci wartości wyrażenia arytmetycznego, zależnie od rodzaju znaku końcowego nazwy (: lub =).
- Pole kodu rozkazu przeznaczone jest na skrót mnemoniczny rozkazu lub dyrektywy języka ASCAM. Wszystkie rozkazy mają skróty mnemoniczne będące nazwami o długości nie przekraczającej czterech znaków.
- Pole argumentu przeznaczone jest na wyrażenie arytmetyczne, którego wartość stanowi argument rozkazu o kodzie mnemonicznym zawartym w polu kodu rozkazu.
- Pole komentarza jest ciągiem znaków alfabetu języka, zaczynającym się znakiem średnika, a kończącym znakiem *CR*.

Wyrażenia arytmetyczne tworzone są z określonych klas, wyrażeń oraz z operatorów. Klasą wyrażenia arytmetycznego może być: liczba, nazwa, licznik przydziału miejsc pamięci. Operatorami są symbole operacji arytmetycznych, wykonywanych na odpowiednich obiektach języka. W języku ASCAM 131 wyróżnia się: operatory unitarne +, —, operator dodawania arytmetycznego +, operator odejmowania arytmetycznego —. Alfabet języka ASCAM 131 jest podzbiorem znaków Międzynarodowego Alfabetu nr 5 (ISO-7). Nazwę tworzy ciąg liter i cyfr alfabetu języka (o dowolnej długości), zaczynający się od litery i zakończony jednym ze znaków końcowych. Znakami końcowymi są następujące znaki =, :, odstęp, *CR*, +, —, ;. Liczby definiowane są jako ciągi cyfr alfabetu języka, ograniczone znakiem końcowym. Wyróżnia się liczby stałoprzecinkowe w zapisie dziesiętnym i ósemkowym.

TRANSLATOR JĘZYKA ASCAM 131

Translator języka ASCAM 131 umożliwia przetłumaczenie programu źródłowego, napisanego według reguł tego języka, na kod maszynowy procesora 131. Translacja następuje w dwóch przebiegach.

W czasie pierwszego przebiegu translatora, w odpowiednich obszarach pamięci operacyjnej tworzone są tablice:

- nazw adresów symbolicznych wraz z odpowiadającymi im aktualnymi wartościami licznika przydziału miejsc w pamięci
- nazw określonych wraz z obszarem zarezerwowanym na wartości, które tym nazwom zostaną przypisane w drugim przebiegu.

W trakcie pierwszego przebiegu translator prowadzi także pełną analizę syntaktyczną wierszy programu oraz kontrolę miejsc w pamięci operacyjnej, przeznaczonych na tablice nazw i określeń.

Właściwe tłumaczenie programu źródłowego następuje podczas drugiego przebiegu translatora. Podczas tego przebiegu obliczane są wartości wyrażeń arytmetycznych, a tablica nazw określonych zostaje uzupełniana wartościami wyrażeń arytmetycznych, przypisanych tym nazwom. Jednocześnie kody mnemoniczne rozkazów i dyrektyw zostają przekształcone do postaci kodu maszynowego, natomiast adresy symboliczne i określenia zostają przetworzone przy wykorzystaniu tablic, powstałych w pierwszym i częściowo w drugim przebiegu translatora. W czasie trwania drugiego przebiegu prowadzona jest ponowna analiza syntaktyczna — oraz dodatkowo — semantyczna programu źródłowego, a ewentualne błędy rejestrowane są w tablicy błędów. Efektem końcowym drugiego przebiegu jest taśma kodu maszynowego oraz wydruk tekstu programu. Na rysunku 2 podano przykładowy wydruk typowego programu źródłowego w języku ASCAM 131.

```
                        ; PROGRAM WYPROWADZANIA ODCINKA PUSTEJ TASMY
                        ; NA  DZIURKARCE  TASMY  DT - 105 S

                  PERF=4
                  BLAN:  EX     N(PERF) A(2) F(26)      ; INTERFEJS DZIURKARKI
000040010 000032202     LLA     0177                    ; ODBLOKUJ LAM2
000040011 000725777     LDA     0                       ; LADUJ REJESTR A
000040012 000440000     EXC     N(PERF) A(2) F(9)       ; ZERUJ AKUMULATOR
000040013 000110202     BRNC    -1                      ; CZY DZIURKARKA GOTOWA ?
000040014 000724022     EX      N(PERF) A(2) F(16)      ; NIE
000040015 000020202     BRLA    -4                      ; DZIURKUJ 1 RZADEK
000040016 000724125     HTNC    0                       ; CZY KONIEC DZIURKOWANIA ?
000040017 000764021                                     ; TAK
S                 KONIEC                                ; BLAD SYNTAKTYCZNY
                  $       KONIEC PROGRAMU BLAN
*SE*
```

Rys. 2. Wydruk po translacji przykładowego programu wyprowadzania odcinka pustej taśmy na dziurkarce; w pierwszych trzech kolumnach znajdują się: kod błędu, aktualna wartość licznika przydziału miejsc pamięci i ósemkowy kod rozkazu

SYSTEM OPERACYJNY

System operacyjny składa się z szeregu modułów funkcjonalnych, sterowanych przez moduł zarządzający. Podstawowymi zadaniami systemu są:

- sterowanie translacją programów źródłowych
- sygnalizacja błędów i przekroczeń
- sterowanie urządzeniami zewnętrznymi
- wykonywanie czynności redakcyjnych.

Ze względu na małą pojemność pamięci operacyjnej funkcje systemu zostały podporządkowane głównie zadaniu efektywnego przeprowadzania translacji. Możliwości redak-

cyjne systemu zostały ograniczone do niezbędnego minimum. Dla celów redakcyjnych opracowano odrębne oprogramowanie [2].

System operacyjny może znajdować się w jednym z dwóch stanów: w stanie pracy konwersacyjnej lub w stanie wykonywania zleceń. W pierwszym — operator ma możliwość wprowadzania i modyfikowania zleceń systemowych; w drugim zaś — system operacyjny realizuje zlecenia operatorskie. Po zakończeniu wykonywania zleceń system przechodzi automatycznie do stanu konwersacji. Do tego stanu system przechodzi także we wszystkich przypadkach, gdy niemożliwa jest realizacja wydanych zleceń lub wystąpiły zdarzenia zakłócające tok wykonywania zleceń (np. błąd parzystości przy wczytywaniu taśmy dziurkowanej).

Zlecenia systemu podzielono na zlecenia sterujące translacją oraz określające zakres czynności redakcyjnych systemu. Możliwa jest jednoczesna realizacja zleceń obu grup, a interpretacja niektórych zleceń zależy od kontekstu. System został tak skonstruowany, że w czasie pracy konwersacyjnej niemożliwe jest wydanie sprzecznych zleceń (np. zlecenia wykonania jednocześnie pierwszego i drugiego przebiegu), co znacznie podnosi wygodę obsługi systemu.

Przykłady zleceń systemowych:

- #1 #L #P# — zlecenie wykonania pierwszego przebiegu translacji programu źródłowego, z jednoczesnym wyprowadzeniem tekstu programu źródłowego wraz z listą błędów syntaktycznych, na monitor systemu oraz ze skopiowaniem taśmy źródłowej programu na dziurkarce taśmy.
- #2 #P# — zlecenie wykonania drugiego przebiegu translacji programu źródłowego, z jednoczesnym wyprowadzeniem taśmy z kodem maszynowym.

* * *

Omówiony system operacyjny oraz translator języka adresów symbolicznych ASCAM 131 zajmują obszar ok. 0,9 K słów pamięci operacyjnej. Możliwa jest więc translacja programów źródłowych zawierających nie więcej niż 50 nazw i określeń. Stanowi to dość istotną wadę systemu, wynikającą wszakże z niewielkiej pojemności pamięci. Dopuszczalność stosowania w języku adresów symbolicznych ASCAM 131 bezwzględnego adresowania pamięci umożliwia — jak wskazuje doświadczenie autora — dość swobodne pisanie programów w obrębie całej dysponowanej pamięci operacyjnej.

System operacyjny został tak zaprojektowany, iż możliwa jest jego łatwa realizacja także w zestawach autonomicznych CAMAC, wyposażonych w pamięć operacyjną o pojemności do 4 K słów.

Czytelnicy zainteresowani użytkowaniem programu proszeni są o skontaktowanie się z autorem.

LITERATURA

[1] Bartyś M.: ASCAM 131. Język adresów symbolicznych na procesor autonomiczny 131 oraz pamięć operacyjną 1 K słów 24 bitowych. Instytut Automatyki Przemysłowej Politechniki Warszawskiej, 1979, opracowanie wewnętrzne, A-188/II

[2] Bartyś M.: EDCAM 131. Interakcyjny system programowy, do zadań przygotowania dziurkowanej taśmy źródłowej i edycji programów, przeznaczony do systemu autonomicznego CAMAC. Instytut Automatyki Przemysłowej Politechniki Warszawskiej, 1979, opracowanie wewnętrzne, A-188/II

[3] Gecow A.: Język i assembler CAMASS B 212 na procesor autonomiczny typ 131. Instytut Badań Jądrowych, Świerk, 1977

[4] PN-72/T-06530 CAMAC. Blokowy system oprzyrządowania elektronicznego do pomiarów automatycznych i sterowania. Konstrukcja i organizacja logiczna

[5] Procesor autonomiczny typ 131. Instrukcja obsługi 155 — IO. ZZUJ POLON, Zakład Aparatury Elektronicznej, Warszawa.

MIĘDZYNARODOWE KURSY KOMPUTEROWE W BUDAPESZCIE

Międzynarodowe Centrum Szkolenia i Informacji Komputerowej SZÁMOK

Adres: H-1502 Budapest 112, skrytka pocztowa 146, Węgry

Międzynarodowe kursy komputerowe, doskonalenia zawodowego na rok 1982

1.	Programowanie konkurencyjne	w jęz. angielskim	1-5 marca 1982 r.
2.	Nowoczesne metody planowania w opracowaniu danych	w jęz. angielskim	8-12 marca 1982 r.
3.	Efektywne strukturalne planowanie w COBOL	w jęz. angielskim	15-19 marca 1982 r.
4.	Wiarygodność systemów komputerowych	w jęz. angielskim	22-26 marca 1982 r.
5.	Praktyka planowania bazy danych	w jęz. angielskim	29 marca – 2 kwietnia 1982 r.
6.	Systemy kierownicze	w jęz. angielskim	5-9 kwietnia 1982 r.
7.	Strukturalne planowanie programów metodą Warnier (praktyka)	w jęz. angielskim	19-30 kwietnia 1982 r.
8.	Zastosowanie symulacji digitalnej	w jęz. angielskim	19-23 kwietnia 1982 r.
9.	Kierownictwo Projekt	w jęz. angielskim	26-30 kwietnia 1982 r.
10.	Strukturalne planowanie programów metodą Jackson (praktyka)	w jęz. angielskim	3-14 maja 1982 r.
11.	Strukturalne projektowanie systemów	w jęz. angielskim	3-5 maja 1982 r.
12.	Praktyka strukturalnego projektowania systemów	w jęz. angielskim	6-8 maja 1982 r.
13.	Kierownictwo zespołami roboczymi planowania	w jęz. angielskim	10-14 maja 1982 r.
14.	Efektywne rozwiązanie problemu za pomocą PROLOG-u	w jęz. angielskim	17-21 maja 1982 r.
15.	Projektowanie wiarygodnych	w jęz. angielskim	17-21 maja 1982 r.
16.	Kontrola systemów komputerowych	w jęz. angielskim	24-28 maja 1982 r.
17.	Zastosowanie grafiki komputerowej w projektowaniu inżynieryjnym	w jęz. angielskim	31 maja – 4 czerwca 1982 r.
18.	Technika stosowania komputeru megamini R-11	w jęz. rosyjskim	7-11 czerwca 1982 r.
19.	Automatyzacja działalności bibliotecznych oraz usług informacyjnych	w jęz. angielskim	6 września – 1 października 1982 r.
20.	Komputerowa statystyka zatrudnionych w przedsiębiorstwie	w jęz. rosyjskim	4-8 października 1982 r.
21.	Analiza systemów kierowania produkcją gospodarki zapasami	w jęz. angielskim	11-15 października 1982 r.
22.	Portabilitacja software	w jęz. rosyjskim	18-22 października 1982 r.
23.	Język programowania ADA	w jęz. angielskim	25-29 października 1982 r.
24.	System Network VIDEOTON	w jęz. angielskim	1-5 listopada 1982 r.
25.	Zastosowanie planowania siatkowego	w jęz. angielskim	8-12 listopada 1982 r.
26.	Analiza efektywności systemów komputerowych	w jęz. rosyjskim	15-19 listopada 1982 r.

Nasz hotel przez cały rok przyjmuje obok słuchaczy kursów również gości zagranicznych.
Wynajmujemy sale dla międzynarodowych konferencji.

SZÁMOK

HOTEL SZÁMOK Budapest, XI. Szakasits Árpád u. 68.
Telefon: 669-377, Telex: 22-4499

EO/670/K/81

Podręczna biblioteczka programisty

Komputer stał się dzisiaj jednym z narzędzi o zasadniczym znaczeniu dla rozwiązywania praktycznych problemów naukowych, inżynierskich i ekonomicznych. Stąd też wzięła się koncepcja otwarcia nowej rubryki w INFORMATYCE pn. ALGORYTMY, która stanowić będzie podręczną biblioteczkę programisty. Będziemy w niej przedstawiać interesujące i ważne w praktyce informatyka algorytmy.

Jakimi algorytmami chcielibyśmy się zajmować? Przede wszystkim nie numerycznymi, gdyż numeryczne można znaleźć w dostępnej literaturze; a ponadto stanowią one jedną z podstawowych części firmowego oprogramowania maszyny cyfrowej. Główny nacisk chcielibyśmy położyć na algorytmy związane ze strukturami danych. Można je podzielić na trzy zasadnicze grupy:

- narzędzia do tworzenia i obsługi baz danych
- narzędzia do generowania danych dla symulacji cyfrowej
- narzędzia do testowania programów z obu powyższych grup.

Ponadto chcielibyśmy przedstawić procedury użytkowe, a zwłaszcza tzw. „chytre algorytmy", które nie są na ogół prezentowane w literaturze w postaci konkretnych programów, lecz stanowią jedynie matematyczne podstawy umożliwiające ich napisanie, podczas gdy informatyk-praktyk potrzebuje z reguły po prostu gotowej procedury.

Jakie są motywacje dla tego rodzaju publikacji? Rozpatrzmy dość typowe zadanie stawiane przed informatykiem. Załóżmy, że pan X ma do napisania program obsługi księgowości w przedsiębiorstwie. W trakcie projektowania systemu zauważy on, że z jego zadania dają się wyodrębnić pewne fragmenty stosunkowo niezależne od koncepcji całości. Będą to na przykład: sortowanie (np. alfabetyczne według nazwisk personelu, według wysokości płacy, działów zakładu pracy itp.), wyszukiwanie informacji (np. wyszukanie wszystkich pracowników, których zarobki mieszczą się w przedziale 100—120 tys. złotych rocznie, czy wszystkich osób mieszkających poza miejscowością zatrudnienia) oraz wiele innych.

Pan X, chcąc osiągnąć dużą efektywność systemu, musi poświęcić dużą część swojego czasu na staranne napisanie procedur obsługujących wyżej wymienione zagadnienia, gdyż wpływają one w sposób zasadniczy na czas pracy całego programu. Dla uzasadnienia tej zależności porównajmy czasy działania dwóch różnych algorytmów sortowania przetestowanych na maszynie CDC 6400 (czas mierzony w sekundach). Algorytm prosty to tzw. sortowanie bąbelkowe (ang. *bubble sort*), „chytry" to sortowanie szybkie (ang. *quick sort*). Przyjrzyjmy się poniższej tabelce:

Liczba pracowników	500	1000	1400
Czas sortowania bąbelkowego	5,6	22,4	358,4
Czas sortowania szybkiego	0,075	0,166	0.8

Jak widać stosowanie „chytrego" sortowania pozwala na uzyskanie sensownego czasu pracy systemu, co implikuje praktyczną konieczność używania dobrych procedur. Sortowanie nie jest oczywiście przykładem odosobnionym. Podobne zależności czasowe można podać dla innych typowych problemów.

Jak z pewnością Czytelnik w swojej praktyce zauważył, na ogół nie udaje się za pierwszym razem napisać poprawnie działającego programu lub procedury o pewnym stopniu komplikacji. Gdyby pan X mógł po prostu przepisać sprawdzony i przetestowany algorytm, uniknąłby takich kroków, jak poprawianie błędów czy testowanie fragmentów nie związanych z istotą jego sytemu. Pozwoliłoby mu to na głębszą analizę specyfiki rozwiązywanego przez niego zadania.

Poniżej chcielibyśmy zaprezentować pierwsze z proponowanych algorytmów. W związku z tym parę słów o przyjętej przez nas konwencji. Do zapisywania algorytmów używać będziemy języka programowania ALGOL 60 bez rekurencji, wzbogaconego o standardowy typ zmiennych **character** (znakowy). Ponieważ instrukcje wejścia/wyjścia zależą od implementacji, będziemy stosować zapis zrozumiały dla każdego informatyka. Wybór tej formy notacji był podyktowany, po pierwsze — czytelnością ALGOLU, a po drugie — łatwością tłumaczenia na FORTRAN, PL/I, COBOL i inne popularne języki programowania. W przypadkach przedstawiania algorytmów opartych na literaturze, podawać będziemy odsyłacz do zródeł i ewentualnie nazwę algorytmu. Ponieważ najbardziej rozpowszechniona jest terminologia anglojęzyczna, obok polskich nazw podawać będziemy oryginalne angielskie.

