1 1982

informatyka

WYDAWNICTWO NOT SIGMA

Od Wydawcy

Po przeszło trzech miesiącach przerwy wznawiamy wydawanie czasopism technicznych. Kierujemy do naszych Czytelników pierwsze tegoroczne numery (kwietniowe) po zmienionych cenach. Ich wzrost do wartości obowiązującej w 1982 r. jest pochodną nowych cen papieru i usług poligraficznych różniących się od cen poprzednich nawet 3,5-krotnie.

Nieukazywanie się tytułów SIGMY w I kwartale 1982 r., a także — w niektórych przypadkach — nieukazanie się numerów z roku 1981 zobowiązuje miejscowe przedsiębiorstwa Kolportażu Prasy i Wydawnictw RSW „Prasa-Książka-Ruch" do zwrotu prenumeratorom odnośnej części przedpłaty.

Wznawiając wydawanie czasopism technicznych liczymy — jak dotychczas — na rozwijającą się współpracę z naszymi Czytelnikami i Autorami.

Wydawnictwo
NOT-SIGMA

Od Redakcji

Wznawiamy działalność. INFORMATYKA ma się teraz ukazywać co miesiąc, w zapowiadanej poprzednio objętości 32 kolumn. Nie zmienił się skład redakcji, nie zmieniły się też nasze założenia i plany. Zmieniły się zaś — niestety — warunki, w których mamy je realizować.

Perspektywy nie są optymistyczne. Obecna sytuacja nie sprzyja rozwojowi informatyki. Nie możemy jednak naszych łamów zmienić w c z a r ę g o r y c z y. Przyjmujemy więc perspektywę przyszłości — kraju z rozsądnie zorganizowaną gospodarką, kraju, w którym nastąpi wreszcie czas b u d o w a n i a.

Informatyka jest szczególnie efektywnym narzędziem społeczeństwa rządzonego racjonalnymi metodami. Wprzęgnięta w system irracjonalny traci rację bytu. Dlatego też naszym punktem odniesienia musi być przyszłość — oczekiwana i konieczna.

INFORMATYKA

NACZELNA ORGANIZACJA TECHNICZNA

WYDAWNICTWO SIGMA

CZASOPISM I KSIĄŻEK TECHNICZNYCH

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

Redaktor naczelny: prof. dr hab. Leon ŁUKASZEWICZ

prof. dr hab. inż. Konrad FIAŁKOWSKI (zastępca redaktora naczelnego), mgr inż. Zbigniew GLUZA, dr Janusz GWIAZDA, Władysław KLEPACZ (zastępca redaktora naczelnego), dr inż. Tomasz PAWLAK, dr inż. Janusz ZALEWSKI
Sekretarz redakcji: mgr Teresa JABŁOŃSKA

RADA PROGRAMOWA

Prof. dr hab. Tadeusz PECHE (przewodniczący), mgr inż. Tomasz BAŃKOWSKI (sekretarz), mgr inż. Antoni BOSSOWSKI, mgr inż. Roman BURNO, prof. dr hab. Andrzej JANICKI, mgr inż. Jan KRAMARCZUK, prof. dr hab. inż. Juliusz KULIKOWSKI, prof. dr hab. Leon ŁUKASZEWICZ, prof. dr hab. Antoni MAZURKIEWICZ, gen. dr inż. Marian PASTERNAK, mgr inż. Bronisław PIWOWAR, mgr Zbigniew SUBSTYK, doc. dr hab. Tadeusz WALCZAK

Materiałów nie zamówionych Redakcja nie zwraca.

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

Zakł. Graf. „Tamka". Zam. 385. Obj. 4,0 ark. druk. Nakład 5000 egz. Z-60.

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

Walczak T.: Informatyka w okresie przemian
INFORMATYKA 1982, nr 1, s. 4

Charakterystyka narastania zjawisk kryzysowych w dotychczasowym rozwoju polskiej informatyki, w tym również kryzysu społecznego zaufania. Na podstawie analizy przyczyn tych zjawisk autor uzasadnia, że wprowadzenie w kraju reformy gospodarczej spowodować musi również wzrost efektywności zastosowań informatyki oraz ich prawidłowe ukierunkowanie na rzeczywiste potrzeby przedsiębiorstw i społeczeństwa.

Вальчак Т.: Вычислительная техника в период перемен

ИНФОРМАТИКА 1982, № 1, стр. 4

Характеристика нарастания кризисных явлений в современном развитии польской вычислительной техники, в том числе общественного кризиса доверия к ней. На основе анализа причин этих явлений автор доказывает, что проведение в стране хозяйственной реформы должно также вызвать повышение эффективности применения вычислительной техники и ее правильную ориентацию на подлинные нужды предприятий и общества.

Fryń R.: Doświadczenia i problemy projektowania aplikacyjnego w FSO
INFORMATYKA 1982, nr 1, s. 7

Charakterystyka przedsięwzięć organizacyjnych wprowadzonych w Fabryce Samochodów Osobowych w Warszawie w celu zwiększenia wydajności programowania systemów informatycznych. Podano przykłady rozwiązań oraz efekty ich zastosowania.

Фрынь Р.: Опыты и проблемы апликационного проектирования в Фабрике легковых автомобилей (ФСО)

ИНФОРМАТИКА 1982, № 1, стр. 7

Характеристика организационных предпринятий введенных в Фабрике легковых автомобилей в Варшаве с целью повышения производительности программирования вычислительных систем. Указываются примеры решений и эффекты их применения.

Kapuścik W.: Eksploatacja baz danych w hucie „Szopienice"
INFORMATYKA 1982, nr 1, s. 11

Charakterystyka rozwiązań oraz prezentacja doświadczeń eksploatacji baz danych w systemach informatycznych huty metali nieżelaznych „Szopienice", zrealizowanych w oparciu o sprzęt i oprogramowanie firmy UNIVAC.

Капусьцик В.: Эксплуатация баз данных в металлургическом заводе „Шопенице"

ИНФОРМАТИКА 1982, № 1, стр. 11

Характеристика решений и представление опытов эксплуатации баз данных в вычислительных системах металлургического завода цветных металлов „Шопенице", реализованных на основе оборудования и программного обеспечения фирмы UNIVAC.

Zalewski J.: ADA — nowy język programowania. Cz. 2. Jednostki programowe i instrukcje
INFORMATYKA 1982, nr 1, s. 15

Druga część prezentacji Języka ADA. Podano charakterystykę podstawowych elementów tego języka, jakimi są tzw. jednostki programów, oraz stosowanych w tych jednostkach instrukcji.

Залевский Я.: АДА — новый язык программирования
(2) Программные единицы и инструкции

ИНФОРМАТИКА 1982, № 1, стр. 15

Вторая часть представления языка АДА. Дается характеристика основных элементов этого языка, которыми являются так называемые единицы программ, а также применяемые в этих единицах инструкции.

Walczak T.: Data processing in the period of changes
INFORMATYKA 1982, No 1, p. 4

Characteristics of the crisis occurrence increase in the past development of polish data processing, as well as of the social confidence crisis. Analysing reasons of these occurrences the author motivates, that the introduction of the economic reform must also result in the effectiveness increase of data processing applications and their proper orientation on the actual business and society needs.

Walczak T.: Datenverarbeitung in der Periode der Umwandlungen
INFORMATYKA 1982, Nr. 1, S. 4

Eine Charakteristik des Anwachsens von Kriseerscheinungen in der bisherigen Entwicklung der polnischen Datenverarbeitung, sowie auch der Krise des gesellschaftlichen Vertrauens. Auf Grund der Ursachenanalyse begründet der Autor, dass die Einführung der Wirtschaftsreform auch die Effektivitätssteigerung der Datenverarbeitunganwendungen und ihre rechtmässige Orientierung auf wirkliche Bedürfnisse der Unternehmungen und der Gesellschaft beeinflussen wird.

Fryń R.: Experience and problems of application designing in FSO
INFOMRATYKA 1982, No 1, p. 7

Characteristics of organizational undertakings applied in the FSO motor-car factory in Warsaw for productivity enhancement of systems programming are given. Solutions, examples and application effects are presented.

Fryń R.: Erfahrungen und Probleme der Anwendungsprojektierung im FSO
NIFORMATYKA 1982, Nr. 1, S. 7

Eine Charakteristik der organisatorischen Massnahmen, die in den FSO-Personenkraftwagenwerken zweks Erhöhung der Programmierungsleistung von EDV-System eingeführt werden. Es wurden Beispiele der Lösungen und die Anwendungseffekte angegeben.