Rozwiązanie pierwszego z poniższych problemów, bardzo zresztą proste, oparte jest na pomyśle, którego nie spotkaliśmy w żadnym znanym nam i stosowanym w kraju systemie informatycznym (z wyjątkiem obsługi centralnej konsoli operatorskiej maszyny CDC 6400). Natomiast drugi jest problemem, z którym chyba każdy programista spotkał się w swojej praktyce, lecz rozwiązanie jest na tyle oryginalne, że uznaliśmy je za warte zaprezentowania.

1.

Opis problemu

Załóżmy, że mamy dany pewien system konwersacyjny, w którym użytkownik może używać słów tylko z pewnego słownika. Jako przykład niech służy baza danych biblioteki, w której Czytelnik chcąc uzyskać informacje o pewnej pozycji — musi podać imię i nazwisko autora oraz jej tytuł. W przypadku prostego rozwiązania musi napisać na klawiaturze wszystkie znaki z powyższego identyfikatora, chociaż w rzeczywistości do rozpoznania niezbędne są tylko niektóre. Na przykład — jeśli katalog składa się z następujących pozycji:

1. Mickiewicz Adam Pan Tadeusz
2. Sienkiewicz Henryk Pan Wołodyjowski
3. Sienkiewicz Henryk Quo vadis
4. Szołochow Michał Cichy Don,

to do rozpoznania wystarczą tylko zaznaczone litery.

Opis procedury PUSZKA

Poniżej przedstawiona funkcja uzupełniać będzie napis oczekując jedynie na znaczące symbole. Wartością funkcji jest adres identyfikatora w słowniku. Przyjęliśmy następujące znaczenie parametrów:

słownik — tablica dwuwymiarowa zawierająca identyfikatory uporządkowane leksykograficznie i pamiętane kolumnami, np. słownik zawierający słowa TATA, MAMA oraz TARA, jest następujący:

M	T	T
A	A	A
M	R	T
A	A	A

dlh — maksymalna długość identyfikatora
ilh — liczba identyfikatorów w słowniku.

Algorytm

```
integer procedure PUSZKA (słownik, dlh, ilh);
comment Procedura Uwzględniająca Symbole Znaczące
w Kluczach Alfabetycznych;
character array słownik; integer dlh, ilh;
begin character ch; integer i, j, nz;
i: = 1; j: = ilh; nz: = 0;
drukuj: nz: = nz + 1;
  if nz > dlh then goto znalezione;
if słownik [nz, i] ≠ słownik [nz, j] then goto czytaj;
  write (słownik [nz, i]); goto drukuj;
czytaj: read (ch);
while słownik [nz, i] < ch do
  if i = ilh then goto błąd else i: = i + 1;
  while słownik [nz, j] > ch do
  if j = I then goto błąd else j: = j—1;
  if j ≥ i then goto drukuj;
błąd: PUSZKA: = 0; goto wyjście;
znalezione: PUSZKA: = i;
wyjście:
end;
```

Oczywiście zakładamy, że czytanie i pisanie odbywa się na tym samym urządzeniu i bez zmiany linii.

Zastosowanie powyższego algorytmu jest chyba najpraktyczniejszą metodą pogodzenia sprzeczności pomiędzy zrozumiałością haseł dla użytkownika, a czasem ich pisania na tym samym urządzeniu i bez zmiany linii.
dzo kłopotliwej analizy syntaktycznej wprowadzanych haseł, co może istotnie ułatwić pisanie programów obsługiwanych interakcyjnie. Algorytm ten jest praktycznie stosowany w SPIK IBIB PAN na minikomputerze MERA 400 w pokazowym systemie medycznym.

2.

Opis problemu

Dane są trzy liczby naturalne określające dzień, miesiąc oraz rok. Należy znaleźć dzień tygodnia odpowiadający tej dacie.

Opis algorytmu (por. Lamotier J. P.: Exercises de programmation en FORTRAN IV, DUNOD, Paris 1977, § 8.1.)

1. Obliczamy N:

Jeśli miesiącem jest styczeń lub luty roku przestępnego, to definiujemy N = 3;
Jeśli miesiącem jest styczeń lub luty roku zwykłego, to N = 4; w pozostałych przypadkach N = 2.
Aby stwierdzić, czy dany rok jest przestępny, rozkładamy liczbę określającą go na dwie części A1 i A2:
A1 zawiera dwie bardziej znaczące cyfry,
A2 zawiera dwie mniej znaczące cyfry,
np. dla roku 1981 A1 = 19, A2 = 81.
Jeśli A2 = 0 i A1 dzieli się przez 4, to rok jest przestępny, jeśli A2 ≠ 0 i A2 dzieli się przez 4, to rok jest przestępny, w pozostałych przypadkach jest on rokiem zwykłym.

2. Obliczamy „kod dnia" stosując formułę:

C = entier (365.25 * A2) + entier (30.56 * miesiąc) + dzień + N.

3. Reszta z dzielenia C przez 7 określa numer dnia tygodnia (poniedziałek = 0, wtorek = 1 itd.).

Opis procedury Day of week

Procedura funkcyjna Day of week ma trzy parametry, będące liczbami naturalnymi:

- day — określa dzień miesiąca
- month — określa numer miesiąca
- year — określa numer roku.

Wynikiem funkcji jest numer dnia w tygodniu, licząc od poniedziałku (poniedziałek jest pierwszym dniem, wtorek drugim itd.).

```
integer procedure Day of week (day, month, year);
integer day, month, year;
comment oblicza na podstawie daty numer dnia w tygodniu;
begin integer A1, A2, N;
A1: = year div 100; A2: = year mod 100;
if month > 2 then N: = 2
else if (A2 = 0&A1 mod 4 = 0) ∨ (A2 ≠ 0&A2 mod 4 = 0)then N: = 3
  else N: = 4;
Day of week: = 1 + ((entier(1.25 * A2) + entier(2,56 * month) + N +
+ day) mod 7)
  end Day of week;
```

* * *

Na zakończenie pozostaje nam wyrazić nadzieję, że proponowany cykl spotka się z przychylnym przyjęciem Czytelników i choćby w minimalnym stopniu zapełni dotkliwą lukę w polskiej literaturze fachowej na ten temat. Oczekujemy uwag krytycznych i propozycji tematycznych, także ciekawych rozwiązań praktycznych problemów.

Andrzej SZAŁAS
Instytut Matematyczny PAN
Zbigniew SWIRSKI
Instytut Biocybernetyki i Inżynierii Biomedycznej PAN

Krajowy sprzęt dla informatyki na 53 MTP

Oferta polskiego przemysłu komputerowego, przedstawiona w roku sześćdziesięciolecia Targów Poznańskich (14—23 czerwca br.) była nie tylko zaskakująco szeroka, ale i bardzo ciekawa. Aczkolwiek oficjalnie przyjęto zasadę pokazywania na Targach urządzeń dostępnych w handlu, można się było łatwo zorientować, że wiele modeli nie weszło jeszcze do produkcji. Sytuację dodatkowo komplikuje fakt, że proponowane urządzenia są bardzo importochłonne. Nic więc dziwnego, że wiele urządzeń opracowano z myślą o konkretnym, zachodnim odbiorcy i ich sprzedaż będzie musiała sfinansować zakup materiałów i podzespołów, zwłaszcza układów scalonych wielkiej skali integracji. Na rynku krajowym popyt na nowoczesny sprzęt komputerowy znacznie przewyższał dotąd jego podaż a obecny kryzys gospodarczy spowoduje, że w praktyce większość urządzeń będzie dla nas w ogóle niedostępna. Poniżej omówię tylko te wyroby, które posiadają istotne cechy nowości i mogą zainteresować szersze grono Czytelników [1]).

System mikrokomputerowy MERA-60

Centrum Naukowo-Produkcyjne Systemów Sterowania w Katowicach

Rodzina mikrokomputerów MERA-60 cechuje się modularnością architektury i oprogramowania. Obecnie podano oficjalnie, że akceptuje ona oprogramowanie maszyny cyfrowej PDP-11/03. Podstawowe dane techniczne systemu: długość słowa maszynowego — 16 bitów, liczba podstawowych rozkazów — 64, pamięć operacyjna od 4 K do 28 K słów, wektoryzowany system przerwań i możliwość bezpośredniego dostępu do pamięci.

Przedstawiono dwa nowe modele z tej rodziny: MERA 60/40 oraz MERA 60/20. Pierwszy z nich jest przeznaczony do sterowania procesami technologicznymi w różnych dziedzinach przemysłu, w systemach kontrolno-pomiarowych itp., drugi natomiast przeznaczony jest do obliczeń inżynierskich, zbierania i przetwarzania danych itp. Oba modele mogą wchodzić w skład sieci komputerowej i w obu zastosowano procesor o zwiększonej o 15% szybkości wykonywania podstawowych operacji (a więc 120 ÷ 280 000 operacji/s) oraz wprowadzono sprzętowy zmienny przecinek.

Całość oprogramowania znajduje się na dyskach elastycznych. Minimalna konfiguracja tych mikrokomputerów obejmuje: zmiennoprzecinkowy procesor (M2), pamięć operacyjną, stację dysków elastycznych, pulpit operatora (monitor ekranowy lub drukarka mozaikowa z klawiaturą) oraz moduł pamięci stałej.

Podstawowe oprogramowanie obejmuje: system operacyjny czasu rzeczywistego RT-60, zawierający monitor jedno- lub dwuzadaniowy, programy usługowe oraz translatory — MACROASSEMBLER, FORTRAN, BASIC.

System wspomagania MERA 80-15

Centrum Naukowo-Produkcyjne Systemów Sterowania w Katowicach

System wspomagania MERA 80-15 zawiera środki programowe i sprzętowe umożliwiające przygotowanie programów dla mikrokomputerów opartych na mikroprocesorze INTEL 8080, a zwłaszcza dla programowanego sterownika sekwencyjnego MERA 80-16. Opisywany system wspomagania jest zbudowany z modułów Mikrokomputerowego Systemu Automatyzacji Pomiarów i Sterowania MERA-80.

Podstawowe parametry systemu są następujące: typ mikroprocesora 8080 A (lub K580NK80), 8 poziomów przerwań priorytetowych maskowalnych, pojemność pamięci EPROM 4 KB, pojemość pamięci RAM 32 KB, kanał bezpośredniego dostępu do pamięci, wspólny system adresacji we-wy, interfejsy — BISS, V-24.

Urządzenia peryferyjne obejmują: monitor ekranowy MERA 7952, pamięć na dyskach elastycznych PL X X 45D, drukarkę znakowo-mozaikową DZM-180, stację przygotowania taśmy papierowej SPTP-3, programator pamięci EPROM typu PRISS-10.

Oprogramowanie podstawowe obejmuje: monitor BOSS-80, makroasembler BOMAS 80/85 V.1.0, edytor tekstów EDIT 80, interpreter języka MIKROBASIC. Zaawansowane oprogramowanie sytemowe stanowi system operacyjny BOSMAN.

Systemy komputerowe przeznaczone do automatyzacji procesów przemysłowych, oparte o system CAMAC

• CAMAC jako kanał przemysłowy dla mikrokomputera MERA-60. Sprzężenie zestawu CAMAC z MERA-60 zostało zrealizowane przez wyspecjalizowane bloki. Od strony kasety CAMAC był to sterownik typu 106A, a od strony mikrokomputera — adapter typu MCM-60.

• Autonomiczny zestaw CAMAC przeznaczony do wspomagania operatora bloku energetycznego

Zakład Aparatury Elektronicznej POLON

• Zestaw CAMAC do optymalizacji wydobycia siarki. Najważniejsze bloki wchodzące w skład tego zestawu to: procesor autonomiczny typu 131, pamięć ferrytowa typu 201, zegar czasu rzeczywistego typu 732, przetwornik A/C integracyjny typu 701 z multiplekserem analogowym.

Przedsiębiorstwo Automatyki Przemysłowej MERA-PNEFAL

Minisystem MERA 2500

Zakłady Mechaniczno-Precyzyjne MERA-BŁONIE

System ten reprezentował sprzęt dla EPD i budził duże zainteresowanie ze względu na niewielkie wymiary i oferowane możliwości, przydatne zwłaszcza dla rozliczeń finansowych, sprawozdawczości itp. Został on skonstruowany w oparciu o mikroprocesor typu 8008 i pamięć operacyjną półprzewodnikową (8 KB EPROM, 8 KB RAM). Urządzenia zewnętrzne tego systemu to: drukarka znakowo-mozaikowa DZM 180L z klawiaturą oraz pamięć na dysku elastycznym. Podstawę oprogramowania stanowi problemowo zorientowany język LOG.

Urządzenia do taśmy dziurkowanej

Zakłady Urządzeń Komputerowych MERA-ELZAB, Zabrze

• System przygotowania danych na taśmie dziurkowanej SPTP-100. Wykorzystuje on dziurkarkę taśmy papierowej DT-105S oraz czytnik taśmy dziurkowanej typu ADDMASTER, model 601-2 (USA). Jest sterowany mikroprocesorem 8085 i pozwala na przygotowanie, kopiowanie i wyprowadzanie danych na taśmie dziurkowanej wraz z ewentualną konwersją kodów. Urządzenie współpracuje z dowolną klawiaturą. Możliwa jest zarówno praca lokalna, jak i zdalne przyłączenie do systemu poprzez interfejs szeregowy.

• Dziurkarka taśmy typu SRL. Ta nowa dziurkarka taśmy papierowej charakteryzuje się uproszczoną konserwacją i regulacją, cichą pracą i niewielkim ciężarem (ok. 15 kg). Prędkość dziurkowania — 50 znaków/s. Spodziewana jest wersja ze sprawdzaniem poprawności wydziurkowania danych i ewentualną automatyczną korekcją błędów.

[1]) Niniejsze informacje zostały przygotowane na podstawie materiałów udostępnianych podczas MTP przez producentów i wystawców opisywanego sprzętu.

Monitory ekranowe

Przedstawiono nową wersję monitora MERA 7952, oznaczoną symbolem MERA 7952 vgd. Jest to terminal kompatybilny z podobnymi konstrukcjami firmy TTI zbudowany na bazie specjalizowanego mikroprocesora. Klawiatura w układzie maszyny do pisania z wydzieloną częścią numeryczną, pojemność ekranu 1920 znaków (24 wiersze po 80 znaków w wierszu), kod ASCII. Transmisja szeregowa, asynchroniczna od 75 do 9600 bitów/s. Wyposażeniem opcjonalnym jest drukarka znakowa.

Drukarki znakowo-mozaikowe

Zakłady Mechaniczno-Precyzyjne MERA-BŁONIE

Zaprezentowano następujący asortyment drukarek znakowych:

- Typ DZM-180AL: zintegrowany transport papieru, prędkość drukowania 180 znaków/s, gęstość pozioma druku: 10; 12; 16,5 znaków/cal, gęstość pionowa druku: 6 wierszy/cal. Interfejs równoległy LOGABAX lub V-24 (opcja).
- Typ DZM-80, przeznaczony dla terminali. Zintegrowany transport papieru, małe gabaryty i ciężar, spodziewana stosunkowo niska cena. Stała szerokość papieru 21 cm, długość wiersza — 80 znaków przy gęstości poziomej 10 znaków/cal. Pionowa gęstość druku — 6 wierszy/cal. Prędkość drukowania — 100 znaków/s. Interfejs szeregowy V-24/RS 232:110-9600 bodów. Część elektroniczna drukarki składa się z układów scalonych LSI do sterowania głowicą drukującą, generatora znaków i pamięci buforowej. Istnieje możliwość drukowania znaków o podwójnej szerokości oraz znaków pochylonych. Zmiana rodzaju druku jest możliwa podczas pracy urządzenia i następuje po odebraniu przez drukarkę odpowiedniego kodu. Niezależnie od tego, użycie dziewięciopunktowej głowicy drukującej pozwala na druk małych liter.
- Typ D-200 był najnowocześniejszą z pokazanych drukarek mozaikowych. Dzięki sterowaniu za pomocą mikroprocesora (8035) uzyskano zupełnie nowe możliwości pracy, takie jak: autotest, druk znaków normalnych, pochylonych i rozciągniętych, automatyczny wysuw po zakończeniu linii i optymalizację ruchu głowicy drukującej. Głowica ta jest napędzana silnikiem skokowym i może drukować w obu kierunkach, dzięki czemu minimalizuje się ruch jałowy. Prędkość drukowania wynosi 180 znaków/s i przewiduje się jej zwiększenie. Może być stosowany papier z brzeżną perforacją, a także bez perforacji, o szerokości od 4 do 17 cali. Pozioma gęstość druku: 10; 12; 16,5 znaków/cal. Gęstość pionowa druku: 6 i 8 wierszy/cal., długość arkusza: 4, 6, 8, 12 cali. Proponowane są różne interfejsy jako opcje, np. BS 4421, V24/RS 232.