Kapuścik W.: Operation of data bases in the smelting works „Szopienice"
INFORMATYKA 1982, No 1, p. 11

Charakteristics of solutions and presentation of data bases operation experience in the data processing systems of the non-ferrous smelting works „Szopienice" are given. The systems are realized using UNIVAC hardware and software.

Kapuścik W.: Betrieb von Datenbasen im Hüttenwerk „Szopienice"
INFORMATYKA 1982, Nr. 1, S. 11

Eine Charakteristik der Lösungen und Vorstellung der Betriebserfahrungen mit Datenbasen in EDV-Systemen der Buntmetallenhüttenwerk „Szopienice". Die Lösungen werden mit Ausnutzung von Hard- und Software der Firma UNIVAC realisiert.

Zalewski J.: ADA — a new programming language. Part 2. Programm units and instructions
INFORMATYKA 1982, No 1, p. 15

Second part of the ADA language presentation. Characteristics of basic elements of the language, which are so called programm units, and applied in these units instructions are presented.

Zalewski J.: ADA — eine neue Programmiersprache. Teil ?. Programmeinheiten und Anweisungen
INFORMATYKA 1982, Nr. 1, S. 15

Zweiter Teil der ADA-Sprache Präsentation. Es wurden die grundsätzlichen Elemente dieser Sprache, sog. Programmeinheiten, und die in diesen Einheiten verwendeten Anweisungen angegeben.

Informatyka

zastosowania w gospodarce, technice i nauce

Nr 1

MIESIĘCZNIK

1 9 8 2

ROK XVII

Kwiecień

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

W NUMERZE:

TADEUSZ WALCZAK
Główny Urząd Statystyczny
Warszawa

Informatyka w okresie przemian

Istniejąca kryzysowa sytuacja w polskiej informatyce ukształtowała się pod wpływem różnych czynników mających swoje źródło zarówno w samej informatyce, jak i poza nią. Prawidłowa identyfikacja tych źródeł ma ze zrozumiałych względów podstawowe znaczenie nie tylko dla prawidłowej diagnozy istniejącego stanu, ale również — a może przede wszystkim — dla określenia przyszłych kierunków i perspektyw.

Niniejszy artykuł odnosić się będzie w przeważającym stopniu do zastosowań informatyki, chociaż jest oczywiste, że problemów zastosowań nie sposób rozpatrywać w oderwaniu od innych części składowych tej dziedziny, a zwłaszcza od produkcji i eksploatacji sprzętu, kwalifikacji kadr czy zaopatrzenia materiałowego.

Symptomy narastania kryzysu w informatyce to przede wszystkim spadek tempa instalowania nowych komputerów, ograniczenie eksportu komputerów, brak postępu w wykorzystaniu sprzętu czy stagnacja zatrudnienia informatyków. Niektóre z wymienionych zjawisk ilustruje tabela 1.

Tabela 1

	Lata					
	1975	1976	1977	1978	1979	1980
Liczba komputerów dużych i średnich	514	623	708	756	812	857
Przyrost procentowy w w stosunku do roku poprzedniego	—	21,2	13,6	6,8	7,4	5,5
Liczba minikomputerów	430	924	1182	1336	1470	1776
Przyrost procentowy w stosunku do roku poprzedniego	—	114,9	27,9	13,0	10,0	20,8
Zatrudnienie w ośrodkach informatyki w tys.	41,3	47,1	51,0	56,2	56,9	57,1
Przyrost procentowy w stosunku do roku poprzedniego	—	14,0	8,4	10,1	1,3	0,0

Od 1977 r. zaobserwować można znaczny spadek wzrostu liczby nowych instalacji komputerowych. Liczba ta wprawdzie nadal się zwiększała, niemniej miał miejsce jednocześnie proces starzenia się parku komputerowego. Świadczą o tym dobitnie dane o procentowej strukturze komputerów (dużych i średnich) według wieku (tab. 2). Jeśli przyjąć, że komputery powinny być wycofane z eksploatacji po mniej więcej ośmiu latach, to ostatnio dostawy nowych komputerów nie pokrywały nawet liczby tych, które powinny być wycofane z eksploatacji. Przykładowo — różnice pomiędzy liczbą nowych komputerów zainstalowanych w danym roku a liczbą komputerów znajdujących się w eksploatacji przez dziewięć i więcej lat wynosiła (in minus) w 1978 r. — 57 szt., w 1979 — 77 szt., w 1980 r. — 135 szt. Faktycznie zatem nie pokrywany był nawet naturalny ubytek parku maszynowego.

Tabela 2

Wiek komputerów	Lata		
	1978	1979	1980
1— 3 lata	30,3	21,6	18,3
4— 5 lat	31,7	29,8	22,8
6— 8 lat	24,0	32,1	37,9
9— 10 lat	8,3	9,0	11,1
11— 15 lat	5,0	6,9	9,2
ponad 15 lat	0,7	0,6	0,7

Obserwuje się również od kilku lat brak postępu w wykorzystaniu komputerów. Tabela 3 przedstawia wykorzystanie komputerów mierzone czasem ich pracy (warto pamiętać o umowności tego wskaźnika). Jak wynika z przedstawionych tu danych komputery duże i średnie eksploatowane były przeciętnie w ciągu niespełnia dwóch zmian, natomiast minikomputery w ciągu niecałej jednej zmiany. Dane te obejmują ogólny czas pracy, a więc tzw. czas włączenia maszyn do sieci, na który składa się zarówno czas produkcyjny (łącznie z przeglądami technicznymi), jak i czas przestojów (z przyczyn technicznych i organizacyjnych).

Tabela 3. Ogólny czas pracy komputerów w godzinach na dzień roboczy

	Lata			
	1977	1978	1979	1980
Komputery duże i średnie	13,2	13,1	13,3	13,1
Minikomputery	6,0	6,5	6,4	6,3

Podział ogólnego czasu na rzeczywisty czas pracy oraz przestoje w przeliczeniu na dzień roboczy w godzinach przedstawia tabela 4.

Przytoczone tu dane świadczą o narastaniu w informatyce szeregu zjawisk ujemnych, które już dzisiaj są powodem pogarszania się jakości obsługi informacyjnej i stanowią zalążek jeszcze większych trudności w przyszłości. Do tych niekorzystnych zjawisk należą także — dotkli-

Tabela 4

	Lata			
	1977	1978	1979	1980
Komputery duże i średnie				
— czas pracy	10,6	11,0	11,1	10,9
— przestoje	2,6	2,1	2,2	2,2
Minikomputery				
— czas pracy	3,4	4,4	4,4	4,3
— przestoje	2,6	2,1	2,0	2,0

wie odczuwane przez ośrodki obliczeniowe — trudności w kompletowaniu koniecznych zestawów komputerowych. Krajowy przemysł realizuje konsekwentnie i z sukcesem monopolistyczny dyktat, narzucający użytkownikowi nie to, co jest mu potrzebne, ale to, co jest wygodne i opłacalne dla producenta. Prowadzi to w konsekwencji do użytkowania niekompletnych i nie zbilansowanych wewnętrznie — pod względem mocy obliczeniowej — konfiguracji komputerowych, nie pozwalających wykorzystać możliwości pracy wieloprogramowej, uruchomić bardziej zaawansowanych programów przetwarzania czy systemów wykorzystujących wspólną bazę danych.

Bardzo trudna jest sytuacja w zaopatrzeniu ośrodków obliczeniowych w materiały eksploatacyjne. Dotyczy to przede wszystkim papieru do drukarek wierszowych, taśm barwiących do urządzeń drukujących oraz wszelkich maszynowych nośników danych, a szczególnie — taśm magnetycznych. W wielu przypadkach krytyczna sytuacja zaopatrzeniowa ośrodków jest powodem przestoju maszyn oraz niewywiązywania się ośrodków z umów wobec swoich zleceniodawców

Znacznie bardziej niepokojącym zjawiskiem od opisanej stagnacji jest coraz wyraźniej dający o sobie znać kryzys społecznego zaufania do informatyki. Kryzys ten, trudny do scharakteryzowania za pomocą konkretnych wskaźników liczbowych, wyraża się przede wszystkim w rozczarowaniu użytkowników do rezultatów zastosowania informatyki oraz w przeświadczeniu informatyków, że ich wysiłki nie dają oczekiwanych efektów ekonomicznych i społecznych, gdyż dostarczane użytkownikom informacje nie są przez nich efektywne wykorzystywane.