Warto także zwrócić uwagę na to, że sprzęt informatyczny był również wystawiany lub oferowany przez przedsiębiorstwa polonijne. Wytwórnia Sprzętu Elektronicznego i Mechaniki Precyzyjnej DANPOL pokazała drukarki mozaikowe EPSON z rodziny MX. Ciekawym rozwiązaniem drukarki był model MX-80 Type II, która oprócz znaków może drukować punktowo obrazy (480 lub 960 punktów w inii). Przedsiębiorstwo MARCO-ELECTRONIC reklamowało mikroprocesorowy system sterowania sekwencyjnego i czasowego dla procesów technologicznych o nazwie SUM 40 oraz „inteligentny" multiplekser danych, typu MARCO MUX-32. Multiplekser ten pozwala na dołączenie do komputera 32 terminali pracujących w kodzie ASCII (drukarki, monitory, plottery) za pomocą pojedynczego, typowego kanału asynchronicznego, według standardów RS 232C, V 24 i RS 423.

Jacek ZEBROWSKI

zjednoczenie informatyki ZETO

Systemy informatyczne ZETO Poznań

Systemy informatyczne, jakie pragniemy zaprezentować czytelnikom INFORMATYKI, dają się ująć w trzy grupy: systemy narzędziowe, przeznaczone dla ośrodków eksploatujących komputery, systemy ogólnego zastosowania (o charakterze typowo ewidencyjnym) oraz systemy umożliwiające automatyczne ocenianie działalności gospodarczej przedsiębiorstw i zjednoczeń.

SYSTEMY NARZĘDZIOWE

PAkiet TESTowania niezależnych modułów — PATEST

PATEST jest narzędziem przeznaczonym dla ośrodków programujących techniką modularną. Usprawnia on proces opracowywania programów, umożliwiających testowanie pojedynczych modułów, całych ich hierarchii, a także kompletnych programów modularnych pisanych w językach ASSEMBLER, COBOL, PL/I i FORTRAN.

PATEST korzystnie wyróżnia się spośród istniejących opracowań tego typu, gdyż:

- testowane moduły nie wymagają żadnych adaptacji
- umożliwia symulowanie modułów podporządkowanych modułom testowanym
- umożliwia ustawianie stałych adresowych na obszarach parametrów dowolnego modułu hierarchii
- przerwania programowe w modułach testowanych są obsługiwane tak, że możliwe jest kontynuowanie pracy programu
- posługiwanie się Pakietem jest proste i nie wymaga dużego nakładu pracy ze strony użytkownika.

W skład PATEST-u wchodzą dwa typy programów: program generujący programy testujące oraz wygenerowane programy testujące. W związku z tym w pracy Pakietu występują dwa etapy: generowanie programów testujących i następnie — uruchomianie wygenerowanych programów testujących, przy czym te ostatnie składają się z modułów testowanych, symulowanych oraz elementów Pakietu.

PATEST może być stosowany na komputerach Jednolitego Systemu pod nadzorem systemu operacyjnego DOS, z pamięcią operacyjną 128 KB i trzema jednostkami pamięci taśmowej lub dyskowej. Pakiet został wdrożony w kilku ośrodkach ZETO i nie stwarzał żadnych trudności w toku wdrażania i eksploatacji.

Efektem wykorzystania Pakietu jest ułatwienie i skrócenie procesu testowania programów modularnych oraz zapewnienie wyższej jakości programów, w których po takim testowaniu nie powinny już występować błędy w czasie eksploatacji, co z kolei oznacza obniżenie kosztów opracowania i eksploatowania systemu.

PAkiety KONwersji PAKON-DOS i PAKON-OS

Pakiety te interesować mogą ośrodki obliczeniowe eksploatujące komputery Jednolitego Systemu, a zamierzające np. przenosić na ten sprzęt systemy eksploatowane poprzednio na komputerach ODRA 1300 lub odwrotnie. Pakiety PAKON służą do dokonywania konwersji zbiorów i bibliotek programów pisanych w COBOL-u między tymi komputerami w obu kierunkach, zaś PAKON-DOS realizuje również konwersję między komputerami serii MINSK-32 i komputerami JS, a także konwersję zbiorów z taśmy dziurkowanej na standard DOS i wydruk zredagowanej taśmy magnetycznej utworzonej przez EGZEKUTOR, system operacyjny GEORGE-2 lub system operacyjny AFRODYTA (z ZETO Gdynia). Konwersja ta następuje w obu pakietach za pośrednictwem taśmy magnetycznej.

W pakiecie PAKON-OS, ze względu na ogólną tendencję do wprowadzania na komputerach ODRA 1300 zmian technicznych zmierzających do zapisu i odczytu zgodnie z normą ISO, przyjęto, że taśmy magnetyczne komputera ODRA muszą mieć rządek kontroli CRC tworzony zgodnie z tą normą. Zastosowano tu też dynamiczne przydzielanie i zwalnianie pamięci operacyjnej w zależności od wymagań programowych.

Konwersje są realizowane dla zbiorów jedno- i wielokrążkowych, dla zbiorów etykietowanych i nie etykietowanych, prostych i złożonych, o rekordach stałej i zmiennej długości, blokowanych lub nie. Jeżeli struktura rekordów w zbiorze jest skomplikowana i nie pozwala na korzystanie przez program konwersji z parametrów wejściowych, użytkownik może dołączać własny moduł do pakietu, celem dokonania zmiany wszystkich pól w rekordzie — lub tylko pól wybranych — i generowania rekordu wyjściowego w takiej strukturze, że poszczególne jego pola będą funkcją innych pól.

Pakiety PAKON mogą być stosowane na komputerach Jednolitego Systemu pod nadzorem systemu operacyjnego DOS lub OS. Wdrażanie i eksploatowanie tych pakietów nie powoduje większych trudności, a od użytkownika wymaga jedynie dobrej znajomości struktur zbiorów poddawanych konwersji. Pakiety te są w dalszym ciągu rozpowszechniane wśród użytkowników komputerów JS, którzy poprzednio stosowali komputery serii ODRA 1300.

SYSTEMY EWIDENCYJNE DLA JEDNOSTEK GOSPODARCZYCH

Kolejne sytemy, które przedstawiamy, to systemy przeznaczone do stosowania w przedsiębiorstwach i innych jednostkach gospodarczych na potrzeby zarządzania, obejmujące gospodarkę środkami trwałymi, gospodarkę wyrobami gotowymi, techniczne przygotowanie produkcji, gospodarkę materiałową, rozliczenia finansowo-księgowe. Przy tworzeniu systemów gospodarki środkami trwałymi, gospodarki wyrobami gotowymi oraz technicznego przygotowania produkcji korzystano z produktu b. OBRI w postaci systemu generacyjnego STEP.

Spośród 13 pakietów programowych systemu STEP, ze względu na realizowane funkcje, w pełni wykorzystano pakiety BAZA i SEJF, zaś w ograniczonym zakresie — pakiety: RAPORT, WEJŚCIE, SELENA. Natomiast w pakietach BIBLIO, PLASTER i ZLECSTER stwierdzono konieczność wprowadzenia szeregu poprawek przez jednostkę autorską, tj. obecnie CPiZI, które są realizowane.

System Gospodarki Środkami Trwałymi — GST-STEP

Jest to system ewidencjonowania i rozliczania środków trwałych, wartości niematerialnych i prawnych wraz ze sporządzaniem sprawozdań okresowych. Podstawowymi zbiorami są tu: kartoteka środków trwałych, wartości niematerialnych i prawnych oraz karoteka środków trwałych postawionych w stan likwidacji i zlikwidowanych. Dane do Systemu pochodzą z

dokumentów źródłowych, mających postać druków powszechnego użytku, oraz z dokumentu zmian i dokumentu „Fizyczna likwidacja środka trwałego", specjalnie zaprojektowanych (z braku odpowiednich druków znormalizowanych). System dostarcza 14 tabulogramów dających obraz zaszłości w zakresie środków trwałych, zestawienia do sprawozdań oraz miesięczne i roczne rozliczenie amortyzacji. Do realizacji Systemu wykorzystano pakiet generacyjny STEP (DANSTER).

System GST-STEP wdrożono już w Poznańskiej Fabryce Maszyn Żniwnych w Poznaniu, w Śląskich Zakładach Mechaniczno-Optycznych w Katowicach, w zakładach UNITRA-TELETRA w Kołobrzegu i w Zakładach Kineskopów Kolorowych w Piasecznie. Z doświadczeń tych wynika, że warunkiem sprawnego przebiegu wdrożenia Systemu jest stosowanie przez użytkownika klasyfikacji GUS dla identyfikacji środków trwałych oraz poprawność dokumentacji tradycyjnej.

Jako efekty uzyskiwane z eksploatowania Systemu należy wymienić eliminację wielu pracochłonnych czynności w komórkach księgowości oraz prawidłowe gospodarowanie majątkiem przedsiębiorstw. Ponieważ rozwiązanie zastosowane w Systemie ma niewątpliwie charakter uniwersalny, istnieje duża szansa wdrażania go w zakładach stosujących klasyfikację GUS w zakresie środków trwałych.

System Gospodarki Wyrobami Gotowymi — GWG-STEP

Również i ten system został zaprojektowany i oprogramowany w oparciu o pakiet STEP (DANSTER). System obejmuje następujące zagadnienia: ewidencję obrotów wyrobami gotowymi, ewidencję produkcji wyrobów w cenach ewidencyjnych, cenach zbytu i cenach porównywalnych, ewidencję sprzedaży wyrobów oraz rozliczanie spisu z natury.

System może być eksploatowany na komputerach Jednolitego Systemu pod nadzorem systemu operacyjnego DOS, z pamięcią operacyjną 128 KB, trzema jednostkami pamięci dyskowej po 30 MB lub czterema po 7,25 MB, czterema jednostkami pamięci taśmowej, czytnikiem i drukarką wierszową. Możliwe jest też wprowadzanie danych bezpośrednio z taśmy magnetycznej przygotowanej na urządzeniu MERA 9150, a więc z pominięciem kart dziurkowanych. Przyjęte rozwiązanie technologiczne, wynikające również z zastosowania systemu STEP, spowodowało, że wdrażanie systemu GWG-STEP nie jest kłopotliwe. Oczywiście w trakcie wdrażania zespół autorski nie zdołał uniknąć pewnych kłopotów, jednak wynikały one z braku doświadczenia w stosowaniu systemu STEP w innych (poza sterowaniem produkcją) dziedzinach.

System GWG-STEP wdrożono jedynie w Śląskich Zakładach Mechaniczno-Optycznych w Katowicach, niemniej nadaje się on do wykorzystywania we wszystkich przedsiębiorstwach przemysłu maszynowego.

System Technicznego Przygotowania Produkcji TPP-STEP

Oprócz zakładania i utrzymywania kartotek bazy danych system ten dostarcza następujących wydruków użytkowych:

- **w postaci katalogów:** pozycji rodzajowych, stanowisk roboczych, detalooperacji wg numeru operacji w kolejności wzrastającej, detalooperacji w analogicznej kolejności oraz w kolejności malejącej z podaniem informacji dotyczącej stanowisk roboczych, operacji wykonywanych na danym stanowisku, z wariantem rozszerzonym, zawierającym informacje o detalu; poza tym uzyskuje się analityczny katalog zbiorczej materiałochłonności na jednostkę wyrobu oraz na plan wyrobu, a także syntetyczny katalog zbiorczej materiałochłonności na jednostkę wyrobu oraz na plan wyrobu
- **w postaci zestawień:** pracochłonności wyrobu wg kategorii zaszeregowania, planowej pracochłonności wg stanowisk roboczych i komórek organizacyjnych, planowanej i jednostkowej pracochłonności w komórkach organizacyjnych, obciążenia stanowisk i odchyleń w stosunku do planu produkcji.

System opracowano w oparciu o system generacyjny STEP (DANSTER). TPP-STEP jet przeznaczony do eksploatacji na komputerach Jednolitego Systemu pod nadzorem systemu operacyjnego DOS z pamięcią operacyjną 128 KB, trzema jednostkami pamięci dyskowej, pięcioma jednostkami pamięci taśmowej, czytnikiem kart i drukarką wierszową.

System TPP-STEP wdrożono w Poznańskiej Fabryce Maszyn Żniwnych w Poznaniu, w zakładach METALCHEM-Toruń oraz w Fabryce Obrabiarek do Drewna w Bydgoszczy, niemniej jego eksploatacja w tych zakładach jest ograniczona na skutek trudności ogranizacyjnych — kłopotów z uzyskaniem prawidłowej dokumentacji obrazującej strukturę wyrobów, braku prawidłowo wypełnionych kart technologicznych, stosowania nieaktualnych normatywów, braku jednoznacznych identyfikatorów pozycji produkowanych w zakładzie, a czasem też z braku materiałów.

Z doświadczeń uzyskanych przy wdrażaniu systemu TPP-STEP wynika, że nie należy przystępować do tego zadania bez uprzedniego zweryfikowania bazy indeksowej i bazy normatywnej; należy też zastanowić się, czy upowszechnianie systemu TPP powinno wyprzedzać wdrożenie innych systemów, którym TPP narzucać będzie pewne ograniczenia, jak np. długość indeksu. Jednostka zakładania i modyfikowania bazy, stanowiąca element systemu TPP, musi być tworzona dla konkretnego użytkownika; jedynie jednostki pracochłonności i materiałochłonności mogą być ewentualnie traktowane jednolicie, wg zasad przyjętych w systemie TPP-STEP.

Eksploatowanie Systemu nie odbywa się cyklicznie, lecz w zależności od liczby danych zgromadzonych do modyfikacji i od stanu kartoteki (stanu obszarów przepełnień).

Poza korzyściami trudnymi do wymierzenia (zwiększenie dokładności i zakresu informacji, obniżenie pracochłonności przygotowywania informacji), wdrożenie systemu TPP-STEP stwarza użytkownikowi możliwość wdrożenia systemów planowania produkcji i sterowania produkcją oraz możliwość emitowania dokumentacji warsztatowej.

Podsystem rozliczeń Finansowo-Kosztowych UCZElni wyższych — FK-UCZE

Podstawową funkcją Podsystemu jest zapewnienie służbom kwestur informacji o obrotach i aktualnych saldach na wszystkich kontach syntetycznych i analitycznych, dostarczanie danych do sprawozdawczości w zakresie kosztów uczelni oraz w zakresie nie rozliczonych rozrachunków z kontrahentami. W tym celu Podsystem tworzy:

- kartotekę INDEKS KONT — na dysku magnetycznym (z aktualizacją)
- zbiór dokumentów finansowych — na taśmie magnetycznej
- KARTOTEKĘ FINANSOWĄ w układzie narastającym — na taśmie magnetycznej (z aktualizacją)
- miesięczny zbiór do wydruku tabulogramów wynikowych
- zbiór nie rozliczonych rozrachunków z kontrahentami — na taśmie magnetycznej (z możliwością wydruku).

Podsystem eksploatowany jest na komputerach Jednolitego Systemu pod nadzorem systemu operacyjnego DOS i wymaga czterech jednostek pamięci taśmowej oraz trzech jednostek pamięci dyskowej po 7,25 MB. W razie stosowania dysków o pojemności po 30 MB zachodzi konieczność odpowiedniej adaptacji Systemu.

Wdrożenie Systemu w Uniwersytecie im. A. Mickiewicza w Poznaniu, Akademiach Rolniczych w Poznaniu i Szczecinie, Wyższej Szkole Morskiej w Szczecinie, Politechnice Szczecińskiej, Uniwersytecie im. M. Kopernika w Toruniu potwierdza, że w porównaniu z tradycyjną metodą prowadzenia ewidencji następuje rozszerzenie zakresu informacji, unowocześnienie warunków pracy i stworzenie warunków do podnoszenia kwalifikacji pracowników kwestur.

W obecnej swej postaci system FK-UCZE może być użytkowany w każdej uczelni wyższej w kraju.

System Informatyczny Gospodarki MATeriałowej — SIGMAT-R

Przedmiotem Systemu jest ewidencjonowanie stanów i obrotów materiałami i przedmiotami nietrwałymi oraz analizowanie zapasów i sporządzanie sprawozdawczości GUS. System zapewnia służbom zaopatrzenia i księgowości informacje o aktualnym stanie i obrotach w odniesieniu do wszystkich materiałów i przedmiotów nietrwałych oraz dostarcza danych do sprawozdawczości zewnętrznej i wewnętrznej z zakresu zaopatrzenia oraz kształtowania się zużycia materiałów i przedmiotów nietrwałych. Zgodnie z opracowanymi w ZETO Poznań kompleksowymi założeniami systemu gospodarki materiałowej przewiduje się poszerzenie sytemu o planowanie i kontrolę realizacji dostaw, normatywne rozliczanie zużycia materiałów, planowanie zużycia materiałów oraz planowanie potrzeb materiałowych.

System emituje 48 tabulogramów wynikowych, o wyborze których decyduje użytkownik, oraz cztery tabulogramy kontrolne w postaci raportów błędów i raportów aktualizacji. Zakres informacji w tabulogramach został tak pomyślany, by przedsiębiorstwo z chwilą wdrożenia Systemu mogło zrezygnować z dotychczasowej techniki prowadzenia ewidencji ilościowo-wartościowej materiałów i przedmiotów nietrwałych. Poza tym System może dostarczać informacji w wielu różnych układach, co przy tradycyjnym sposobie prowadzenia ewidencji — z powodu dużej pracochłonności — nie jest możliwe.

Warunkiem wdrożenia systemu SIGMAT-R jest opracowanie bazy normatywnej, wprowadzenie wzorów dokumentów źródłowych (zgodnie z wymaganiami Systemu), opracowanie instrukcji wypełniania, kontrolowania i obiegu dokumentów źródłowych, utworzenie komórki lub stanowiska ds. przetwarzania, a także przeszkolenie pracowników w zakresie wypełniania i kontrolowania dokumentów źródłowych oraz korzystania z informacji zawartych w tabulogramach.

System SIGMAT-R jest przeznaczony do eksploatacji na komputerach Jednolitego Systemu pod nadzorem systemu operacyjnego DOS. Wersja systemu o nazwie SIGMAT-1 może być eksploatowana na komputerach serii ODRA 1300.

SIGMAT jest eksploatowany w kilku przedsiębiorstwach: m. in. w Wojewódzkich Zakładach Motoryzacyjnych w Poznaniu — w wersji na komputery serii ODRA 1300 (SIGMAT-1), w Wojewódzkim Przedsiębiorstwie Komunikacyjnym w Kaliszu, Zakładach Sprzętu Oświetleniowego POLAM Piła i Pabianice, Wielkopolskich Zakładach Przemysłu Ziemniaczanego. SIGMAT-R jest wdrażany w Zakładach Piwowarskich w Koszalinie, Fabryce Maszyn i Urządzeń Pakujących SPOMASZ w Gnieźnie, Rejonowym Przedsiębiorstwie Przemysłu Paszowego BACUTIL w Gnieźnie i w Kombinacie POLSREBRO w Poznaniu. Dostarczając szerszej informacji w sposób szybki i wieloaspektowy system SIGMAT-R stanowi istotny środek usprawnienia organizacji i zarządzania w przedsiębiorstwie.

SIGMAT-R cechuje się takimi właściwościami, jak: wyliczanie cyfr kontrolnych dla kodu towarowo-materiałowego (KTM) oraz wydruk pełnych indeksów, sporządzanie wykazu przeprowadzonych w ciągu roku inwentaryzacji materiałów i przedmiotów nietrwałych, dokonywanie przeceny zużycia materiałów za okres roku bieżącego (okres poprzedzający zmianę cen ewidencyjnych) oraz ubiegłego (za analogiczny okres od zmian cen ewidencyjnych roku bieżącego).

SYstem ewidencji taśm MagnETycznych — SYMET

SYMET jest przeznaczony dla ośrodków obliczeniowych i służy do ewidencjonowania i rozliczania kosztów za dzierżawę taśm magnetycznych. Służbie eksploatacyjnej ośrodka dostarcza on informacji o ruchu i aktualnym stanie wszystkich taśm magnetycznych w ośrodku; dzierżawcom taśm dostarcza danych o stanie taśm i o kosztach dzierżawy.

Funkcję tę System realizuje poprzez prowadzenie kartoteki wszystkich taśm magnetycznych eksploatowanych w ośrodku obliczeniowym, aktualizowanie tych kartotek, tworzenie i aktualizowanie zbioru informacji o wydzierżawionych taśmach magnetycznych — przez obliczanie należności z tego tytułu i sporządzanie odpowiednich tabulogramów. Funkcje te wykonuje siedem oddzielnych programów wchodzących w skład pakietu. Dokumentami źródłowymi są: wypełniany przez operatora wykaz taśm magnetycznych, wypełniane przez jej użytkownika polecenie kasowania taśmy oraz raport używanych taśm

SYMET eksploatowany jest w ZETO Poznań, w jego zakładach obliczeniowych w Kaliszu i Pile oraz w ZETO Łódź.

System jest obecnie eksploatowany przy użyciu komputerów Jednolitego Systemu pod nadzorem systemu operacyjnego OS (poprzednio tylko na komputerach ODRA 1300). Opierając się na doświadczeniach w upowszechnianiu systemu SYMET można przewidywać, że w swej nowej wersji SYMET-OS znajdzie również wielu użytkowników w ośrodkach pracujących na komputerach JS i IBM 360/370, ponieważ zapewnia uporządkowanie zapisów taśmowych i obiegu dokumentów, jak również ułatwienie ewidencjonowania i rozliczania.

SYSTEMY AUTOMATYCZNEJ OCENY DZIAŁALNOŚCI JEDNOSTEK GOSPODARCZYCH

Na koniec — systemy ZETO Poznań zupełnie odmiennego rodzaju i przeznaczenia: systemy umożliwiające automatyzację oceny działalności przedsiębiorstw i zjednoczeń oraz oceny wpływu struktury asortymentowej na wyniki ekonomiczne przedsiębiorstw przemysłowych.

Parametryczny uniwersalny Automatyczny System Oceny działalności jednostek i organizacji gospodarczych — ASO-AM

ASO-AM usprawnia i automatyzuje badanie zjawisk gospodarczych. Obszar, zakres i cel oceny analizowanego obiektu ustala sam użytkownik, który sporządza odpowiedni model diagnostyczny lub symulacyjny oraz wyznacza analizatory poszczególnych obszarów. System ten można stosować dla dowolnego modelu arytmetycznego lub logicznego, co pozwala na uwzględnienie wszelkich niestabilności, wynikających z charakteru poszczególnych gałęzi gospodarki narodowej, przepisów oraz polityki ekonomicznej. System dopuszcza też przekształcanie modelu diagnostycznego na symulacyjny poprzez określenie zmiennych decyzyjnych, wskaźników symulowanych oraz sformułowanie równań i kryteriów oceny.

Pełne wykorzystanie wszystkich funkcji Systemu możliwe jest przy jednoczesnym analizowaniu organizacji gospodarczej i podległych jej jednostek organizacyjnych lub przedsiębiorstw wielozakładowych. ASO-AM w zależności od potrzeb użytkownika dostarcza następujące wydruki:

- bazy danych, uporządkowany według grup tematycznych dla jednostki organizacyjnej i żądanego okresu
- wyników działalności gospodarczej według wzorców dla jednostki organizacyjnej lub jednostki gospodarczej i żądanego okresu.
- wartości dynamik oraz udział jednostek gospodarczych w wynikach działalności organizacji gospodarczej według wzorca i dla żądanego okresu
- czynników wpływających na poziom kształtowania określonej wielkości ekonomicznej w kategoriach charakterystycznych dla jednostki gospodarczej i wybranego okresu (w przypadku modelu diagnostycznego)
- udziału jednostek gospodarczych w uzyskaniu podstawowych wyników ekonomicznych w organizacji gospodarczej, w żądanym okresie i dla wybranych przez użytkownika wielkości ekonomicznych

• zestawienia wariantów modelu symulacyjnego dla jednostki organizacyjnej w żądanym okresie

• zestawienia zbiorczego wskaźników ocen według kryterium symulacyjnego dla jednostki organizacyjnej w żądanym okresie

• wartości dynamik oraz udziału jednostek gospodarczych w symulowanej działalności organizacji gospodarczej, według żądanego kryterium dla żądanego okresu.

Dla zapewnienia prawidłowej i wszechstronnej eksploatacji Systemu nie trzeba dokonywać jakichkolwiek zmian organizacyjnych, nie jest też konieczne zatrudnienie dodatkowych pracowników u użytkownika. Osobą koordynującą pracę oraz merytorycznie nadzorującą przebieg eksploatowania Systemu i wykorzystywania wyników powinien być kierownik działu ekonomicznego, natomiast wszystkie funkcje wykonawcze i techniczne należą do operatora Systemu.

ASO-AM jest przeznaczony do eksploatacji na komputerach Jednolitego Systemu pod nadzorem systemu operacyjnego DOS przy minimalnej konfiguracji obejmującej 30 MB pamięci dyskowej, jednostkę pamięci taśmowej, czytnik kart i drukarkę.

System został wdrożony m.in. w Zjednoczeniu Przemysłu Taboru Kolejowego w Poznaniu; Zakładach Naprawczych Taboru Kolejowego w Poznaniu oraz Zjednoczeniu Przemysłu Sprzętu Optycznego i Medycznego.

Inny wariant tego typu systemu to:

Problemowa ocena wpływu struktury asortymentowej na wyniki działalności przedsiębiorstw przemysłowych — PA-T/79

System PA-T/79 umożliwia dokonywanie ocen wpływu zmian struktury asortymentowej na wyniki działalności przedsiębiorstw przemysłowych. Bada on ten wpływ na poziom zmian produkcji dodanej, akumulacji, pracochłonności rzeczywistej i normowanej, wybranych do analizy materiałów (zwłaszcza limitowanych), a także wpływ zmian cen i kosztów na poziom akumulacji i produkcji dodanej. System może dostarczać trzy rodzaje ocen: bieżącą (tj. analizę wykonania planu), retrospektywną (tj. analizę wykonania w porównaniu z wykonaniem okresu poprzedniego) oraz prospektywną (tj. analizę planu okresu następnego).

System jest przeznaczony do eksploatacji na komputerach Jednolitego Systemu pod nadzorem systemu operacyjnego DOS, przy minimalnej konfiguracji obejmującej jednostkę pamięci dyskowej 7,25 MB, trzy jednostki pamięci taśmowej, czytnik kart i drukarkę wierszową. W razie stosowania dysków o pojemności 30 MB zachodzi konieczność odpowiedniego zaadaptowania Systemu. System PA-T/79 został wdrożony m.in. w Zjednoczeniu Przemysłu Taboru Kolejowego w Poznaniu i Zakładach Naprawczych Taboru Kolejowego w Poznaniu.

Wdrażanie powyższych systemów oceny przebiegać będzie pomyślnie, jeśli użytkownik — bez konieczności dokonywania jakichkolwiek zmian organizacyjnych u siebie — będzie umiejętnie wykorzystywał dane zawarte w sprawozdaniach przesyłanych do GUS-u i planach społeczno-gospodarczych oraz korzystał z danych zawartych w bieżącej ewidencji. Opanowanie poprawnego zbierania danych decyduje o jakości informacji wynikowych oraz prawidłowym przebiegu eksploatacji systemów.

•

Na tym kończymy prezentację systemów ZETO Poznań, przeznaczonych do wielokrotnego stosowania u użytkowników. Interesujące byłoby niewątpliwie, gdyby na łamach INFORMATYKI można było znaleźć podobne przeglądy dorobku innych ośrodków projektujących, również spoza sieci Zjednoczenia Informatyki.

Henryk ADAMCZEWSKI
Zakładowy Ośrodek INTE
ZETO Poznań

Metodyka tworzenia systemów informatycznych

Na podstawie materiału ankietowego otrzymanego z 16 największych zakładów sieci ZETO opracowano syntetyczny opis metodyk projektowania systemów informatycznych stosowanych w Zjednoczeniu Informatyki. Prace prowadzone były na zlecenie Ośrodka Normalizacji ZI przy ZETO Katowice. Odpowiedzi, związane z okresem drugiej połowy 1980 r., pozwoliły na przeanalizowanie głównych zagadnień związanych z procesem tworzenia systemów informatycznych.

TWORZENIE SYSTEMÓW INFORMATYCZNYCH

W zakładach sieci ZETO tworzone są przede wszystkim systemy: indywidualne (obiektowe), powtarzalne (typowe), z zastosowaniem bazy danych oraz z wykorzystaniem odpowiedniego oprogramowania. Ponadto w pojedynczych przypadkach tworzone są systemy lub oprogramowanie uniwersalne i narzędziowe. Na 16 badanych Zakładów — 13 stosuje obowiązującą oficjalnie „Metodykę projektowania systemów informatycznych" OBRI Warszawa (z 1977 r.). Niemniej większość z nich stosuje również inne opracowania metodyczne, własne lub obce: instrukcje w sprawie zawartości dokumentacji, wytyczne, standardy, zasady postępowania przy realizacji określonych prac, dokumentację określonych etapów prac oraz vademecum programisty. Są to głównie opracowania ZETO we Wrocławiu, Gdańsku, Bydgoszczy, Łodzi i Opolu.

Jednocześnie realizowane są obecnie prace nad następującymi opracowaniami unifikacyjnymi:

• Metodyka realizacji zastosowań informatycznych przy wykorzystaniu koncepcji wspólnej bazy danych, ze szczególnym uwzględnieniem przetwarzania transakcyjnego;

• Metodyka rozwoju zastosowań przy użyciu systemu STEP;

• Rozliczanie pracy komputerów JS według zasobów;

• Kwalifikacja obciążenia systemów cyfrowych (do oceny bilansowania mocy komputerów);

• Pakiet projektowania strukturalnego.

Podział procesów tworzenia systemów informatycznych na określone etapy i fazy jest w zakładach sieci ZETO bardzo zróżnicowany. Jedynie dwa Zakłady stwierdzają, że proces ten dzielony jest na etapy zgodnie z ustaleniami cennika usług informatycznych (Nr 1-U/80) oraz metodyką OBRI. Wszystkie Zakłady w zasadzie wyróżniają podstawowe etapy procesu tworzenia systemu informatycznego, a niektóre ponadto — etapy związane z pracami wstępnymi oraz z eksploatacją i konserwacją systemu. Istniejący stan jest wynikiem własnej, wieloletniej praktyki oraz wypracowanych metod pracy w poszczególnych

Zakładach ,a także niejednakowego pojmowania procesu tworzenia systemów informatycznych.

Występuje duża różnorodność metod, technik i narzędzi pracy stosowanych w poszczególnych etapach procesu tworzenia systemów. Do przodujących w rozpoznawaniu i opanowywaniu zachodnich metod, technik i narzędzi pracy oraz opracowywaniu własnych narzędzi pomocniczych, należy zaliczyć ZETO we Wrocławiu, Krakowie, Białymstoku, Opolu, Łodzi, Poznaniu i Lublinie. Pozostałe Zakłady posługują się nadal przede wszystkim tradycyjnymi metodami, próbując ewentualnie niektórych „nowości" przy pracach programowych.

Aktualny zestaw metod, technik i narzędzi pracy stosowanych w zakładach sieci ZETO zawiera tabela.

Tabela. Metody, techniki i narzędzia pracy stosowane w Zakładach sieci ZETO

Etapy prac	Metody	Techniki	Narzędzia
Analiza systemowa	Analiza systemowa Metoda wywiadowa Metoda diagnostyczna Strukturalna Analiza Systemowa (SAS) Systems Analysis and Design Technique (SADT)	Tablice decyzyjne HIPO Analiza funkcjonalna, np. wg metody BISAD	Standardowe ankiety do badania istniejącego stanu Typowe formularze Wykresy i schematy procedur Tablice decyzyjne
Założenia systemu	Strukturalna Analiza Systemów (SAS) Metoda „top-down" HIPO Metoda diagnostyczna Metoda prognostyczna	HIPO Tablice decyzyjne Arkusz powiązań danych Formy graficzne	Karty struktury systemu Schematy ogólne HIPO Schematy powiązań Schematy przetwarzania
Projekt techniczny systemu	HIPO Formularzowa metoda opisu procesu przetwarzania danych Projektowanie modularne	Tablice decyzyjne HIPO Instrukcje ZETO Wrocław dotyczące zawartości dokumentacji projektowej i programowo-eksploatacyjnej systemu informatycznego	USZBD-RODAN STEP AWIT PAKOW
Projekt techniczny oprogramowania	Metoda programowania przez łączenie elementów zestandaryzowanych Metoda projektowania oprogramowania Jacksona HIPO Strukturalne projektowanie programów	Modularne przygotowanie oprogramowania Tablice decyzyjne Programowanie „top-down" Programowanie przez łączenie elementów standardowych	Generatory programów źródłowych SZBD Biblioteka zadań COBOL-u z opisami typów zapisów Formularze opisu procesu przetwarzania Standardy własne
Oprogramowanie systemu	Metoda „top-down" Programowanie strukturalne Programowanie modularne Metoda łączenia elementów standardowych	Programowanie strukturalne Programowanie modularne Technika funkcjonalna Łączenie elementów standardowych Rozwiązania z bazą danych Programowanie „top-down" Zespół programisty wiodącego	projekt programów strukturalnych [niepełne pakiet PATEST system testowania modułów STEM segmentacja (nakładkowość) kodowanie programów biblioteka zadań źródłowych [USZBD-RODAN DMS-2 Generator raportów SYWIN Tablice decyzyjne-algorytmy Pakiet uzupełnień DOS-PUDOS — programy reorganizacji zbiorów sekwencyjnych i indeksowo-sekwencyjnych System SZKRAB — ujednolicenie technologii sporządzania wydruków Pakiet generacyjny STEP Pakiet programowania liniowego LPS/JS Procedury kontroli formalnej Zakładanie i aktualizowanie zbiorów Wydruk nagłówków na RIAD Pakiety wspomagające ZETO-FLOW ZETO-TEST System CRJE i TSO — konwersacyjne zdalne wyprowadzanie zdań System generowania programów źródłowych (ODRA) - MAKROCOBOL Dokumentowanie prac programowych — FLOWCHARTER Pakiet automatyzacji procesu testowania programów (ODRA) OLIMP-20
Wdrażanie systemu	bezpośrednia równoległa pilotowa		

Całość dokumentacji tworzonego systemu informatycznego związana jest z liczbą wyróżnianych etapów. Ponieważ liczba etapów w procesie tworzenia systemu jest zróżnicowana, stąd też liczba części dokumentacji jest różna i waha się od sześciu do dziewięciu. Szereg Zakładów wykonuje dokumentację stosownie do rodzaju tworzonego systemu, zgodnie z własnymi ustaleniami i instrukcjami. Dokumentacja określonych etapów prac oraz zalecane wzory formularzy w metodyce OBRI nie znalazły w praktyce szerszego zastosowania. Stąd zróżnicowany zakres i stopień szczegółowości, co powoduje duże trudności przy rozpowszechnianiu systemów.