W informatyce szczególnie zaobserwować można dysproporcję pomiędzy ogromnymi możliwościami sprzętu, oddziałującymi na wyobraźnię człowieka zdolnością wykonywania milionów operacji na sekundę oraz pamiętania nieograniczonej liczby informacji, a praktyką przetwarzania danych. Dysproporcje te są wykorzystywane często jako argument przeciwko komputeryzacji. Jeśli jednak podejść do tego problemu bez uprzedzeń, to w znacznej większości przypadków okaże się, że informacje o możliwościach komputerów (wyrażone za pomocą parametrów technicznych) nie są wcale przesadzone, natomiast niepowodzenia zastosowań wynikają z tego, że popełniono błąd w doborze komputera lub jego konfiguracji czy też nie stworzono odpowiednich warunków do właściwego wykorzystania środków informatyki.

Użytkownicy często zapominają, że zakup komputera stanowi tylko część inwestycji, mającej unowocześnić system informacyjny. Część znacznie trudniejsza, a często nie mniej kosztowna, polega na zapewnieniu odpowiednich warunków jego właściwego wykorzystania. Użytkownik mógłby uniknąć wielu trudności i rozczarowań, gdyby przed zastosowaniem komputera sformułował w sposób możliwie dokładny podstawowe cele i zadania tego przedsięwzięcia.

Od komputerów stosowanych w zarządzaniu oczekuje się zwykle osiągnięcia dwóch podstawowych celów. Pierwszy polega na zwiększeniu sprawności funkcjonowania systemu przetwarzania danych oraz uzyskaniu informacji niższym nakładem środków. Cel drugi natomiast polega na osiągnięciu przez użytkownika — dzięki zastosowaniu informatyki — lepszych wyników jego podstawowej działalności (obniżka kosztów produkcji, poprawa jakości wyrobów).

Stopień osiągania pierwszego celu można nazwać s k u t e c z n o ś c i ą informatyki (ang. *officiency*). Miarę osiągania drugiego celu nazywam e f e k t y w n o ś c i ą informatyki (ang. *efectiveness*). Dla większości rodzajów zastosowań oba te cele spełniane są jednocześnie. System skuteczny jest zarazem efektywny. Trzeba bowiem założyć, że system przetwarzania danych, będący jedną z funkcji aparatu zarządzania, spełnia swe zadania zgodnie z podstawowymi celami jednostki organizacyjnej, której służy. Mogą jednak zdarzyć się sytuacje, kiedy system przetwarzania danych jest efektywny, mimo że nie spełnia kryteriów skuteczności. Przykładowo, w skomputeryzowanych systemach rezerwacji miejsc w transporcie lotniczym lub w systemach kontroli stanu wkładów w kasach oszczędności skuteczność wyrażająca się wykorzystaniem, poszczególnych urządzeń komputera może być niewysoka, natomiast efektywność tych systemów (wyrażająca się w lepszym wykorzystaniu miejsc, ograniczeniu pustych przelotów, lepszej obsłudze klientów, ograniczeniu społecznych strat czasu na oczekiwanie w kolejkach, zmniejszeniu zdenerwowania klientów itp.) może być wysoka.

Zdarza się też niestety odwrotnie — bardzo dobrze wykorzystany, a więc skuteczny system przetwarzania danych bywa nieefektywny. Niewykorzystywanie lub niewłaściwe użycie przez użytkownika informacji dostarczanych przez ten system stanowi, zresztą, jeden z podstawowych zarzutów, jakie formułowane są przez informatyków pod adresem użytkowników informacji.

Uzyskiwanie niezbędnej informacji w wymaganym terminie jest jednym z istotnych warunków sprawnego kierowania działalnością społeczno-gospodarczą na różnych szczeblach zarządzania. Nie jest to jednak warunek jedyny. Konieczne jest także, by kierownictwo jednostek organizacyjnych podejmowało decyzje korygujące odchylenia sygnalizowane w otrzymywanych informacjach oraz realizowało w sposób optymalny podstawowe zadania danej jednostki. W przeciwnym bowiem razie, najsprawniejszy nawet system informacyjny nie zwiększy efektywności zarządzania i nie uzasadni celowości finansowania go.

W pierwszym, inauguracyjnym numerze czasopisma MASZYNY MATEMATYCZNE (poprzedni tytuł INFORMATYKI) w 1965 r. zamieszczono artykuł poświęcony komputeryzacji systemu planowania zaopatrzenia w części zamienne do pojazdów mechanicznych. Szczególną uwagę czytelnika zwracało zdjęcie pustych regałów sklepowych oraz podpis pod zdjęciem: „Puste półki w sklepach Motozbytu. Nie gwarantujemy, że w roku 1966 będą już wszystkie części zamienne, ale wiadomo chociaż, na jakie części będzie zapotrzebowanie" [1]).

Ten przykład sprzed kilkunastu lat skonfrontowany z naszą dzisiejszą oceną poprawy zaopatrzenia w części zamienne do pojazdów mechanicznych może służyć jako ilustracja sformułowanej poprzednio tezy, że nawet bardzo sprawny system informatyczny okaże się nieefektywny, jeśli nie współdziałają z nim czynniki ekonomiczne i organizacyjne zapewniające podejmowanie skutecznych działań, reagujące na sytuację sygnalizowaną przez system informatyczny.

REFORMA GOSPODARCZA — SZANSE INFORMATYKI

Zgodnie z założeniami Reformy, podstawowym zadaniem przemian jest wprowadzenie w życie takich zasad i takiego mechanizmu funkcjonowania gospodarki, które zapewnią jej wysoką społeczną efektywność. [2]) Oznaczać to więc musi przywrócenie rangi gospodarności oraz liczenie kosztów i efektów we wszystkich dziedzinach gospodarowania, zarówno w skali przedsiębiorstw, jak i całej gospodarki narodowej.

Analogicznym wymogom efektywności muszą być podporządkowane przedsięwzięcia w dziedzinie informatyki. Spowoduje to zapewne:

- Dokonanie krytycznej analizy tzw. dużych systemów informatycznych o zasięgu wojewódzkim i krajowym, które były ściśle związane z obowiązującym dotąd rozdzielczo-nakazowym systemem zarządzania. Prawdopodobnie niektóre z tych systemów okażą się zbędne lub zbyt kosztow-

[1]) Jan Przybylski: Automatyzacja planowania Behamot. MASZYNY MATEMATYCZNE, 1965, nr 1, s. 18—20.

[2]) Kierunki reformy gospodarczej, projekt, nakładem Trybuny Ludu, Warszawa, lipiec 1981, s. 8.

ne w zestawieniu z efektami ich funkcjonowania i będą musiały być zlikwidowane (bez większej szkody i dla społeczeństwa, i dla informatyki). W większości tych systemów poniesiono dotąd jedynie nakłady na prowadzenie badań i projektowanie, niemniej w wielu przypadkach są to nakłady niemałe.

• Przewartościowanie celów i funkcji zastosowań informatyki w przedsiębiorstwach. Nie jest tajemnicą, że w przeszłości w wielu przypadkach zastosowanie informatyki nie wynikało z potrzeb przedsiębiorstw, lecz z innych przesłanek, wśród których niepoślednią rolę odgrywały wspomniane poprzednio przesądzone opinie o tym, że „komputer jest dobry na wszystko", a także: obawa kierownictwa przedsiębiorstw przed zarzutem hamowania postępu, naciski jednostki nadrzędnej, pokrywanie wydatków na informatykę z funduszy na prace badawczo-rozwojowe czy naciski na ograniczenie zatrudnienia w komórkach zarządu, przedsiębiorstwa i jednoczesny brak ograniczeń wydatków na usługi EPD.

Wymienione wyżej czynniki spowodują najprawdopodobniej okresowe zmniejszenie się zapotrzebowania na informatykę. Dotyczyć to będzie zwłaszcza tych „wydumanych" zastosowań i systemów, które budowane były na miarę wybujałej ambicji władzy i schlebiających jej informatyków, a nie na miarę potrzeb gospodarki i społeczeństwa. Jednocześnie jednak Reforma uruchomi dźwignie przyspieszające szereg takich kierunków działań, dla urzeczywistnienia których zastosowanie środków informatyki stanie się naturalną potrzebą.