W każdym Zakładzie dokonywana jest ocena wewnętrzna prac projektowo-programowych (za pomocą analizy dokumentacji poszczególnych etapów prac). Jest ona zróżnicowana i występuje albo w formie odbioru dokumentacji na posiedzeniu zakładowej komisji, zespołu lub rady technicznej, albo oceny i weryfikacji przez kierownika określonego działu lub zespołu, względnie przez wyznaczonego specjalistę.

STOSOWANE METODYKI I TECHNIKI

Ocena efektywności stosowanych metodyk (głównie OBRI) przez poszczególne Zakłady jest zróżnicowana. Niemniej nieliczne tylko Zakłady pozytywnie oceniają efektywność metodyki OBRI i tylko dwa z nich nie zgłosiły stwierdzonych w tej metodyce wad.

Ogólnie, wskazane wady dotyczą takich aspektów metodyki OBRI, jak:

- charakter metodyki oraz stopień szczegółowości
- zakres tworzonych systemów informatycznych
- przydatność załączonych wzorów formularzy
- powtarzalność niektórych elementów w dokumentacji etapów
- brak szeregu nowoczesnych metod, technik i narzędzi pracy
- brak zgodności z cennikiem usług informatycznych
- brak wskazówek odnośnie organizacji prac projektowo-programowych.

Zasadnicze wady — jak wynika z ankiety — to:

- konieczność powtarzania pewnych elementów dokumentacji w różnych etapach prac
- trudne modyfikowanie dokumentacji systemu
- brak aktualnie stosowanych metod, technik i narzędzi pracy
- brak metodycznych wskazań („jak robić"), a głównie omówienia składu dokumentacji poszczególnych etapów prac
- mała elastyczność w stosunku do potrzeb różnych systemów informatycznych
- za duża pracochłonność oraz papierochłonność dokumentacji
- zbyt duża liczba formularzy, w tym część nie dostosowana do potrzeb
- niewystarczające uwzględnienie nowych technologii przetwarzania danych i możliwości technicznych sprzętu (np. baz danych).

Pozostałe metodyki projektowania systemów informatycznych to różne indywidualne opracowania stosowane na ogół w tych Zakładach, w których powstały. Wyjątek stanowią opracowania ZETO Wrocław, które (przedstawione w formie instrukcji) stosowane są również w kilku innych Zakładach.

Oceniając zalecane w metodyce OBRI techniki stwierdzić trzeba, że ograniczają się one — poza formą opisu słownego — jedynie do form graficznych, w postaci rysunków i schematów w załączonych wzorach formularzy oraz tablic krzyżowych do powiązań danych. Ocena generalna obowiązującej metodyki OBRI jest negatywna; wynika to głównie z braku uwzględnienia: wszystkich rodzajów opracowywanych obecnie systemów informatycznych; stosowanych i zalecanych w ostatnich latach nowych metod, technik i narzędzi pracy; wskazań metodycznych dla twórców systemów, odpowiadających na pytanie „jak zrobić" oraz wskazówek dotyczących organiazcji prac projektowo-programowych.

POTRZEBY UŻYTKOWNIKÓW METODYKI

W przyszłej, zmodyfikowanej metodyce tworzenia systemów informatycznych w sieci ZETO należy m.in. uwzględnić następujące zasady:

- tworzony system powinien w optymalny sposób wykorzystywać właściwości komputera, przy użyciu którego ma być eksploatowany
- wybór metod, technik i narzędzi pracy powinien być podyktowany wymogami prac projektowo-programowych oraz wymogami technologicznymi użytkowego przetwarzania w ramach systemu
- tworzony system powinien gwarantować łatwą jego konserwację przez osoby spoza zespołu autorów
- dokumentacja systemu powinna być zwarta, jednoznaczna dla użytkownika i ośrodka obliczeniowego oraz łatwo modyfikowalna.

Opracowanie nowej, ujednoliconej metodyki dla wszystkich Zakładów sieci ZETO jest potrzebne z uwagi na duże zróżnicowanie poziomu omawianych prac w poszczególnych Zakładach. Nowa metodyka pozwoli zapoznać wszystkich twórców systemów informatycznych z nowoczesnymi metodami, technikami i narzędziami prac projektowo-programowych, co wpłynie na podniesienie nowoczesności i jakości tworzonych systemów oraz ułatwi współpracę między Zakładami.

Zdaniem większości Zakładów sieci ZETO, nowa metodyka powinna:

- omawiać w ogólnych ramach główne problemy prac projektowo-programowych oraz specyficzne prace związane z tworzeniem poszczególnych rodzajów systemów i rozwiązań technologicznych, wynikających ze specyfiki komputerów ODRA i RIAD
- posiadać załączniki w formie szczegółowych instrukcji, zawierających specyfikację czynności do wykonania w ramach określonych etapów prac projektowo-programowych
- zawierać szczegółowe omówienie sposobu wykonania poszczególnych prac i czynności.
- rozwiązywać problemy związane z tworzeniem, wdrażaniem, eksploatacją i konserwacją systemów
- zawierać zasady efektywnej organizacji prac projektowo-programowych oraz kryteria i zasady oceny i weryfikacji wyników prac poszczególnych etapów
- zawierać charakterystykę przydatności nowych metod, technik i narzędzi pracy i zasad ich stosowania w pracach projektowo-programowych.

Ponadto, w aspekcie wymogów i kryteriów jakościowych, nowa metodyka powinna być zgodna z prawidłową technologią tworzenia systemów oraz z istniejącymi normami i cennikami, powinna zapewniać efektywność prac oraz gwarantować zwięzłą, jednoznaczną i łatwo modyfikowalną dokumentację systemów. Bardzo istotnym elementem jakościowym przyszłej metodyki będzie ujęcie w niej nowych i sprawdzonych w praktyce przez Zakłady w kraju metod, technik i narzędzi pracy z krajów zachodnich. Z nowości zgłoszono do rozpatrzenia i stosowania:

Metody:

- Analizy strukturalne

— Structured Analysis (SA)
— Structured Systems Analysis (SSA)

- Projektowanie oprogramowania Constantine'a
- Projektowanie oprogramowania Warnuer'a

Techniki:

- Transform Analysis L. Constantine'a
- Sterowanie przetwarzaniem zadań z TM w systemie operacyjnym DOS/JS
- Technika interpretacyjnego raportowania
- PSL/PSA
- PROTEE

Narzędzia:

- SYKON, SYWIN, PROSTR, PSL/PSA, PROTEE
- Słownik Danych
- Pakiet konwersji zbiorów RIAD-ODRA.

Dokumentacja tworzonych systemów według nowej metodyki powinna:
- składać się z zasadniczych części wykonywanych w ramach określonych etapów prac
- zawierać formy graficzne (schematy, tablice, formularze itp.)
- zapewniać niezbędną komunikatywność, czyli:
  — prawidłowość i jednoznaczność terminologii
  — łatwość modyfikacji i konserwacji
  — zwięzłość opracowania
  — niepowtarzalność treści
  — aktualność.

Powinien też następować obowiązkowy odbiór wewnętrzny (w Zakładzie) dokumentacji poszczególnych etapów prac przed przekazaniem do zleceniodawcy.

NOWA METODYKA

Na podstawie przeprowadzonej ankiety, w ZETO Gdynia opracowano model wzorcowy metodyki, przeznaczony dla wszystkich Zakładów Zjednoczenia Informatyki. Generalnie przyjęto, że metodyka ta jest logicznym, ułożonym chronologicznie zbiorem wskazań dotyczących metod i technik, które należy stosować. Opisany w metodyce proces tworzenia systemów informatycznych obejmuje prace związane z całym cyklem „życia systemu", a więc z jego opracowaniem, wdrożeniem, eksploatowaniem i konserwowaniem. Metodyka jest w dużej mierze funkcją kwalifikacji kadr projektowo-programowych, stanu sprzętu komputerowego i jego oprogramowania oraz metod, technik i narzędzi pracy, a ponieważ czynniki te rozwijają się dynamicznie — metodyka musi być okresowo aktualizowana (minimum co 2—3 lata).

Nadrzędnym celem nowej metodyki ma być wzrost efektywności procesu tworzenia systemów. Przyjmuje się zasadę, że jest j e d n a metodyka tworzenia systemów informatycznych, a jej zakres powinien obejmować główne problemy prac projektowo-programowych oraz wdrożeniowych i konserwacyjnych, a ponadto specyficzne problemy związane z tworzeniem poszczególnych rodzajów systemów informatycznych.

Proponuje się poniższą formę przyszłej metodyki:

Część zasadnicza metodyki. Zawierać będzie omówienie problemów zasadniczych, a więc:
- pojęcia systemu informatycznego, jego budowy oraz ogólnej charakterystyki wyróżnionych rodzajów systemów
- modelu procesu tworzenia systemu informatycznego, a w tym takich elementów, jak: stadia, etapy, fazy, kroki i czynności
- typowych prac w poszczególnych etapach oraz ogólnej charakterystyki oferowanych metod, technik i narzędzi pracy
- organizacji prac projektowo-programowych, tj. prac zespołu projektowo-programowego lub zespołu programisty wiodącego
- zasad dokumentowania wyników prac, a także wymogów dokumentacji i jej zawartości
- kontroli realizacji prac związanych z tworzeniem systemu, kryteriów oceny wyników etapów oraz odbioru wyników prac
- udziału przyszłego użytkownika systemu w tworzeniu systemu oraz jego przygotowania do wdrożenia i eksploatacji systemu.

Załączniki szczegółowe metodyki. Zawierać będą w oddzielnych opracowaniach omówienie sposobu wykonywania określonych czynności, w ramach poszczególnych etapów prac oraz specyfikację dokumentacji. W formie oddzielnych załączników przewiduje się opracowanie:
- specyfikacji czynności i sposobu ich wykonania, przy tworzeniu określonych rodzajów sytemów informatycznych oraz szczegółowej zawartości dokumentacji
- sposobu stosowania określonych metod, technik i narzędzi w pracach projektowych i (lub) programowych oraz towarzyszącej im dokumentacji
- szczegółowej problematyki organizacyjnego, technicznego i kadrowego przygotowania przyszłego użytkownika systemu.

Nowa metodyka powinna być opracowana w formie specjalnych skoroszytów, zawierających wpięte kartki opracowania. Stworzy to dogodne warunki bieżącej aktualizacji poszczególnych części metodyki.

Mając na względzie powszechne stosowanie metodyki, przewiduje się:
- ścisłe przestrzeganie zaleceń metodyki przez wszystkie Zakłady
- zapewnienie pełnej swobody Zakładom w stosowaniu zaleceń metodyki oraz możliwości doboru odpowiednich metod, technik i narzędzi pracy (spośród zaproponowanych), odpowiednio do potrzeb określonego systemu i wymogów przyszłego użytkownika
- zezwolenie na stosowanie własnych narzędzi w pracach projektowo-programowych, po uprzednim uzgodnieniu z Ośrodkiem Normalizacji.

Jednocześnie zapewnienie sprawnej i efektywnej pracy zespołów twórców systemów informatycznych wymagać będzie odpowiedniego zgromadzenia narzędzi wspomagających przede wszystkim prace projektowe, programowe i wdrożeniowe.

Postuluje się zorganizowanie warsztatu pracy projektanta i programisty, który będzie sukcesywnie uzupełniany i aktualizowany.

Obejmować on powinien między innymi:
- metodykę tworzenia systemów informatycznych
- komplet najnowszych w kraju metod, technik i narzędzi pracy
- inne zalecane opracowania metodyczne z zakresu projektowania, programowania oraz wdrażania i konserwowania systemów
- standardy projektowo-programowe
- PN, BN, ZN w zakresie tworzenia systemów
- oprogramowanie użytkowe.

Prace metodyczno-normalizacyjne nad opracowaniem nowej metodyki dla zakładów sieci ZETO realizowane będą w ramach prac badawczo-rozwojowych Ośrodka Normalizacji ZI w latach 1981—1985.

Nowa metodyka powstawać będzie sukcesywnie, tzn. kolejno będą regulowane i opracowywane zagadnienia tworzenia, wdrażania, eksploatowania i konserwowania systemów, łącznie z opracowywaniem potrzebnych narzędzi pracy i norm zakładowych. Na uwagę zasługuje fakt, że będzie ona narzędziem nie tylko dla projektantów i programistów w Zakładach sieci ZETO, ale również w innych zakładach i ośrodkach usług informatycznych w kraju.

Roman RONKOWSKI
ZETO Gdańsk

Samorząd kontra Zjednoczenie

Wstępny okres organiazcji samorządów w sieci ZETO zakończyło Zebranie Ogólne Delegatów Samorządów Zakładów (Łódź, 5 października 1981 r). Do wniosku (skierowanego do MNSzWiT) o niezwłoczne rozwiązanie Zjednoczenia Informatyki dołączono wówczas postulat powołania mieszanej komisji likwidacyjnej (przy założeniu nieprzekraczalnego terminu zniesienia Zjednoczenia: 31.12.81 r.). Nie wyrażono przy tym zgody na prowadzenie jakichkolwiek negocjacji w sprawie przyszłych form organizacyjnych sieci ZETO przed zakończeniem postępowania likwidacyjnego.

Zniesienie Zjednoczenia, choć już przesądzone, nadal nie jest faktem. Resort wprawdzie wyraźnie stwierdził, iż nie zamierza go bronić, niemniej nie ma ciągle koniecznej (tj. wymaganej przez obowiązujące prawo) uchwały Rady Ministrów. Bez niej — jak twierdzi Ministerstwo — działania likwidacyjne byłyby zbyt skomplikowane.

Samorządy, nie czekając na ostateczne decyzje, przygotowują własny program, wykorzystujący wszystkie dodatnie cechy dotychczasowej formy organizacji.

(Z.)

Najnowsze tendencje w dziedzinie komputerowych systemów sterowania[1)]

Pod auspicjami Międzynarodowej Federacji Sterowania Automatycznego IFAC[2)], a właściwie jej Komitetu ds. komputerów (Technical Committee on Computers) odbyło się w ubiegłym roku kilka konferencji międzynarodowych, związanych tematycznie z dziedziną komputerowych systemów sterowania, a mianowicie:

- 6th IFAC/IFIP Conference on Digital Computer Applications to Process Control (DCAPC), w Düsseldorfie (RFN), 14—17 październik
- 2nd IFAC Workshop on Distributed Computer Control Systems (DCCS), w Ste. Adele (Kanada), 29 wrzesień — 1 październik
- IFAC Workshop on Scientific Experiments and Laboratory Procedures Automation (SELPA) w Smolenicach (Czechosłowacja), 10—12 listopad
- IMEKO/IFAC Symposium on Application of Microprocessors in Devices for Instrumentation and Control (AMDIAC), w Londynie, 18—20 listopad.

Za główną należy uznać pierwszą z wymienionych, organizowaną już po raz szósty, chociaż pozostałe stanowiły jej znakomite uzupełnienie jeśli chodzi o przedstawienie najnowszych kierunków badań. Poniżej zostaną omówione zagadnienia wiodące, które pozwalają stworzyć obraz aktualnych tendencji światowych w tej dziedzinie.

UKŁADY MIKROPROCESOROWE

Fala układów o dużym stopniu scalenia, jaka ostatnio zalewa rynek światowy, wywołała zmiany strukturalne w komputerowych systemach sterowania. Jednakże, postęp technologiczny w dziedzinie układów elektronicznych następuje szybciej niż rozwój struktur sterowania i trudno jest uchwycić właściwe tendencje, gdyż gwałtowność zmian uniemożliwia pełną diagnozę. Nie zawsze robią to nawet autorzy opracowań przeglądowych.

Wpływ techniki mikroprocesorowej na komputerowe systemy sterowania jest dostrzegalny przede wszystkim ze względu na ich lepszą jakość, malejące koszty realizacji oraz istotne zmiany w architekturze (G. Faerber, DCAPC).

Jakość systemów sterowania zbudowanych z elementów o dużym stopniu scalenia osiąga poziom wręcz niewiarygodny. Jako ilustracja tego stwierdzenia może posłużyć fakt, że mikroprocesorowy układ sterowania elektrodami w piecu łukowym wykazał dyspozycyjność (ang. availability) 99,89% w ciągu dwóch lat pracy (G. T. Gray, DCAPC).

Zmniejszenie kosztu realizacji systemów umożliwia zwiększenie liczby oraz łączenie wykonywanych przez nie funkcji. Dotyczy to nie tylko regulatorów konwencjonalnych, lecz także innych funkcji logicznych i obliczeniowych, włącznie z estymacją (L. Fortuna i in., DCAPC) i adaptacją parametrów (H. Boehm i in., DCAPC). Jak istotna jest bariera kosztu, świadczą inne, narazie sporadyczne, próby wykorzystania np. metod przetwarzania optycznego (S. Sajc, P. Valina, SELPA) lub włókien światłowodowych (S. Geckeler, DCAPC). Z całą pewnością barierę tę przekroczono w produkcji programowanych sterowników logicznych, które w wielu przypadkach stają się konkurencją dla mikroprocesorów (H. Westcott, AMDIAC).

Różnorodność zastosowań mikroprocesorów jest ogromna. Nie ograniczając się tylko do układów sterowania wskazać można na szczególnie duże możliwości komputeryzacji pomiarów. Wiele zagadnień z tej dziedziny poruszono na sympozjum londyńskim (AMDIAC), przedstawiając użycie mikroprocesorów w najprostszych pomiarach klasycznych, w analityce, a także — co uważam za najważniejsze — w realizacji urządzeń do przetwarzania sygnałów, takich jak generatory sygnałów testujących lub analizatory częstotliwości, które uzyskują rzeczywistą rację bytu dopiero dzięki zastosowaniu tej techniki.

Zmiany w architekturze sytemów następują przede wszystkim w wyniku przestrzenego rozłożenia mocy obliczeniowej. Jest to jakościowo nowe rozwiązanie, wraz z którym pojawiły się jednak nowe problemy, zarówno w sferze sprzętu, jak i oprogramowania. Zagadnienia sprzętowe dotyczą przede wszystkim rodzaju sieci i połączeń, natomiast zagadnienia programowe — uzyskania odpowiednio wydajnych metod (w tym — języków) programowania tego rodzaju systemów. Tradycyjne metody projektowania w tych przypadkach zawodzą, a ich stosowanie może prowadzić do niewłaściwego działania układów sterowania (D. H. Sawin III, AMDIAC).

SYSTEMY ROZŁOŻONE PRZESTRZENNIE

Możliwość użycia mikroprocesorów w różnych miejscach systemu sterowania, tj. bezpośrednio tam gdzie zachodzi potrzeba, zwiększa znacznie jego zdolności obliczeniowe, sterujące i pomiarowe, lecz stwarza trudności w skoordynowaniu działania tych — w dużym stopniu niezależnych — urządzeń. Powstają nowe zagadnienia układowe i transmisyjne, jak też programowe, wiążące się z potrzebą opanowania zwiększonej liczby zadań wykonywanych — w większości przypadków samodzielnie — przez dużą liczbę niezależnych urządzeń.

Problematyka sytemów rozłożonych przestrzennie jest więc na tyle ważna i dojrzała, że poświęcono jej odrębne sympozjum (DCCS). Oprócz wielu zastosowań przedstawiono na nim także próby normalizacji rozwiązań, co stanowi zagadnienie podstawowe.

Spośród wielu prób normalizacyjnych, dotyczących przyszłej tzw. lokalnej sieci komputerowej, przeznaczonej do wielu różnych zastosowań i to nie tylko do systemów sterowania, należy wymienić przynajmniej trzy (M. J. Kryskow[3)], DCCS). Są to prace dotyczące tzw. magistrali PROWAY (IEC[4)], oraz opracowywanej przez IEEE[5)] sieci zgodnej z wymaganiami ISO[6,7)] i tzw. sieci DIX — wspólnego dzieła firm DEC, INTEL i XEROX, które zamierzają wprowadzić jednolite, zapewniające wzajemną zgodność produkowanych urządzeń, rozwiązanie.

W wielu referatach podkreślono ogromną rolę systemów rozłożonych przestrzennie, stosowanych np. w gazownictwie (G. Lappus, G. Schmidt, DCAPC), w energetyce (T. McNeil i in., DCCS), czy w przemyśle farmaceutycznym (D. J. Fraade, DCCS).

Główne tendencje obecnego rozwoju, to z jednej strony coraz głębsze przenikanie techniki mikroprocesorowej do układów pomiarowych (czujników i przetworników) a z drugiej strony rozprzestrzenienie się mocy o-

---

[1)] Artykuł o takim samym tytule, który można traktować jako część pierwszą tej problematyki, został opublikowany przez autora w nr 9/1980 INFORMATYKI (s. 34—36).

[2)] International Federation of Automatic Control

[3)] J. M. Kryskow, C. K. Miller, Computer Design, Vol. 20, No. 2, p. 22, No. 3, p. 12 (1981).

[4)] IEC — International Electrotechnical Commission

[5)] IEEE — Institute of Electrical and Electronic Engineers

[6)] ISO — International Organization for Standarization

[7)] H. Zimmermann, IEEE. Trans. Comm., Vol. 28, p. 45 (1980).

bliczeniowej i zdolności mikroprocesorów do pełnienia roli systemu przetwarzania danych (H. Kopetz, DCCS). Stwarza to, bardziej niż kiedykolwiek przedtem, rzeczywistą możliwość połączenia podsystemów pomiarowych, układów sterowania, systemów automatyzacji oraz konwencjonalnych systemów przetwarzania danych w jeden zintegrowany system informacyjny, obejmujący swym zasięgiem całą instalację przemysłową lub instytucję.

Prace nad takimi systemami są narazie w stadium eksperymentalnym jednak wiele ich właściwości zostało już zidentyfikowanych. Konieczna jest więc duża niezawodność sprzętu i jego odporność na uszkodzenia, łatwość pielęgnacji oprogramowania oraz zwiększenie funkcjonalności sprzęgów człowiek-komputer. Od tych czynników zależy skuteczność działania instalacji przemysłowej, zakładu lub instytucji, w której zastosowano zintegrowany system informacyjny.

Oprócz prac praktycznych i doświadczalnych prowadzone są za pomocą ścisłego aparatu matematycznego także badania teoretyczne, polegające np. na określeniu równomierności obciążenia poszczególnych procesorów lub — ich szybkości reakcji (C. M. Woodside, DCCS).

Doświadczenia przy programowaniu systemów rozłożonych przestrzennie potwierdzają fakt istnienia szeregu nowych problemów. Wynikają one z niedostosowania dotychczasowych metod programowania do właściwości strukturalnych tych sytemów. Jednym z niezbędnych kroków w kierunku zastosowania języków wysokiego poziomu do programowania mikroprocesorów, stanowiących główny składnik systemów rozłożonych przestrzennie, jest zwiększenie modularności systemów operacyjnych oraz ułatwienie generowania użytkowych modułów wykonawczych (H. U. Steusloff, DCAPC).

ZAGADNIENIA OPROGRAMOWANIA

Złożoność współczesnych systemów sterowania, będąca skutkiem wzrastających wymagań użytkowników, powoduje, że znane dotychczas środki programowania, takie jak język i jego translator, stają się niewystarczające. Już na początkowym etapie projektowania oprogramowania należy przede wszystkim precyzyjnie określić wymagane właściwości systemu sterowania. W tym celu obecne metody projektowe, oparte głównie na doświadczeniu i intuicji, należy zastąpić metodami bardziej systematycznymi. Mają temu służyć omawiane w wielu wystąpieniach różne narzędzia projektowania, np. system EPOS (Entwurfsunterstütztes PEARL — orientiertes Spezifikationssytem), obejmujący wszystkie podstawowe fazy projektowania systemu sterowania: specyfikację, realizację, testowanie i konserwację (R. Lauber, J. Biewald i in., DCPC).

Znaczny nacisk, jaki obecnie kładzie się na precyzyjne projektowanie oprogramowania użytkowego, a szczególnie — jego pierwszą fazę (specyfikację), wynika z faktu, że zawodność oprogramowania jest spowodowana najczęściej błędami w działaniu programów, powstałymi głównie wskutek niewłaściwego projektowania. Dlatego, słuszna jest generalna zasada pisania programów mająca w pierwszym rzędzie ułatwić ich weryfikację i atestację (W. D. Ehrenberger, DCAPC). Im wcześniej rozpoczniemy formalizację prac związanych z projektowaniem oprogramowania, tym łatwiej będzie je weryfikować. Nie należy przy tym zapominać, że sposoby weryfikacji programów muszą być na tyle łatwe, aby umożliwić wykonywanie tego zadania użytkownikom o przeciętnych kwalifikacjach programistycznych.

Oczywiście zarówno wymienione zadania, jak i inne, wynikające z zależności czasowych, jakim podlegają komputerowe systemy sterowania, wymagają istnienia specjalnych, mających odpowiednie cechy języków wysokiego poziomu. Jednym z takich języków jest PEARL (T. Martin, DCAPC), stosowany obecnie z powodzeniem (głównie w RFN) do automatyzacji różnego rodzaju procesów przemysłowych. Podstawowa wersja tego języka stanowi już w RFN normę krajową (DIN 66253, 1979), która została dostarczoną także do rozpatrzenia przez ISO. Sprawa normalizacji języków programowania systemów czasu rzeczywistego jest — ze względów użytkowych — nadal istotna, choć ostatnio, wraz z postępami w opracowaniu języka ADA, nieco mniej widoczna. Pewne jest jednak, że stosowanie standardowych języków wysokiego poziomu jest jedynym sposobem uniezależnienia oprogramowania od gwałtownych zmian w technologii eksploatacji sprzętu.

Z przebiegu omawianych konferencji wynika, że nie dokonano istotnego postępu w rozwoju systemów operacyjnych. Może to nastąpić dopiero z chwilą wprowadzenia nowej, jednolitej koncepcji, jeśli chodzi o synchronizację procesów i ochronę użytkowanych przez nie obszarów pamięci. Ma to szczególne znaczenie w programowaniu systemów o rozłożonej inteligencji. Podstawę oprogramowania tych systemów stanowią, jak dotychczas, jedynie ustalenia odnośnie protokołów komunikacyjnych na różnych poziomach, np. według modelu organizacji sieci zaproponowanego przez ISO. Obecnie, we wszystkich zastosowanych regułą jest przyjęcie protokółów HDLC (High-Level Data Link Control) lub X.25, ustalonych przez organizacje międzynarodowe ISO i CCITT.[8]).

[8]) CCITT — Comite Consultatif International Telegrafique et Telefonique UIT (Union International des Telecomunications).

WSPÓŁPRACA CZŁOWIEKA Z KOMPUTEREM

Wzrost liczby mikroprocesorów używanych w systemach sterowania jak też rozprzestrzenienie tych systemów i zwiększenie samodzielności poszczególnych jednostek funkcjonalnych, nie oznacza wcale zastąpienia operatora procesu i zwolnienia go od wykonywania wszystkich zadań. Choć złożoność funkcji przypadających poszczególnym jednostkom na ogół się zmniejsza, to system automatyzacji jako całość staje się znacznie bardziej skomplikowany. Zwiększa się zatem rola i odpowiedzialność operatora czuwającego nad tą całością. Choć, prawdą jest, że zadania operatora ulegają pewnemu ograniczeniu, to pozostałe jego funkcje stają się o wiele bardziej złożone. Rola operatora zmienia się więc tak, jak zmieniają się jego funkcje, które ze sterujących przekształcają się na nadzorcze (G. Geiser, DCAPC).

Okazuje się jednak, że nie we wszystkich przypadkach (dotyczy to w szczególności procesów bardziej złożonych) operator dysponuje dostatecznie jasnym obrazem procesu lub skutecznym zestawem środków oddziaływania. Przykładowo, znaczne trudności może sprawić odróżnienie istotnego alarmu od sygnalizacji mniej ważnych stanów krytycznych.

Niedawna awaria reaktora na wyspie Three Miles Island, w USA, stanowi jaskrawy przykład, jak można uniknąć wypadku dysponując lepszym układem sprzęgającym człowieka z komputerem. Istota problemu tkwi w tym, że w tak złożonym systemie, jak układ sterowania elektrownią jądrową nie wystarcza sztywne przypisanie priorytetów określonym stanom alarmowym. Znaczenie wykrytej awarii zależy od konkretnej sytuacji i wywołanie alarmu, konieczne w danych warunkach, może być bez znaczenia w innych (M. R. Johnson, AMDIAC).

Rozwój prac nad stworzeniem operatorowi możliwości wygodnego i skutecznego współoddziaływania z procesem idzie we wszystkich kierunkach związanych z wykorzystaniem trzech zmysłów angażowanych w proces komunikacji z maszyną, tj. wzroku, dotyku i słuchu. W związku z tym zasadniczej zmianie ulega wygląd pomieszczenia operatora, które nie zawiera już niezliczonej ilości przyrządów i wskaźników, lecz jedynie kilka monitorów ekranowych z oprogramowaniem graficznym. Szczególnie zaawansowane są prace nad rozwojem układów do współpracy człowieka z komputerem w systemach sterowania elektrowni jądrowych (B. Wahlstroem, L. Tuominen, DCAPC) i konwencjonalnych (W. Pillmann i in., DCAPC).

AUTOMATYZACJA BADAŃ DOŚWIADCZALNYCH

Oprócz wielu zastosowań komputerów do sterowania różnymi procesami przemysłowymi, wyróżniłbym zupełnie odrębny obszar zastosowań tzw. automatyzację laboratoriów (ang. laboratory automation). Tematyce tej poświęcono oddzielne sympozjum (SEPA), którego celem było dokonanie przeglądu zagadnień teoretycznych i praktycznych w tej dziedzinie.

Za jedną z najważniejszych tendencji w automatyzacji laboratoriów badawczych należy tu uznać dążenie do normalizacji sprzętu. Spełnienie tego warunku jest to o tyle istotne, że wówczas eksperymentatorzy będą mogli przy użyciu standardowego sprzętu sami konfigurować zestawy pomiarowo-sterujące, nie angażując specjalistów od techniki komputerowej. Rolę znormalizowanego sprzętu w zestawach pomiarowo-sterujących (np. systemu CAMAC lub wg normy IEC-625) podkreślało wielu autorów (J. Niesterichin, R. Chmurny i in., P. Horvath i in., J. Lukacs).

Konieczność normalizacji oprogramowania, zarówno języków i systemów operacyjnych, jak też programów użytkowych, nie jest jeszcze dostatecznie głęboko zrozumiana i na tym sympozjum — z jednym wyjątkiem (Z. Banasik i in.) — nie była poruszana. Wydaje mi się, że odzwierciedla to ogólny kryzys w dziedzinie standaryzacji oprogramowania. Z tym większym zaskoczeniem przyjąłem fakt, że niektórzy autorzy (J. Skakala, P. Vykouk, J. van Duijn) podkreślali konieczność opracowania jeśli nie norm, to przynajmniej jednolitej metodyki organizacji w działalności tak trudno poddającej się unifikacji jak automatyzacja eksperymentów naukowych. Zgadzając się całkowicie z tym poglądem należy dodać, że dotąd tylko jeden aspekt tej problematyki został jasno określony — specyfikacja. Pozostałe — wymagają przeprowadzenia dalszych, dokładniejszych analiz.

Wydaje się, że pełne rozważenie tego zagadnienia i uzyskanie wierygodnych wniosków jest możliwe tylko przy realizacji dużego projektu automatyzacji eksperymentu, gdzie system komputerowy spełnia wiele różnorodnych funkcji. Prace takie są także prowadzone, a stopień złożoności i struktura niektórych układów stawia je w rzędzie wspomnianych uprzednio zintegrowanych systemów informacyjnych (R. Friehmelt i in., DCAPC).

Spośród innych zagadnień na czoło wybijają się problemy automatycznego projektowania regulatorów (I. S. Baruh, I. G. Benkoff, W. Schaufelberger i in., DCAPC). Metody te jednak nigdy nie mogą być w pełni automatyczne, a ich skuteczność zależy w zbyt dużym stopniu od rozwoju technologii elementów elektronicznych.

Natomiast rozwój algorytmów sterowania jest nadal ogromny. Za najważniejsze uznałbym prace nad regulatorami adaptacyjnymi i samonastrajalnymi (R. Isermann, J. van Amerongen, R. K. Wood i in., DCAPC) oraz kontynuację badań nad bardziej konwencjonalnymi metodami teorii sterowania, jak sterowanie czasowo-optymalne lub sterowanie z predykcją (H. B. Verbruggen, P. M. Bruijn, DCAPC).

Reasumując, wykorzystanie komputerów do sterowania przechodzi duże przeobrażenia, wynikające przede wszystkim z rozwoju techniki mikroprocesorowej i związanego z tym rozprzestrzeniania funkcji, a zatem i samego sprzętu cyfrowego. Powstaje wiele nowych problemów wymagających szybkiego rozwiązania, jak choćby opracowanie metod programowania systemów rozłożonych przestrzennie, czy udoskonalenie metod współpracy operatora z komputerem wykonującym różnorodne funkcje pomiarowe, obliczeniowe i sterujące. Jeżeli dochodzą do tego zupełnie nowe zastosowania komputerowych systemów sterowania, jak choćby w automatyzacji laboratoriów badawczych, gdzie zarówno różnorodność funkcji jak i częstotliwość ich zmian może być znacznie większa niż w przypadku procesów przemysłowych, to należy stwierdzić, że zadania, jakie czekają automatyków i informatyków w najbliższej przyszłości, są o wiele trudniejsze niż przed kilkoma jeszcze laty, mimo niewątpliwego udoskonalenia narzędzi, jakimi się mogą obecnie posługiwać. Nakłada to znacznie większe obowiązki na szkolnictwo, o czym musimy pamiętać kształcąc specjalistów w tej dziedzinie wiedzy.

Janusz ZALEWSKI

RECENZJE

Po(d)stępy bazorządctwa

Słynne **raporty auerbachowskie**, których komplet w prenumeracie rocznej kosztuje więcej niż przeciętny samochód osobowy, w ciągu ostatnich kilkunastu lat kilkakrotnie zmieniały swój kształt, nawet gdy stały się uznanym w środowisku informatyków katalogiem encyklopedycznym sprzętu komputerowego. W świecie postępu obowiązuje bowiem bezwzględna zasada, że kto nie pcha się do przodu, ten przesuwany jest do tyłu. Zgodnie z tą zasadą były wiceprezes filadelfijskiej centrali firmy AEUERBACH, Thomas A. Rullo, podjął ambitny zamiar uruchomienia sześciu nowych serii wydawniczych dla menażerów, obejmujących najważniejsze odcinki współczesnego „frontu informatycznego" [1]).