Podstawowe zasady funkcjonowania przedsiębiorstw po reformie gospodarczej, a więc samodzielność i samorządność przedsiębiorstw, przejawiające się m.in. w odpowiedzialności samorządu i kierownictwa przedsiębiorstwa za skutki podejmowanych decyzji, postawią ją wobec nieubłaganych wymagań efektywności gospodarowania. Spowoduje to konieczność zorganizowania wiarygodnego i sprawnego systemu informacji wewnętrznej. Musi on zapewnić operatywną ewidencję zasobów, ewidencję i kontrolę umów kooperacyjnych i kontrolę ich realizacji, kalkulację nakładów i wyników, a także niezbędną informację rynkową, gwarantującą zachowanie profilu produkcyjnego zgodnego z oczekiwaniami i potrzebami odbiorców wyrobów przedsiębiorstwa. Wydaje się mało prawdopodobne, aby taki system informacyjny, zwłaszcza w większych przedsiębiorstwach, można było zorganizować bez zastosowania środków informatyki.

Radykalne zwiększenie samodzielności przedsiębiorstw i ograniczenie (a docelowo — likwidacja) zadań dyrektywnego i administracyjnego rozdzielnictwa środków nie oznacza, że również w przyszłości nie będzie konieczne podejmowanie znaczących społecznie decyzji na szczeblu centralnym. Decyzje te dotyczyć muszą zwłaszcza: programów rozwoju kraju (a zwłaszcza tempa i podstawowych proporcji rozwojowych), tworzenia i podziału dochodu narodowego, polityki zatrudnienia, dochodów i cen, działalności inwestycyjnej, budownictwa mieszkaniowego czy współpracy gospodarczej z zagranicą. Aby to centralne kierowanie gospodarką było możliwe musi zostać stworzony odpowiedni system przepływu informacji z podstawowych jednostek gospodarczych do centralnych. Informacje te będą się zapewne znacznie różnić od treści istniejących obecnie centralnych sytemów informacyjnych, niemniej dla ich zebrania, przetworzenia i udostępnienia niezbędne będzie zastosowanie odpowiednich systemów informatycznych i teleinformatycznych.

Dodatkowym argumentem za szerokim stosowaniem informatyki w tych systemach będzie konieczność zorganizowania przepływu informacji bezpośrednio z przedsiębiorstw (przykładowo — za pośrednictwem organów statystyki państwowej) w związku z likwidacją lub zmianą funkcji pośrednich szczebli zarządzania (zjednoczeń, zrzeszeń), które obecnie uczestniczą w procesie zbierania i opracowania informacji przekazywanej z przedsiębiorstw na szczebel centralny.

Zwiększy się też bez wątpienia zapotrzebowanie na usługi informatyczne ze strony banków, w związku ze zmianą i rozszerzeniem ich zadań w systemie funkcjonowania gospodarki.

Znacznie wyższą niż dotąd rangę musi ponadto uzyskać zastosowanie informatyki w systemach związanych z usprawnieniem obsługi ludności. Dotyczy to zwłaszcza usług pocztowo-telekomunikacyjnych, rezerwacji i sprzedaży biletów w komunikacji drogowej, kolejowej i lotniczej, obsługi kredytowo-finansowej czy usług komunalnych.

W sumie — po pewnym zahamowaniu zapotrzebowania na informatykę, wynikającym z ogólnego kryzysu oraz załamania się niektórych dotychczasowych koncepcji zastosowań — można oczekiwać stopniowego wzrostu zapotrzebowania na usługi informatyczne. Aby to jednak nastąpiło, konieczne jest spełnienie szeregu warunków metodologicznych i technicznych w samej informatyce oraz w niektórych dziedzinach gospodarki pracujących dla informatyki lub ściśle z nią związanych. Chodzi tu głównie o usunięcie barier, które w przeszłości często, niezależnie od obiektywnych, niesprzyjających efektywnemu wykorzystaniu informatyki okoliczności, zniechęcały użytkowników do informatyki.

• Informatycy muszą przestać rozpatrywać komputeryzację jako cel sam w sobie i zacząć ją traktować jako metodę i środek osiągania celu, którego sformułowanie musi należeć do użytkownika. Dlatego też niezbędnym warunkiem zaprojektowania i uruchomienia sprawnego systemu informatycznego, który jednocześnie spełniałby kryteria efektywności, wyrażające się w usprawnieniu zarządzania i poprawie wyników działalności jednostki, jest aktywne współdziałanie użytkowników informacji w toku projektowania i wdrażania systemu. W praktyce prace te zbyt często powierza się informatykom przy jednoczesnym braku dostatecznego zaangażowania się odbiorców informacji oraz bez jakiejkolwiek kontroli z ich strony. W tych warunkach informatycy mają skłonność do tworzenia systemów nadmiernie złożonych, podporządkowanych bardziej wymaganiom technologii przetwarzania danych niż potrzebom użytkownika, natomiast użytkownicy systemów wpadają w zbyt dużą zależność od systemu skomputeryzowanego, a jednocześnie nie mogą wpływać na jego doskonalenie. Użytkownicy informacji potrzebnej do zarządzania, a więc kierownicy jednostek i komórek informacyjnych wszystkich szczebli, muszą posiąść umiejętność znajdowania wspólnego języka z projektantami systemów informatycznych i innymi informatykami zaangażowanymi w proces przygotowania niezbędnej do zarządzania informacji. Rzecz nie w tym, aby użytkownicy informacji zbyt szczegółowo wnikali w tajniki funkcjonowania sprzętu komputerowego, czy w technikę programowania. Problematyka ta jest i musi nadal pozostawać domeną ekspertów — informatyków, natomiast użytkownicy muszą się nauczyć rozumieć język tych ekspertów, wyzbyć się kompleksu pozornej niedostępności informatyki, a gdy trzeba — potrafić narzucić rozwiązanie informatyczne odpowiadające ich potrzebom informacyjnym.

W tym celu konieczne jest stałe podnoszenie wiedzy informatycznej wśród aktualnych i potencjalnych użytkowników informacji, a więc wśród szerokiego kręgu pracowników aparatu zarządzania, członków samorządu pracowniczego, studentów wyższych szkół ekonomicznych i technicznych, nauczycieli mających wpływ na wychowanie nowego pokolenia specjalistów z różnych dziedzin.

• Informatycy muszą bardziej niż dotąd wnikać w rzeczywiste potrzeby użytkowników i proponować zastosowanie środków informatyki w tych dziedzinach, których właściwe funkcjonowanie ma najistotniejsze znaczenie dla użytkownika i w których pozytywne efekty zastosowań mogą przejawić się najszybciej i w sposób przekonywający. Oznaczać to powinno — między innymi — konieczność wdrażania systemów ewidencyjnych, usprawniających obieg informacji w przedsiębiorstwie i zapewniających doskonalenie kontroli wykorzystania środków, walkę ze stratami i niegospodarnością czy racjonalizację zatrudnienia.

• Przemysł środków informatyki musi przestawić się z pozycji monopolisty, dostarczającego użytkownikowi sprzęt i oprogramowanie w ilości i asortymencie według potrzeb i wymagań, lecz według jego własnych chęci, wyobrażeń lub możliwości. Jeśli nie zostaną tu dokonane radykalne zmiany, to wszelkie plany racjonalizacji i zwiększenia efektywności informatyki będą nierealne.

• Konieczna jest zasadnicza poprawa jakości i niezawodności dostarczonego sprzętu i oprogramowania oraz zwiększenie odpowiedzialności dostawcy za ich prawidłowe funkcjonowanie. Jedną z form obrony użytkownika przed złą jakością sprzętu i oprogramowania powinno być szerokie wprowadzenie zasady dzierżawy sprzętu oraz uzależnienie wysokości opłat dzierżawnych od rzeczywiście osiąganych wyników pracy sprzętu.

RYSZARD FRYŃ
Zakład Elektronicznej Techniki Obliczeniowej FSO
Warszawa

Doświadczenia i problemy programownaia aplikacyjnego w FSO

Problem oprogramowania systemów użytkowych przejawia się w dwóch podstawowych aspektach: organizacji procesów programowania oraz jego metodologii (technologii). Oba te aspekty są ściśle ze sobą powiązane i wzajemnie uzależnione. Przy retrospektywnym ujęciu problemu można zaobserwować znamienną ewolucję poglądów na programowanie. Z jednej strony — u laików — dotychczasowe przekonanie, że programowanie jest specyficznym rodzajem „wiedzy tajemnej" ustępuje miejsca traktowaniu go jako normalnego zawodu. Zjawisko to jest wynikiem rozpowszechniania się informatyki i jej zastosowań. Z drugiej strony dotychczasowe doświadczenia uzyskane z wdrożeń eksploatacji i konserwacji systemów informatycznych skłaniają kierownictwa ośrodków obliczeniowych do przywiązywania coraz większej wagi do problemu programowania aplikacyjnego.