Wieloletnie doświadczenie organizacyjne pozwoliło Tomowi Rullo nawiązać kontakty z osiemdziesięcioma wybitnymi informatykami i „wydusić" od nich we właściwym terminie 75 przekrojowych referatów.

Dziesięć z nich posłużyło do zmontowania pierwszego tomu rocznika „bazoznawczego", jak w wolnym przekładzie można oddać tytuł *ADVANCES IN DATA BASE MANAGEMENT* [2]).

Oryginalny tytuł zawiera w sobie świadomą dwuznaczność. Z jednej strony kojarzy się z systemami zarządzania bazami danych, z których istnienia współczesny menażer musi sobie zdawać sprawę. Z drugiej strony — kojarzy się z postępami w zarządzaniu przedsięwzięciami informatycznymi, czyli — nazwą wspomnianej sześcioseryjnej biblioteki dla menażerów. Jest to świetny chwyt marketingowy. Zresztą, w obecnej powodzi publikacji na ten temat baz danych przeciętny menażer po prostu się gubi,

[1]) Heyden ADVANCES Library in EDP Management, Thomas A. Rullo (Editor); obejmuje serie wydawnicze (roczniki): Advances in DATA PROCESSING Management, Advances in DATA BASE Management 1980, Advances in COMPUTER PROGRAMMING Management, Advances in DISTRIBUTED PROCESSING Management, Advances in DATA COMMUNICATIONS Management, Advances in COMPUTER SECURITY Management. Adres wydawcy: Heyden & Son Inc., 247 So. 41 st St., Philadelphia, PA 19104, USA.

[2]) RULLO Thomas A.: Advances in Data Base Management (vol. 1). Heyden Son, Philadelphia — London — Rheine, 1980, 207 str.; 57 rys.; bibl. 287 poz.; ISBN 0 85501 602 7; ISSN 0196-8718.

tu zaś ma możność nabycia jednej książki w ciągu roku i wyrobienia sobie pewnego syntetycznego poglądu na ryzyko i szanse forsowania pomysłów w tej dziedzinie, *„dopiero wychodzącej z agresywnej beztroski wieku maturalnego"*, jak można sparafrazować słowa Toma Rullo z przedmowy tomu pierwszego.

Zaletą tego tomu jest skupienie się na cyklu żywotności bazy danych — od „przedkoncepcji", poprzez sukcesywne realizacje, aż do ekspansywnej konserwacji. Zaletą T. Rullo jest bezsprzeczna ostrość spojrzenia organizatorskiego i skłonienie współpracowników do zwięzłości sformułowań oraz unikania „fasadowości".

Załączony rysunek, zaczerpnięty z otwierającej tom pracy L. E. Townera, ilustruje obrazowo tezę, że najważniejszym problemem tworzenia baz danych jest „polityka", która najbardziej obciąża „górę lodową informatyki" — zwłaszcza, gdy chodzi o wspomaganie kierownictwa, które tak często ulega syndromowi NIH [3]), a z drugiej strony — tak łatwo daje się wciągnąć w grzęzawisko systemów typu *on-line*.

Jeżeli chodzi o wady, to za główną uważam brak indeksu, chociaż dotychczas z reguły wydawnictwa typu *„Advances in..."* takowych nie prowadziły. Biblioteka heydenowska jednak jest przeznaczona dla specyficznego kręgu odbiorców (wyższa kadra kierownicza), którzy jako nieinformatycy mogą mieć poważne kłopoty z szybkim dotarciem do potrzebnych definicji. Niemniej układ poszczególnych referatów monograficznych jest bardzo usystematyzowany, toteż już po jednorazowej lekturze czytelnik może nabrać wprawy w poruszaniu się po „Postępach". Przykładowo, dla menażera nie jest ważną ścisła definicja dostępu swobodnego, ale informacja, że przy projektowaniu fizycznym taka organizacja pamięci jest zalecana przy małej aktywności bazy (powiedzmy poniżej 20%, jak proponuje T. R. Finneran). O ścisłą definicję i szczegółowe wyliczenia opłacalności menażer będzie się zwracać do specjalistów.

Przez cały pierwszy tom „Postępów" przewijają się różne rozważania „opłacalnościowe", niestety — wyłącznie w nawiązaniu do skali cen świata zachodniego. Podane sposoby rozliczania nakładów stanowią jednak wyraźną wskazówkę co do struktury efektów i mogą stać się przestrogą przed projektanckim hurra-optymizmem. Monograficzność ujęcia problematyki baz danych zmusza wprost czytelnika-menażera do uzmysłowienia sobie, że istnieją różne możliwości rozwiązań na każdym etapie cyklu rozwojowego.

Dla mnie najważniejszym rozdziałem w „Postępach" jest chyba praca małżonków Wiórkowskich z Dallas w Teksasie — udokumentowana ponad setką pozycji bibliograficznych, ze słynnym raportem Nolana o „kryzysie informatycznym" na czele. Otóż Wiórkowscy podkreślają, że wprawdzie ok. połowy komputerów dużych i średnich posiada już obecnie wdrożone systemy zarządzania bazami danych, a w rozwoju perspektywicznym informatyki docelowo ok. 60% łącznej mocy obliczeniowej będzie dotyczyć baz danych, to jednak w pełni zintegrowane systemy baz danych nadal są „niedojrzałe". Uwagi te są syntezą licznych ankiet przeprowadzonych wśród użytkowników baz danych. Dla wielu działaczy okażą się zapewne orzeźwiającym kubłem zimnej wody.

Advances in Data Base Management nie są przewodnikiem po najnowszych osiągnięciach w dziedzinie projektowania baz danych i wdrażania systemów zarządzania bazami zintegrowanymi. Oczywiście, projektanci i wdrożeniowcy mogą wiele skorzystać na lekturze „Postępów", ale na zasadzie oglądania swego zawodowego podwórka oczami osoby z zewnątrz. Tego rodzaju książki powinny znaleźć się w każdym większym ośrodku obliczeniowym, jest to jednak literatura uzupełniająca, nie zastępująca szczegółowych podręczników i dokumentacji oprogramowania. Postępy nie dają też cenzurek typu „która baza jest najlepsza?" — byłyby to bowiem niedopuszczalne substytuty indywidualnych analiz. Dlatego też współautorzy dzieła konsekwentnie, a jest to chyba wyraz koncepcji edytorskiej Thomasa A. Rullo, koncentrowali się na ujęciu teleologicznym. Zamiast przytaczania motywacji negatywnych w cytowanych przykładach — uwypuklano przede wszystkim cele rozwojowe.

Adam B. EMPACHER

[3]) ang. *Not-Invented-Here* (dosłownie: nie tutaj wymyślone).

O pisowni nazw języków programowania

Sądzę, że przed rozpowszechnieniem nowego języka programowania, jakim jest Ada (p. J. Zalewski, Ada — nowy język programowania, s. 10), warto uzasadnić, poprawną pisownię tej nazwy w języku polskim. Przed tym jednak należałoby się głębiej zastanowić nad pisownią nazw innych języków.

Od dawna wydaje mi się, że pisanie dużymi literami niektórych nazw języków programowania nie jest dostatecznie uzasadnione, nawet jeśli są to skróty. Wynikiem braku jednoznacznych ustaleń jest występowanie podwójnej pisowni — ALGOL i Algol, FORTRAN i Fortran, itp. Przyjmując, że w języku polskim nazwy języków programowania są nazwami własnymi, wystarczy używać jednej formy, tj. takiej, jaka ogólnie obowiązuje w przypadku nazw własnych i polega na rozpoczynaniu tej nazwy dużą literą. Nikt przecież, pisząc tekst o Wiśle nie pisze WISŁA, czy o Warszawie — WARSZAWA.

Regułą tą można także objąć akronimy będące nazwami własnymi i wtedy będą jej podlegać także nazwy języków programowania. Oczywiście, skrótów będących nazwami własnymi, złożonych z pierwszych liter, nie można pisać inaczej niż dużymi literami, np. PL/I, APL, PDP-11.

W przypadku języka Ada sytuacja jest zupełnie jasna. Ponieważ sama nazwa pochodzi od imienia „programistki" Charlesa Babbage'a (uważanej za pierwszą), Ady Augusty Byron, zatem — zgodnie z typową pisownią imion — nie ma podstaw, aby nazwę Ada pisać inaczej. Tak zresztą używana jest ta nazwa w podręczniku języka (Reference Manual for the Ada Programming Language, US Departament of Defense, Washington, DC, July 1980) i taka będzie — w przygotowywanej normie międzynarodowej. Zatem użycie pisowni ADA, zamiast Ada, należy w języku polskim uznać za błąd.

Poprawny sposób pisania nazw języków programowania można uzasadnić jeszcze inaczej. W tym celu należy zastanowić się, w jaki sposób powstały nazwy języków, takich jak: Algol, Fortran, Cobol itp. Jest powszechnie wiadomo, że nazwa Algol stanowi skrót od angielskiego określenia *algorithmic language*, podobnie Fortran — od *formula translator*, Cobol — od *common business oriented language*. Zatem proponowane akronimy utworzone od wymienionych określeń są następujące: algol, fortran, cobol. Ponieważ wiadomo, że te akronimy oznaczają języki programowania, a więc są nazwami własnymi, należy je rozpoczynać od dużej litery — tak, jak inne nazwy własne.

Inaczej natomiast, wygląda sprawa ze skrótami, takimi jak APL, PDP-11 i in. Nazwy te należy pisać dużymi literami, ponieważ stanowią skróty innych nazw własnych. Przykładowo, nazwa A Programming Langugage jest sama nazwą odpowiedniego języka programowania, a więc — nazwą własną (tak jak nazwa Niemiecka Republika Demokratyczna i jej skrót NRD są nazwami własnymi). Jeżeli natomiast uznalibyśmy, że nazwy Formula Translator lub Algorithmic Language są nazwami własnymi, to jeszcze nie stanowi podstawy do stosowania pisowni: FORTRAN i ALGOL.

Zasadę, która mówi, że nazwy własne (także te, będące akronimami) należy tylko rozpoczynać dużą literą, można zastosować również do nazw komputerów z firm produkujących komputery. Wtedy okaże się, że szereg nazw jest pisanych niewłaściwie. Przykładowo, należałoby pisać: Mera 400, a nie MERA 400; Cyber 200, a nie CYBER 200 itd. W tym jednak wypadku, w przedstawionym kryterium poprawnej pisowni należałoby jeszcze uwzględnić fakt, że wymienione nazwy są nazwami handlowymi. Stanowiłoby to, być może, uzasadnienie ich obecnej pisowni.

Trzeba dodać, że w niektórych wypadkach przyjęło się pisanie dużymi literami nazw własnych nie będących skrótami. Przykładowo na tablicach informacyjnych pisze się WARSZAWA i WISŁA, a nie Warszawa lub Wisła. Gdyby odnieść tę uwagę do pisowni nazw języków programowania, to można zaakceptować nazwy złożone z dużych liter, np. na stronach tytułowych książek, podręczników, kart programowych itp. W takiej formie nie jest to w sprzeczności z przedstawioną wcześniej zasadą pisowni nazw, choć należałoby zadbać, aby w tekstach nazwy te były pisane poprawnie.

Omawiane zagadnienie jest zresztą znacznie szersze i dotyczy innych rodzajów nazw, np. nazw czasopism. Nazwa miesięcznika „Informatyka" jest pisana także dużymi literami, choć nie ma to uzasadnienia w przedstawionej regule.

Na zakończenie warto rozstrzygnąć kwestię pisowni nazwy PL/I czy PL/1. Ponieważ normy źródłowe — ANSI X3.53 (1976), a za nią ISO 6160 (1978) dotyczą języka PL/I, nie ma wątpliwości, że tylko taka pisownia jest poprawna.

Janusz ZALEWSKI

Od Redakcji

Nazwy własne czasopism, firm, urządzeń, języków, pakietów programowych itp. piszemy dużymi literami, aby — podobnie jak ma to miejsce w przypadku tablic informacyjnych — wyróżnić je w tekście i tym samym ułatwić Czytelnikowi szybsze wyszukanie interesujących go informacji. Autor uznając handlowe uzasadnienie stosowania dużych liter powinien uznać również zasadność przyjętej przez nas konwencji zapisu.

LISTY

Szanowna Redakcjo!

Chciałbym i ja dorzucić swoje „trzy grosze" do dyskusji o naszej informatyce. Prawie wszyscy zgadzamy się, że jest źle, znajdujemy różne przyczyny tej sytuacji, a także kierunki dalszych działań. Niestety — jak do tej pory — odbywa się to wszystko na zasadzie „nocnej Polaków rozmowy". Dlatego zamiast określenia własnej diagnozy i sposobu leczenia wnioskuję, ażeby INFORMATYKA, która dociera do całego, niebywale rozdrobnionego środowiska, podjęła się zadania utworzenia społecznej komisji. Gremium to na bazie toczącej się dyskusji — zarówno na łamach INFORMATYKI, jak i w organizacjach społecznych i politycznych — opracowałoby całościowy raport o stanie informatyki oraz kierunkach i uwarunkowaniach jej rozwoju na najbliższe 5—10 lat.

W wyniku konfrontacji (raportu społecznego i opracowanego już raportu rządowego — powinien powstać kompleksowy program rozwoju i organizacji informatyki w Polsce. Celem zapobieżenia ewentualnym podmiotowym zmianom oraz zagwarantowania jego realizacji, należy dążyć do uchwalenia go przez Sejm PRL.

Ryszard BABIARZ
Lublin

Od Redakcji

Powyższą propozycję uznajemy za całkowicie uzasadnioną. Nie widzimy jednak możliwości opracowania raportu społecznego siłami Redakcji czy choćby — w oparciu o nie. Przygotowaniem tego raportu zajmuje się Polskie Towarzystwo Informatyczne.

Infologiczne problemy przetwarzania danych

Krótki, lecz nader interesujący list Adama Jagły z Poznania (INFORMATYKA nr 7—8/1981) zachęcił mnie do podjęcia próby uogólnienia myśli, jakie zostały w nim zawarte. Można w nich odczytać chęć zwrócenia uwagi specjalistów zajmujących się organizacją i realizacją przetwarzania danych na informację — jako zasadniczy obiekt tych procesów, „surowiec", bez którego owe procesy tracą wszelki sens.

Podzielam pogląd, iż wielu informatyków tak fascynuje się sprzętem komputerowym i jego uruchomieniem, że zaczyna zupełnie tracić z pola widzenia to, co musi być wzięte pod uwagę w pierwszej kolejności: informacje, które mają być opracowywane za pomocą tego sprzętu. Prawda, nowoczesny sprzęt komputerowy może rzeczywiście zafascynować specjalistów wprost niezwykłymi możliwościami. Ale trzeba pamiętać, że rozwiązanie ciekawych, związanych z nim problemów w istocie będzie miało sens tylko wtedy, kiedy posłuży opracowaniu i udostępnieniu informacji, które są komuś potrzebne.

Niemniej p. Jagła w swojej wypowiedzi odwraca sprawę: nawołuje do współpracy nad powszechnym ujednoliceniem „wszelkich kodów, począwszy od materiałowego, a skończywszy na symbolach Polskich Norm". Nie zastanawia się przy tym nad zasadnością takiego ujednolicenia. To prawda, że ujednolicone kody przyczyniłyby się do zwiększenia spójności różnych systemów i rozwiązań informatycznych. Ale czy jest to zawsze możliwe i celowe? Przecież jeśli kody są bardzo rozbudowane, nie tylko nie ułatwiają, lecz przeciwnie — zaczynają utrudniać przetwarzanie. Mogą na przykład grozić pojawianiem się dużej liczby różnych pomyłek, które będą występować z tym większym prawdopodobieństwem, im bardziej kod będzie wydłużany. A jak sprawić, by kod materiałowy (lub inny podobny) był krótki i służył wszystkim użytkownikom w kraju, skoro jeden z nich będzie chciał go wykorzystać przede wszystkim do identyfikowania materiału ze względu na proces jego wytwarzania, zaś inny — ze względu na walory użytkowe? Statystyka państwowa inaczej rozpatruje takie materiały, a inaczej magazynier, który musi przyjąć do magazynu nie jakąś grupę towarową, lecz konkretny wyrób. W takich warunkach albo kod będzie bardzo długi, albo pominięte zostaną potrzeby niektórych jego użytkowników.

Najczęściej informacje odsuwane są na plan dalszy, akcentuje się zaś technologię i organizację przetwarzania. Tymczasem słuszniej byłoby najpierw rozważyć, co ma być chronione, przed kim i dlaczego, jak odróżnić dane w pełni zasługujące na zaufanie od danych obciążonych błędami, jak opracować kody, by zapewniały one zgodność z potrzebami użytkowników i możliwość jednoznacznego, prostego odczytania reprezentowanych przez nie informacji. Są to bowiem sprawy zasadnicze, od których wypada rozpoczynać prace nad nowymi technologiami przetwarzania, nad nowymi systemami informatycznymi.

W istocie problemy tego rodzaju wyłaniają się zawsze wtedy, kiedy tylko przystępujemy do zbierania, przechowywania i opracowywania informacji. Występowały one jeszcze przed erą techniki komputerowej, występują i obecnie — wtedy, kiedy do opracowania wyników zamierzamy wykorzystać duży komputer, jak i wówczas, gdy wystarczą do tego środki skromniejsze. Nie sprzęt bowiem je rodzi, lecz sam fakt operowania informacjami.