Pod wieloma względami typowy jest jego rozwój w Zakładzie Elektronicznej Techniki Obliczeniowej Fabryki Samochodów Osobowych w Warszawie. Ośrodek ten ma bardzo bogate tradycje, sięgające końca lat pięćdziesiątych, kiedy to powstała w FSO stacja maszyn licząco-analitycznych, która w latach siedemdziesiątych rozwinęła się w bogato wyposażony, jeden z większych w Polsce, komputerowy ośrodek obliczeniowy.

Pierwotnie problem programowania w FSO dostrzegano jedynie w jego aspekcie organizacyjnym, a dokładniej precyzując, sprowadzano go do kwestii umiejscowienia programisty w strukturze organizacyjnej ośrodka obliczeniowego.

Odpowiedź na to pytanie stała się konieczna w momencie przekształcenia się kilku a następnie kilkunastoosobowej grupy projektowo-programowej, realizującej jeden projekt, w zespół kilkudziesięcioosobwy, realizujący równolegle kilka projektów, często luźno lub wcale ze sobą nie powiązanych. Bezpośrednie kierowanie tak dużym zespołem projektantów i programistów, a także kilkoma równolegle realizowanymi projektami oraz konserwacją już eksploatowanych systemów stało się bardzo trudne i mało efektywne. W tej sytuacji zespół podzielono na monotematyczne grupy projektantów i programistów. Utworzono wówczas kilka zespołów, z których każdy realizował tylko jeden projekt od początku prac aż do momentu jego wdrożenia do eksploatacji użytkowej, a następnie przeprowadzał kompleksową konserwację systemu.

Organizacja taka ma wiele zalet, z których podstawową jest bezpośrednia współpraca analityków i projektantów z programistami. Najczęściej trudno jest wyznaczyć w takim zespole dokładną granicę między poszczególnymi wykonawcami, ponieważ często projektanci podejmują się zadań programowych, a programiści wykonują zadania projektowe. Bliski kontakt z projektantem sprawia, że programista zna bardzo dobrze projekt, co jest szczególnie istotne przy opracowywaniu i uruchamianiu programów. Dysponując projektantami i programistami, kierownik zespołu może działać w sposób operatywny, zgodnie z potrzebami projektu, co znacznie przyspiesza jego realizację i uruchomienie. Dotyczy to szczególnie mniejszych systemów o krótkim czasie realizacji. Praca zespołu projektowo-programowego jest podporządkowana podstawowemu celowi — szybkiej realizacji projektu, czemu sprzyja opisany sposób organizacji.

Jednakże w przypadku monotematycznych grup projektowo-programowych o sztywnej strukturze, która w warunkach polskich głównie ze względów płacowych (choć nie jedynie) jest dominująca, powstaje bardzo poważny problem natury organizacyjnej, a mianowicie co zrobić z zespołem, który zakończył pracę nad systemem i przekazał go do eksploatacji. Narzucające się alternatywne rozwiązania, polegające na likwidacji zespołu albo zleceniu mu nowego systemu, okazują się w praktyce trudne do realizacji. Wynika to z konieczności konserwacji, a zwłaszcza stałego dostosowywania eksploatowanego systemu do zmieniających się warunków.

Przy organizowaniu grup projektowo-programowych do tego rodzaju zadań występują następujące trudności:

- niekompletność i usterki w istniejącej dokumentacji (mankament powszechnie występujący)
- brak sprecyzowania, kto ma wykonywać konserwację systemu.

Jeżeli zespół został rozwiązany, to pojawia się problem, komu zadanie to należy przekazać. W większości przypadków można się spodziewać, że każdy z istniejących zespołów będzie się bronił przed przejęciem takiego zadania.

Jeżeli zaś zespół zostanie utrzymany, to oprócz nowych zadań projektowych będzie musiał poświęcać część swego potencjału wykonawczego na konserawcję tego, co uprzednio stworzył. W zależności od czasu istnienia zespołu, jakości wykonania sytemu oraz zmienności warunków, w jakich systemy te działają, prędzej czy później dochodzi do sytuacji, kiedy opieka nad starymi systemami praktycznie uniemożliwia tworzenie nowych.

Dużą wadą organizacji grup projektowo-programowych jest również stosunkowo mała średnia wydajność pracy poszczególnych projektantów, a jeszcze bardziej — programistów. Wynika to z istoty procesu tworzenia systemów informatycznych. Jak wiadomo proces taki dzieli się na szereg etapów projektowych i programowanych, występujących w określonej kolejności. Programista może rozpocząć pracę dopiero po zakończeniu (w najlepszym przypadku dostatecznym zaawansowaniu) prac wykonywanych przez analityków i projektantów. Powoduje to nierównomierne w czasie rozłożenie obciążenia pracą projektantów i programistów. Na etapie analizy i projektowania systemu programiści są praktycznie nie obciążeni, wykonując ewentualnie czynności nie wchodzące w zakres obowiązków programisty (np. maszynopisanie, kreślenie rysunków czy inne prace pomocnicze).

Zalety monotematycznego zespołu projektowo-programowego, ujawniają się tylko wtedy, kiedy wszyscy — lub de-

Mgr Ryszard FRYŃ ukończył w 1971 r. studia na Wydziale Ekonomii Politycznej Uniwersytetu Warszawskiego o kierunku ekonometrii. Od września 1971 roku pracuje w ZETO-FSO, początkowo w charakterze programisty, a od 1975 roku na stanowisku kierownika Działu Programowania EPD.

cydująca większość jego członków — mają wysokie kwalifikacje i praktyczną umiejętność samodzielnego wykonywania zawodu.

Adaptacja i przyuczanie do zawodu w małym zespole, gdzie funkcje są ściśle rozgraniczone i prawie każdy jego członek ma inną specjalność zawodową, są bardzo trudne (projektant nie jest najlepszym nauczycielem programowania, a sytuacja odwrotna jest praktycznie absurdalna).

Etapowość tworzenia systemów (analiza-projektowanie-programowanie-testowanie-wdrożenie) powodowała powstawanie w poszczególnych zespołach okresowych deficytów potencjału wykonawczego programistów, hamując skutecznie postęp prac nad projektem. Często powstawać mogą sytuacje wynikające z różnic w stopniu zaawansowania poszczególnych projektów, a mianowicie, w jednym zespole grupa programistów będzie miała zadania przekraczające jej możliwości, podczas gdy w tym samym czasie szef innego zespołu będzie miał duże kłopoty z pełnym zatrudnieniem swoich programistów. Oczywiście po pewnym czasie sytuacja się zwykle odwraca.

Aby rozszerzyć tego rodzaju okresowe „wąskie przekroje" w programowaniu, kierownicy zespołów dążyli do zatrudniania programistów w liczbie odpowiadającej potrzebom maksymalnego spiętrzenia prac programowych. Niewiele to pomagało z powodu chronicznego braku wysokokwalifikowanych, samodzielnych programistów, natomiast doszkalanie pracowników zatrudnionych odbywało się zbyt wolno.

Dlatego też kierownicy zespołów raczej z ulgą przyjęli koncepcję reorganizacji, polegającą na utworzeniu ze wszystkich programistów odrębnego Działu Programowania, którego zadaniem jest oprogramowywanie wszystkich projektów. Reorganizacja ta początkowo dotyczyła jedynie zespołów korzystających z systemu komputerowego IBM 370, a następnie — również systemu R-32. Uznano za niecelowe łączenie programistów obsługujących różnego typu minikomputery. Jednocześnie powstał Dział Projektowania, którego pracownicy udzielają zleceń na oprogramowanie swoich projektów, popartych sformułowaniem założeń dla poszczególnych programów.

Powstanie Działu Programowania stworzyło przesłanki organizacyjne dla wzrostu wydajności pracy programistów. Skupienie wszystkich programistów pod jednym kierownictwem pozwoliło bardziej równomiernie przydzielać zadania poszczególnym wykonawcom, a dzięki odpowiedniemu ich rozłożeniu w czasie poprawiła się rytmiczność pracy oraz wskaźnik wykorzystania czasu pracy. W efekcie udało się w zasadzie zlikwidować pracę w godzinach nadliczbowych, co występowało już nagminnie i było przyczyną licznych skarg.

Oczywiście, programy tworzone w dużej mierze podczas drugiej, a nawet trzeciej zmiany nie mogły być najwyższej jakości, co często w trakcie eksploatacji powodowało konieczność dokonywania licznych poprawek. Poprawki te zwiększały liczbę zadań programowych i tym samym obciążenie programistów.

Włączenie wszystkich programistów do jednej komórki organizacyjnej spowodowało ujednolicenie kryteriów oceny ich pracy, co niewątpliwie wpływa na poprawę stosunków międzyludzkich a więc nie jest bez znaczenia dla wydajności pracy. Nastąpiło również częściowo uniezależnienie programisty od nieprzewidzianych przerw w pracy i awarii komputera. Wykonując jednocześnie kilka programów może on, w przypadku wystąpienia okresowych trudności z dostępem do komputera, realizować prace nad innymi programami, których stopień zaawansowania jeszcze tego dostępu nie wymaga.

Nie bez znaczenia są również takie możliwości, jak lepsze dopasowanie zadań do umiejętności, kwalifikacji i doświadczenia poszczególnych członków oraz adaptacja zawodowa nowych pracowników, która zawsze jest szybsza w zespołach jednorodnych zawodowo. Wyłączenie programistów z grup projetkowo-programowych, gdzie oprócz programowania wykonywali szereg prac pomocniczych i skupienie ich w zespole, którego zadaniem jest wyłącznie programowanie, powoduje szybsze podnoszenie kwalifikacji zawodowych. Tak więc, stworzone zostały warunki do doskonalenia metodologii pracy programisty i technologii programowania.

Ponad pięcioletnie doświadczenie istniejących w FSO-ZETO struktur i powiązań organizacyjnych programistów z projektantami wykazały, że ich rozdzielenie w dużej mierze spowodowało utratę emocjonalnego przywiązania programisty do projektu, co tylko pozornie jest zjawiskiem negatywnym.

Pisząc program w zespole projektowo-programowym programista w przypadku niejasności w założeniach (niezdefiniowaniu funkcji, niezgodności danych z założeniami itp.) często, w najlepszej wierze, powodowany wspólnym dążeniem zespołu do jak najszybszego zakończenia projektu, sam rozwiązuje napotkane problemy, nie zawsze konsultując je z projektantem. Niestety, rozwiązania programisty są nie zawsze zgodne z intencjami kierownika projektu i najczęściej nie są odpowiednio udokumentowane. W takiej sytuacji program może działać w sposób zaskakujący dla projektanta, przedłużając pracę nad wdrożeniem systemu, lub powodować wadliwe jego funkcjonowanie w czasie eksploatacji.

Ogólnie rzecz biorąc, programista w monotematycznym zespole projektowo-programowym kieruje się w pierwszym rzędzie kryterium osiągnięcia zgodności programu z celem opracowywanego systemu. Innym kryterium kieruje się programista pracujący w osobnej, wyspecjalizowanej w oprogramowywaniu zastosowań komórce organizacyjnej. Dla niego istotna jest zgodność programu z dokumentacją zawierającą założenia programowe, natomiast mniej go interesuje sam system jako przedmiot projektu i gdy w założeniach programowych występują luki lub niejasności, od projektanta żąda on wyjaśnień i poprawienia dokumentacji. Dzięki tej weryfikacji poprawiają się walory eksploatacyjne systemu, jego jakość i dokumentacja.

METODY I TECHNOLOGIE PROGRAMOWANIA

Niektóre osoby może razić termin „technologia" w odniesieniu do programowania, które jest pracą umysłową o dużym udziale twórczości intelektualnej. Jednak w przypadku Działu Programowania w FSO termin ten jest w pełni uzasadniony, jeżeli uwzględni się fakt, że dział ten wytwarza dużą liczbę programów aplikacyjnych według ściśle określonych wymagań. Programista ma narzucony sposób pisania programów, a przekazywany do eksploatacji program musi nie tylko realizować określone w założeniach funkcje, ale również mieć określoną konstrukcję i formę.

W FSO korzysta się jedynie z dwóch języków programowania, a mianowicie ASSEMBLERA oraz PL/I, przy czym w oprogramowaniu zastosowań wykorzystuje się głównie PL/I, dlatego wprowadzone usprawnienia dotyczą programowania przede wszystkim w tym języku. Mają one na celu podwyższenie wydajności pracy oraz jakości programów. Związek wydajności programowania z ich jakością jest wyraźnie widoczny szczególnie w Dziale Programowania EPD, którego zadaniem jest — jak już wspomniano wyżej — nie tylko oprogramowanie nowych projektów, ale również obsługa programowa już eksploatowanych systemów.

Początkowo starano się podnieść wydajność programowania poprzez próbę stworzenia zbioru uniwersalnych programów i modułów, realizujących wiele różnych funkcji. Szczególnie dotyczyło to oprogramowania problematyki kadrowej. W tej dziedzinie występuje największa liczba tzw. „pilnych" (krótkoterminowych) zamówień, charakteryzujących się z reguły niewielkim stopniem trudności. Niestety próba realizacji tego zamierzenia nie przyniosła spodziewanych rezultatów. O niepowodzeniu zdecydowały dwie podstawowe przyczyny. Pierwsza, to bardzo duża różnorodność zamówień użytkowników, przy jednoczesnej dużej niechęci do korzystania z materiałów wcześniej opracowanych dla innych użytkowników (o innym wzorze wydruku niż żądany). Druga, wynikająca z właściwości programów uniwersalnych, polega na tym, że im program jest bardziej uniwersalny (tzn. im większa jest różnorodność realizowanych funkcji), tym gorsze są jego walory eksploatacyjne: dłużej się wykonuje, wymaga większej pojemności pamięci wewnętrznej, blokuje lub hamuje wykonywanie innych programów (przy wieloprogramowości) i wreszcie jest kłopotliwy w operowaniu. Uwzględniając fakt, że stworzenie uniwersalnego programu jest raczej trudne, ograniczono te przedsięwzięcia do niewielkiej liczby realizujących proste funkcje na poziomie programów pomocniczych.

Bardzo dobre rezultaty i stosunkowo dużą oszczędność czasu kodowania przyniosło stworzenie wspólnej dla wszystkich programów biblioteki wszystkich zapisów w stałych zbiorach danych. Dzięki tej bibliotece programista nie musi kodować (a nawet nie wolno mu tego robić) struktur zapisów zbiorów umieszczanych w programie. Skraca się więc czas kodowania programu, zmniejsza prawdopodobieństwo pomyłki, a ponieważ projektant w swojej dokumentacji — przekazywanej do Działu Programowania — zobowiązany jest również odwoływać się do skatalogowanych w bibliotece struktur zapisów (dotyczy to oczywiście istniejących już zbiorów danych), postępowanie takie zapobiega powstawaniu ewentualnych błędów podczas definiowania danych. Mimo, że biblioteka struktur zapisów jest narzędziem prostym, to w praktyce okazuje się bardzo efektywna.

Drugim przedsięwzięciem, podnoszącym wydajność pracy programisty było wyposażenie Działu w monitory ekranowe oraz system CRJE (Communication Remote Job Entry), umożliwiający kodowanie i wprowadzanie programów do komputera z pominięciem kart dziurkowanych. Skróciło to w znacznym stopniu czas zapisywania programu na nośnik maszynowy, poprawiania programu w bibliotece oraz operowania programem podczas eksploatacji. Przy przekazywaniu programu do Działu Eksploatacji programista nie ma prawa wstępu na salę komputerową, a przecież do czasu oczekiwania w kolejce i działania programu w komputerze oraz czasu przekazywania wyników przez operatora maszyny trzeba dodać jeszcze czas przyjmowania kart przez operatora i ich wczytywania. System CRJE w pewnym stopniu uniezależnił więc programistę od operatora.

Największe efekty i największy wpływ na podniesienie wydajności pracy programistów przyniosło doskonalenie technologii programowania oraz podniesienie kwalifikacji programistów. W zakres zadań zarówno Działu, jak i obowiązków poszczególnych jego pracowników, obok normalnych zadań produkcyjnych, wpisane zostało stałe szkolenie i podnoszenie kwalifikacji. Zadanie to realizuje się wykorzystując różne formy kształcenia, poczynając od indywidualnego samokształcenia i studiowania literatury fachowej, poprzez organizowanie kursów wewnętrznych aż do specjalistycznych kursów zewnętrznych.

Stałe szkolenie jest warunkiem doskonalenia metod i technik programowych, a tym samym jednym z najtańszych sposobów podnoszenia wydajności pracy programisty oraz zapewnienie awansu zawodowego.

W unowocześnianiu metodologii i technologii programowania skorzystaliśmy z doświadczeń firmy IBM, w szczególności w zakresie programowania strukturalnego, wchodzącego w skład IPT (Improved Programming Technology).