Problemy tego rodzaju nazwiemy problemami infologicznymi. Terminem tym podkreślamy ich niezależność od technologii i organizacji przetwarzania, od stosowanego sprzętu technicznego. Jak dotąd, nie były one przedmiotem odrębnych badań i analiz. Mówiło się o nich zazwyczaj przy okazji omawiania innych spraw — problemów budowania systemów informatycznych, projektowania, programowania komputerów itp. I to najczęściej bez bliższej analizy ich istoty. A przecież zasługują na to, aby stać się obiektem odrębnych studiów i szkoleń.

Problemów zaś jest niemało; choćby następujące:

- określenie zakresu i rodzaju informacji, które powinny być zebrane i opracowane, aby użytkownik mógł otrzymać wyniki, jakie są mu potrzebne
- określenie ich logicznej struktury, odpowiadającej potrzebom użytkowników i zgodnej z ich punktem widzenia zjawisk i zdarzeń, jakie są opisywane przez rozpatrywane informacje
- ustalenie należytej jakości zbieranych i opracowywanych informacji: ich rzetelności, dokładności, prawdziwości, kompletności, terminowości
- ustalenie granic i możliwości wykorzystania komputera do automatycznego wykrywania błędów w danych wejściowych oraz ich automatycznego korygowania, w tym także w warunkach banków danych
- zapewnienie spójności rozpatrywanych informacji (na przykład przetwarzanych w danym systemie informatycznym) z innymi informacjami, z których użytkownik będzie także korzystać
- ustalenie potrzeb ochrony zbieranych informacji przed ich niewłaściwym wykorzystaniem
- ustalenie efektywności rozpatrywanego rozwiązania informatycznego, rozumianej jako stopień potrzeb informacyjnych określonych użytkowników.

Lista ta nie jest kompletna. Życie będzie dopisywać do niej nowe problemy, rodzące się w miarę rozwoju procesów przetwarzania danych i naszego poznawania własności informacji. Ale już ta niepełna lista wskazuje na złożoność i różnorodność problemów infologicznych. Wypada zatem zachęcić projektantów systemów informatycznych i banków danych, projektantów wszelkich systemów przetwarzania danych, programistów i innych potencjalnie zainteresowanych specjalistów, aby problemów infologicznych nie tracili z pola widzenia i kręgu swych zainteresowań, oraz poświęcili im tyle uwagi i czasu, na ile zasługują.

Bogdan STEFANOWICZ

Bibliografia wydawnictw polskich z dziedziny informatyki

● **System Selektywnej Dystrybucji Informacji na mc serii RIAD SDI/R — Centrum Komputerowych Systemów Automatyki i Pomiarów MERA-ELWRO, Wyd. Politechniki Wrocławskiej, Wrocław 1980, s. 28**

Przedmiot, zakres i główne funkcje systemu. Struktura, opis i efekty wdrożenia systemu. Dokumenty źródłowe. Tabulogramy użytkowe. Techniczne środki eksploatacji. Warunki wdrożenia. Zakres dostawy. Materiały przeznaczone są dla użytkowników systemu wyszukiwania danych bibliograficznych.

● **Wytyczne do tworzenia komputerowych modeli planowania zysku, tłum. wyd. ang. z 1979 r. Wyd. Zjednoczenia Informatyki. Centrum Projektowania i Zastosowań Informatyki, Warszawa 1980, s. 86, cena 126 zł**
Europejski Program Badawczy Diebolda. Zeszyt (E — 173)

Model przedsiębiorstwa. Planowanie zysku — szersze implikacje. Wytyczne do skutecznej realizacji modeli. Korzystanie z usług zewnętrznych przy modelowaniu dla potrzeb naczelnego kierownictwa finansowego. Modelowanie a zarządzanie zasobami informacyjnymi. Materiały przeznaczone są dla kierownictwa i służb finansowych przedsiębiorstw.

● **Podstawy i praktyka programowania mikroprocesorów — GRABOWSKI J., KOŚLACZ S., WNT, Warszawa 1980, s. 336, cena 70 zł**

Opis funkcjonalny struktury i rozkazów mikroprocesora. Otoczenie mikroprocesora. Asembler i asemblacja. Asemblery strukturalne. Niektóre techniki programowania. Proces uruchamiania programu. Systemy wieloprocesorowe i wielokomputerowe. Zastosowanie mikroprocesorów. Dodatki.
Książka przeznaczona jest dla inżynierów, elektroników oraz programistów zajmujących się problemami programowania układów mikrokomputerowych.

● **Organizacja systemów informatycznych w transporcie samochodowym — KOLBUSZ E., WKiŁ, Warszawa 1980, s. 235, cena 40 zł**

Podstawy organizacji przetwarzania danych. Problemy projektowania systemów informatycznych. Przedsiębiorstwo transportu samochodowego jako obiekt zastosowań informatyki. Struktura funkcjonalna systemu. Informacyjne zasilania wyjściowe systemu. Baza normatywna systemu. Informacyjne zasilanie wejściowe systemu. Funkcjonowanie sytemu przy zastosowaniu technik konwencjonalnych. Organizacja bazy danych w systemie informatycznym przedsiębiorstwa transportu samochodowego. Problemy realizacji systemu.
Materiały przeznaczone są dla pracowników służb ekonomicznych i eksploatacyjnych przedsiębiorstw transportu samochodowego.

● **Organizacja maszyn cyfrowych w ujęciu strukturalnym — TANENBAUM A. S. Tłum. wyd. ang. z 1976 r., PWN, Warszawa 1980, s. 517, cena 120 zł**

Organizacja maszyn cyfrowych. Konwencjonalny poziom maszynowy. Poziom mikroprogramowania, systemu operacyjnego maszyny i języka asemblera. Maszyny wielopoziomowe. Sugestie dotyczące dalszej lektury i wykaz literatury. Dodatki i Arytmetyka o skończonej precyzji i liczby dwójkowe. Liczby zmiennopozycyjne. Algebra Boole'a.
Książka ta ma służyć jako podręcznik do wstępnego wykładu z programowania w języku asemblera i organizacji maszyn cyfrowych. Do jej zrozumienia nie jest potrzebne specjalne przygotowanie matematyczne ani techniczne. Materiały przeznaczone są dla studentów wydziałów informatyki i elektroniki oraz programistów i użytkowników maszyn cyfrowych.

● **Metody planowania i oceny zastosowań APD. Tłum. wyd. ang. z 1979 r., Wyd. Zjednoczenia Informatyki — Centrum Projektowania i Zastosowań Informatyki, Warszawa 1980, s. 72, cena 126 zł. Europejski Program Badawczy Diebolda. Zeszyt 118 (E 182)**

Środowisko planowania. Konwencjonalne metody planowania i oceny. Nowe metody planowania i oceny. Metodyka SAPE (Structured Application Planning and Evaluation) Załącznik: Lista pytań kontrolnych SAPE.
Opracowanie jest przeznaczone dla kierownictwa wszelkich szczebli, które użytkuje lub planuje środki APD.

● **Minikomputery w procesie dydaktycznym — CWIRKO-GODYCKI J., Wydawnictwa Szkolne i Pedagogiczne, Warszawa 1980, s. 194, cena 89 zł**

Minikomputer Papy'ego. Minikomputer dwójkowy. Minikomputer trójkowy. Minikomputer Whitneya. Minikomputery w procesie dydaktycznym.
Materiały przeznaczone są dla nauczycieli i uczniów szkół podstawowych (w szczególności działających w kółkach matematycznych) zgodnie z zmodernizowanym programem matematyki.

● **Opis języka FORTRAN-20 — WIGORA A. M. i inni, Wyd. Czasopism i Książek Technicznych SIGMA, Warszawa 1980, s. 355**

Cz. 1. Wstęp do FORTRAN-20. Elementy instrukcji. Instrukcja podstawienia. Instrukcje sterujące. Instrukcje wejścia-wyjścia. Instrukcje FORMAT. Instrukcje specyfiki. Funkcje i procedury. Segment, program, wsad. Dyrektywy pomocnicze.
Cz. 2. Obsługa operatorska programu. Obsługa sytuacji wyjątkowych. Operacje na plikach. Śledzenie programu.
Cz. 3. Kompilacja programu. Plik listengu, sygnalizacja błędów. Programowanie wielojęzykowe. System plików PAM. Dodatki.
Materiały przeznaczone są dla programistów.

● **Materiały pomocnicze do nauczania podstaw informatyki (tekst programowany) — FONFARA M., GROSZAK R., SZAJBA E., Wyd. Akademii Ekonomicznej w Poznaniu, Poznań 1980, s. 96, cena 10 zł.**
Skrypty uczelniane. Zeszyt nr 277

Wskaźniki metodologiczne. Cz. 1. Arytmetyka maszyn cyfrowych. Cz. 2. Algorytmy i schematy blokowe.
Materiały przeznaczone są dla studentów wyższych uczelni ekonomicznych.

● **Elementy elektronicznej techniki obliczeniowej, metod numerycznych i programowania matematycznego — WOSIEWICZ B., Wyd. Akademii Rolniczej w Poznaniu, Poznań 1980, s. 243, cena 30 zł**
Cz. I. Elektroniczna technika obliczeniowa: Maszyny matematyczne. Arytmetyczne i logiczne podstawy działania maszyn cyfrowych (mc). Bloki funkcjonalne mc. Programowanie mc. Algorytmy i ich zapis. ALGOL-60. Systemy użytkowania mc. Kalkulatory programowane. Zastosowanie elektronicznej techniki obliczeniowej w budownictwie wodnym i melioracjach.
Cz. 2. Metody numeryczne: Macierze. Rozwiązywanie układów równań liniowych. Interpelacja i aproksymacja. Całkowanie numeryczne. Rozwiązywanie równań nieliniowych i różniczkowych.
Cz. 3. Programowanie liniowe i sieciowe.
Skrypt jest przeznaczony dla studentów Wydziału Melioracji Wodnych Akademii Rolniczej w Poznaniu.

● **Układy cyfrowe i mikroprocesory. Skrypt dla studentów II i III roku Wydziału Elektroniki — BARSKI M., JĘDRUCH W., Wyd. Politechniki Gdańskiej, Gdańsk 1980, s. 299 cena 36 zł**

Wiadomości podstawowe. Sposoby przedstawienia informacji w układach cyfrowych. Dwuelementowa algebra Boole'a. Synteza układów kombinacyjnych. Synteza układów sekwencyjnych. Realizacja techniczna układów logicznych. Typowe układy cyfrowe. Mikroprocesory.

● **Mikroprocesory seryjne — WAGNER F., Wyd. Wyższej Szkoły Inżynierskiej im. Jurija Gagarina w Zielonej Górze, Zielona Góra 1980, s. 134**

Cz. I. System mikrokomputerowy: Jednostka centralna mikrokomputera. Pamięci w systemie mikrokomputerowym. Współpraca systemu mikrokomputerowego z urządzeniami zewnętrznymi (peryferyjnymi). Programowanie mikrokomputerów.
Cz. 2. System 8080A, Jednostka Centralna 8080A. Rejestry interfejsowe systemu. Podłączenie urządzeń realizujących transmisję szeregową. Obsługa przerwań. Realizacja przesyłania z bezpośrednim dostępem do pamięci (DMA). Odmierzanie czasu i liczenie impulsów. Skrypt przeznaczony jest dla studentów wydziałów elektronicznych wyższych uczelni technicznych.

Oprac. A. K.

Isotimpex

ELEKTRONICZNA KASA REJESTRUJĄCA ELKA 90

Elektroniczna kasa rejestrująca ELKA 90 jest terminalem kasowym z możliwością automatycznego wprowadzania danych z etykiet towarowych. Jest ona przeznaczona zarówno dla dużych, jak i małych sklepów, domów towarowych, restauracji, magazynów itp. placówek handlowych i usługowych.

Charakterystyka podstawowych możliwości funkcjonalnych:

- rejestry podstawowe 5
- rejestry stornujące 5
- rejestry dla kelnerów z dostępem poprzez klucz kodowy 7
- rejestry pomocnicze dla kwot wprowadzanych 1
- rejestry pomocnicze dla kwot kasowanych 1
- rejestry dla różnych sposobów płatności (gotówka, czeki, kupony, kredyt) 4

Każdy z powyższych rejestrów ma odrębny 4-cyfrowy licznik służący do sumowania liczby dokonanych transakcji.

- rejestry wyrobów określane przez wprowadzany symbol kodowy 76

Pojemność powyższych rejestrów – 8 cyfr.

- rejestry programowane:
  - procent
  - data
  - symbol waluty
- liczniki kontrolne:
  - liczby rachunków – 4-cyfrowy
  - kontroli zerowej – 4-cyfrowy.

Podstawowe operacje:

zapamiętywanie, kasowanie, mnożenie, zwiększanie o procent, zmniejszanie o procent, korygowanie błędnego zapamiętania, zerowanie, tworzenie sumy łącznej, czeki kredytowe, kupony. ELKA 90 ma dwa tryby wprowadzania danych: ręczny oraz automatyczny – za pomocą przenośnego czytnika etykiet towarowych.

Tryb pracy ręcznej:

- wprowadzanie symbolu wyrobu – do 11 cyfr
- wartość – do 7 cyfr
- ilość – do 5 cyfr

Tryb pracy automatycznej:

wprowadzanie danych odbywa się z etykiet towarowych zakodowanych magnetycznie przy użyciu czytnika z podawaniem ręcznym. Maksymalna liczba zakodowanych na etykiecie symboli – 45 (symbol identyfikacyjny – do 11 cyfr, cena – do 7 cyfr, dodatkowe cechy identyfikujące, takie jak wysokość, rozmiar, kolor – do 24 cyfr). Rozróżnialne w sposób wizualny informacje dotyczące ceny i symbolu identyfikacyjnego są wprowadzane na nośnik papierowy.

Czytnik magnetyczny:

- szybkość odczytu 8 ÷ 80 cm/s
- kierunek przesuwu etykiety – do przodu i wstecz
- długość przewodu łączącego – do 2 m

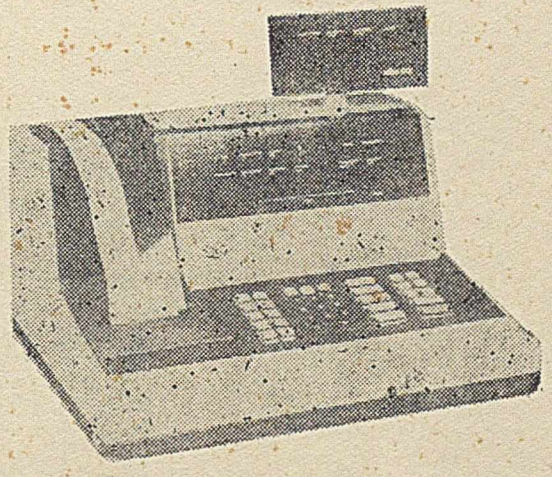
Dane wprowadzone prawidłowo są potwierdzane krótkim sygnałem dźwiękowym.

ELKA 90 ma 2 tryby operacji:

- autonomiczny
- systemowy, który zapewnia przyłączenie za pomocą 4-żyłowego interfejsu do rejestratora cyfrowego albo do koncentratora danych, działającego w ramach systemu transmisji danych.

Po zarejestrowaniu każdej transakcji ELKA 90 podaje następujące dane:

cenę jednostkową, ilość oraz identyfikator wyrobu, symbol waluty, typ operacji, dodatkowe wprowadzone poprzez czytnik etykiet cechy identyfikujące oraz symbole kontrolne.



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

EO/403/K/81-B

robotron

Automat księgująco-fakturujący
Księgowanie Fakturowanie Gospodarka materiałowa

Gdzież można znaleźć takiego księgowego? Takiego, który pracuje zawsze z jednakową precyzją, szybkością i obiektywnością, który jest specjalistą od wszystkiego i któremu można jeszcze dołożyć trochę pracy?
Właściwa odpowiedź: automat księgująco-fakturujący A 5110.
Będzie Was wspierał przy rozwiązywaniu wielu problemów.
A oto bliższe dane:
Na podstawie wewnętrznego systemu działania następuje programowanie poszczególnych programów standardowych, aż do makroprogramu. Pojemność pamięci sięga przy tym 36 K bajt.
Model standardowy automatu księgująco-fakturującego daje możliwość wielu zmian i różnych zestawień. Jego specjalnością jest księgowanie i fakturowanie.
Co A 5110 oferuje w zakresie oprogramowania?
Moduły problemowe, uwzględniające specyfikę danego kraju, generator programowy do łączenia modułów, jak również stosowane w danym kraju programy standardowe oraz wiele indywidualnych rozwiązań. Bez obaw można mu zlecić rozwiązywanie problemów z zakresu fakturowania, księgowości, gospodarki materiałowej, kalkulacji wynagrodzeń i płac.
Nasz A 5110 może być także stosowany do opracowywania zleceń przy remontach pojazdów samochodowych lub rozliczania sprzedaży. Sporządza rachunki, księgi rachunkowe, spisy inwentarza, statystyk lub upomnień.

Zapraszamy do obejrzenia Waszego nowego księgowego.

robotron

Robotron Export-Import
Państwowe Przedsiębiorstwo
Handlu Zagranicznego Niemieckiej Republiki
Demokratycznej
NRD 1080 Berlin, Friedrichstrasse 61

Automat księgująco-fakturujący

A 5110