O wyborze metody programowania strukturalnego przesądziły niżej opisane względy. Tworzone w FSO systemy i programy mają przeważnie długi żywot, przekraczający często pięć lat oraz są silnie związane z użytkownikiem systemu. Ze względu na dużą zmienność wymagań użytkowników, co jest zjawiskiem normalnym w działalności wielkiego, wielozakładowego przedsiębiorstwa, systemy użytkowe ulegają stosunkowo czzstym zmianom i uzupełnieniom, co oczywiście powoduje konieczność modyfikowania programów. Wobec tego szczególnie ważna jest dobra dokumentacja programowa. Doświadczenie wykazało, że każda dokumentacja programów bardzo szybko ulega dezaktualizacji, a tym samym staje się bezużyteczna.

Metoda programowania strukturalnego, zakładająca samodokumentowanie się programów, pozwoliła skutecznie rozwiązać problem dokumentacji programowej. Rozwiązanie to, jak i również ujednolicenie i zdyscyplinowanie sposobów rozwiązań programowych sprawiły, że wszystkie programy są czytelne i zrozumiałe dla kadżego programisty, który bez większej trudności może dokonywać zmian nawet w programach, których nie jest autorem. Tak więc programowanie strukturalne nie tylko zmniejszyło pracochłonność modyfikowania gotowych, eksploatowanych już programów, ale również zwiększyło elastyczność wykorzystywania kadry programistów.

O ile przed wprowadzeniem przed czterema laty metody programowania strukturalnego, 70—80% globalnych zasobów czasu pracy Działu Programowania było przeznaczone na poprawienie i modyfikowanie już eksploatowanych programów, a jedynie 20—30% na tworzenie nowych (dla nowych zastosowań), to obecnie proporcje uległy odwróceniu mimo, że powiększyła się liczba eksploatowanych programów, a tym samym wzrosło globalne zapotrzebowanie na modyfikację. Jednocześnie stan zatrudnienia w Dziale zmniejszył się o połowę, co świadczy o znacznym wzroście wydajności pracy. Dane te potwierdzają bardzo dużą efektywność stosowania metody programowania strukturalnego i wysoką jego opłacalność, a także słuszność obranego kierunku rozwoju oprogramowania użytkowego.

Proces wdrażania metody programowania strukturalnego jest trudny i wymaga od kierownictwa dużej konsekwencji oraz umiejętnego zachęcania programistów do jej stosowania, zwłaszcza doświadczonych programistów, którzy w początkowym okresie odnoszą się do tej metody z dużą nieufnością i sceptycyzmem. Wdrażanie metody programowania strukturalnego w FSO rozpoczęto od przeszkolenia na czterodniowym kursie zewnętrznym (zorganizowanym przez IBM) z oderwaniem od pracy. Objęło ono połowę składu osobowego Działu Programowania, głównie doświadczonych pracowników. Kolejnym etapem był wewnątrzdziałowy kurs-seminarium, prowadzony przez absolwentów wspomnianego kursu zewnętrznego, którzy przed skierowaniem na ten kurs zostali zobowiązani do przekazania uzyskanych wiadomości pozostałej części pracowników.

Wewnątrzdziałowy kurs programowania strukturalnego miał w dużej mierze charakter seminaryjny i dzięki niemu osiągnięto następujące cele:

- zapoznanie pozostałych pracowników Działu z nową metodą programowania

- dokładne wyjaśnienie problemów, które nie zostały przez wszystkich zrozumiane na pierwszym kursie oraz wspólne opracowanie i przyjęcie określonych wniosków, wynikających z przyjęcia nowej metody do codziennej praktyki programowania.

Uwzględniając reguły programowania strukturalnego oraz zasadę samodokumentowania programów użytkowych opracowano instrukcję programowania, obejmującą wymogi, jakie powinna spełniać lista przekazywanego do eksploatacji programu.
Wymogi te są następujące:

- **kolumny 71 i 72** w liście programu są zarezerwowane na znaki ograniczające komentarz, zawierający wykonywane tylko przy testowaniu programu diagnostyczne fragmenty kodu (przy odpowiednim ustawieniu parametru SORMGIN);

- **program** (procedura) powinien składać się z czterech części: czołówki, bloku deklaracji, bloku inicjacji i „właściwego" programu; wymienione części powinny znajdować się na osobnych stronach;

- **czołówka programu** powinna zawierać:

a) nazwę procedury, zapisaną począwszy od kolumny 2, zgodną z założeniami projektanta

b) krótki, opisowy tytuł programu (procedury), zapisany jako komentarz w tym samym wierszu co nazwa

c) w komentarzach bloku informacji:
— opis funkcji procedury
— krótki opis danych wejściowych (zbiory, parametry)
— krótki opis danych wyjściowych (zbiory, tabulogramy);

d) informacje o programiście oraz o ewentualnych modyfikacjach programu (autor oraz data i zakres modyfikacji), zapisane w komentarzu na końcu czołówki

- **blok deklaracji** powinien zawierać:

a) w pierwszym wierszu, począwszy od kolumny 2, słowo kluczowe PLI PROCEDURE oraz listę parametrów

b) poczynając od drugiego wiersza — wszystkie deklaracje wg następujących zasad:
— słowo kluczowe DECLARE (zawsze od kolumny 2)
— etykiety pól winny zawsze zaczynać się od kolumny 7 (zaleca się unikanie etykiet z łącznikiem „—")
— atrybuty typu i precyzji deklarowanego pola powinny być zapisane począwszy od kolumny 16, w tym samym wierszu co etykieta
— pozostałe atrybuty deklarowanego pola, razem z jego inicjacją, powinny być pisane w osobnym wierszu — począwszy od kolumny 16

— komentarze opisujące pola deklarowane powinny zawierać się między kolumnami 33—70

— zaleca się opisywanie w komentarzu każdego deklarowanego pola

— w przypadku deklarowania struktur danych, indeksy poziomu należy wpisywać w kolumnie o numerze określonym zgodnie z następującą regułą: nr kolnmny = indeks poziomu + 1

— średnik kończący zdanie deklaracji powinien znajdować się w osobnym wierszu w kolumnie 2

— zaleca się minimalizowanie liczby słów kluczowych DECLARE;

- **blok inicjacji** powinien zawierać wszelkiego rodzaju inicjacje pól, otwarcie zbiorów i warunki ON CONDITION. W zdaniu OPEN każdy zbiór powinien być specyfikowany w osobnym wierszu

- zaleca się unikania w programie używania etykiet (z wyłączeniem etykiet pól), a jeżeli jest to konieczne, to powinny być one zapisane w osobnym wierszu, począwszy od kolumny 2

- zdania podrzędne powinny być przesunięte w prawo, w stosunku do zdań nadrzędnych, najlepiej o dwie kolumny

- zdania DO, BEGIN i END powinny być zawsze w osobnych wierszach

- w grupach DO i BEGIN zdanie END powinno być zapisane do tej samej kolumny co zdanie rozpoczynające

- w zdaniach typu PUT i GET listy zmiennych i listy formatów powinny być umieszczone w osobnych wierszach

- w zdaniu IF THENELSE słowa kluczowe powinny być zapisane w osobnych wierszach od tej samej kolumny

Jak widać z powyższego, reguły te choć szczegółowe są dosyć proste i w praktyce okazały się bardzo przydatne. Kontrola przestrzegania powyższych reguł pisania programów należy do obowiązków bibliotekarza działu informatycznego, który przejmuje gotowy, przetestowany program od programisty. Po formalnym zweryfikowaniu programu zostaje on przekazany do Działu Eksploatacji z jednoczesnym usunięciem z działowej biblioteki programów.

Oprócz wymienionych obowiązków bibliotekarz opiekuje się (w szerokim tego słowa znaczeniu) wszystkimi bibliotekami programów Działu. Ze względu na znaczną liczbę programów i bibliotek oraz częste modyfikacje programów, zadanie to jest bardzo odpowiedzialne i trudne.

Nie wszystkie elementy programowania strukturalnego i IPT zostały w FSO wykorzystane i wdrożone. Na przykład, bezpośrednie przeniesienie, bez uwzględnienia istniejących w fabryce warunków (zresztą typowych dla gospodarki polskiej), koncepcji zespołów programisty wiodącego mogłoby przynieść poważne szkody. Wdrożenie tej koncepcji było niemożliwe z dwóch podstawowych przyczyn:

— wyrównanego pod względem ambicji, wiedzy i umiejętności zespołu. Wprowadzenie nowych hierarchicznych struktur organizacyjnych mogłoby zburzyć atmosferę dobrej wzajemnej współpracy, gdyż prawdopodobnie członkowie nowo utworzonych grup programowych niechętnie poddaliby się kierownictwu swych dotychczasowych kolegów;

— trudności w uzyskaniu dodatkowych środków na wypłacenie programistom wiodącym ekwiwalentu pieniężnego za pełnienie przez nich funkcji kierowniczych.

W FSO zastosowano rozwiązanie zastępcze, a mianowicie zespoły programisty wiodącego powoływano doraźnie do oprogramowania konkretnego projektu. Programistą wiodącym zostawał członek zespołu w danym momencie obciążony pracą, do którego dokooptowywano kilku programistów. Jednocześnie programiści wiodący byli członkami grup, którymi nie kierowali. Ten model organizacji sprzyja kształtowaniu atmosfery współpracy, dając jednocześnie możliwość wyróżnienia się bardziej ambitnym pracownikom.

Nie ustrzeżono się pewnych błędów przy wdrażaniu jednego z elementów IPT, a mianowicie techniki analizy i oceny programu, tzw. walks-through, czyli analitycznej dyskusji na temat programu w oparciu o zasadę tzw. „burzy mózgów". Podstawowym błędem, jaki popełniło kierownictwo, była chęć zinstytucjonalizowania tych spotkań dyskusyjnych. Usiłowano w ten sposób przyspieszyć wdrażanie, jak się później okazało — z miernym skutkiem. Nie wykorzystano panującego od dawna wśród programistów zwyczaju korzystania z pomocy kolegów. W tym przypadku wystarczyło podać tylko kilka obowiązujących zasad, jakimi należy się kierować podczas tych koleżeńskich dyskusji, aby stały się one bardziej owocne.

Niejako samorzutnie, tzn. bez nacisku ze strony kierownictwa, rozpowszechnił się pseudokod, aczkolwiek część programistów w dalszym ciągu preferuje schematy blokowe.

Ostatnio wprowadzono wymóg posiadania udokumentowanego projektu rozwiązania programu. Dokumentacja może być w dowolnej formie, nawet w brudnopisie. Dzięki temu można w szerszym gronie omówić proponowane rozwiązania, co z jednej strony eliminuje ewentualne błędy, a z drugiej skłania programistę do zwrócenia większej uwagi na wstępne etapy pracy nad programem, mające decydujące znaczenie dla jego struktury oraz niezawodności i elastyczności. Dokumentacja ta jest traktowana jako robocza i nie jest archiwizowana.

Trzeba stwierdzić, że wprowadzenie metody programowania strukturalnego okazało się z perspektywy 3—4 lat bardzo efektywne i przyniosło wiele korzyści. Oszczędności, jakie osiągnięto dzięki unowocześnieniu metody programowania, przewyższają znacznie nakłady poniesione na ten cel. Są one porównywalne, a nawet przewyższają efekty niektórych zainstalowanych w tym czasie systemów aplikacyjnych.

Efekty te oraz doświadczenia w zakresie unowocześniania i wprowadzania bardziej wydajnych metod pracy są szczególnie istotne obecnie, w perspektywie znacznych ograniczeń inwestycyjnych w informatyce. W tej sytuacji konieczne jest lepsze wykorzystywanie posiadanego sprzętu, a przede wszystkim zasobów kadrowych. W tej dziedzinie mamy jeszcze znaczne rezerwy.

WALDEMAR KAPUŚCIK
Zakładowy Ośrodek Informatyki
Huty Metali Nieżelaznych „Szopienice"
Katowice

Eksploatacja baz danych w hucie „Szopienice"

Głównym zadaniem Zakładowego Ośrodka Inormatyki (ZOI) w Hucie Metali Nieżelaznych „Szopienice", od momentu jego powstania w 1974 r., było opracowanie i wdrożenie systemu planowania i kontroli produkcji dla Wydziału Walcowni Taśm, wyposażonej w nowoczesne urządzenia i technologie do produkcji blach oraz taśm z miedzi i mosiądzu. Uruchomienie sprzętu komputerowego ZOI nastąpiło w I kwartale 1978 r., natomiast eksploatacja pierwszych podsystemów (przyjmowanie zamówień, wysyłka materiałów gotowych) rozpoczęła się 1 stycznia 1979 r.

Posiadany sprzęt komputerowy (UNIVAC 1106 jako procesor główny i UNIVAC 6145 jako procesor komunikacyjny typu „front-end") pozwala użytkownikowi realizować — bezpośrednio na wydziale produkcyjnym — pracę w czasie rzeczywistym w oparciu o dużą bazę danych, zawierającą: dane technologiczne, informacje o zamówieniach i stanie ich realizacji a także raporty pracy urządzeń. Organizacja systemu planowania dla wspomnianej walcowni, jak i innych wdrożonych później systemów („Namiarowanie i korekta wsadu w piecach topielnych dla Wydziału Wałków i Tulei" oraz „Planowanie i kontrola produkcji dla Wydziału Wałków i Tulei") opiera się całkowicie na bazie danych, do utworzenia i eksploatacji której używa się standardowego pakietu DMS-1100.

Ponad dwuletnia eksploatacja baz danych w rygorystycznym reżimie produkcyjnym (praca na trzy zmiany łącznie z niedzielami i świętami, aktualizacja bazy przez osoby nie będące informatykami z terminali zlokalizowanych na stanowiskach produkcyjnych) pozwoliła na zebranie szeregu doświadczeń, które pokrótce pragnę przedstawić.

BAZA DANYCH

System Zarządzania Bazą Danych (SZBD) DMS-1100 opiera się na koncepcji Komitetu CODASYL, realizowanej na komputerach UNIVAC serii 1100. Bazą danych wg tej koncepcji jest grupa uporządkowanych zbiorów o następujących cechach:

- ze zbiorów może korzystać wiele programów, nawet równocześnie
- dane w zbiorach są zorganizowane z uwzględnieniem ich logicznej współzależności
- zbiory są zapamiętane w pamięci masowej
- dane zmieniają się w czasie
- z danych korzystają głównie programy działające w czasie rzeczywistym lub trybie konwersacyjnym.

Ostatnie ograniczenie wiąże się z obserwowaną tendencją powrotu do przetwarzania tradycyjnego w trybie wsadowym, natomiast bazy danych powinny być w pierwszym rzędzie stosowane w systemach wymagających szybkiego dostępu do danych. Wynika to z wyższego niż w tradycyjnych systemach kosztu przechowywania danych, na który składają się: koszt wysoce specjalistycznego oprogramowania, koszt dodatkowego szkolenia projektantów i programistów, wreszcie — koszty technologiczne (np. większa złożoność programów czy wydłużony czas pracy systemu).

GENEROWANIE SZBD

Generowanie SZBD polega, z grubsza biorąc, na dopasowaniu standardowych możliwości oferowanych przez producenta do realnych potrzeb i możliwości użytkownika. Chodzi tutaj zarówno o wybór procedur tworzących system, jak i określenie wielkości systemowych tablic i buforów. Pozwala to na optymalny wybór wielkości SZBD dla danej konfiguracji, co ma szczególnie duże znaczenie, zwłaszcza w przypadkach małej pojemności pamięci operacyjnej. Dla przykładu — wygenerowany w „Szopienicach" SZBD zajmuje ok. 33 K słów 36-bitowych, podczas gdy w innych ośrodkach ma on średnią wielkość w granicach 50—55 K słów. Osiągnięto to przez ograniczenie liczby typów alokacji rekordów (rezygnacja z metody indeksowo-sekwencyjnej), zmniejszenie liczby typów grup logicznych (ang. *set*) oraz zadeklarowanie odpowiednio małych tablic. Na wielkość niektórych tablic mają wpływ przyjęte założenia eksploatacyjne, jak np. liczba równolegle eksploatowanych baz czy liczba równolegle pracujących programów; na inne — wpływa fizyczna struktura bazy, głównie zaś wielkości stron (bloków).

Celem lepszego wykorzystania przez różne programy buforów przeznaczonych na strony zaleca się ujednolicenie wielkości stron we wszystkich obszarach lub — co najwyżej — używanie dwóch ich wielkości (np. 448 słów i 224 słów). Należy pamiętać o tym, że wielkość tych buforów ma decydujący wpływ na czas opracowania transakcji, więc należy wygenerować je tak duże, jak jest to możliwe.

Do generowania SZBD zaliczyć można również wprowadzanie poprawek czy przeróbek w standardowych programach systemu. Chodzi tutaj zarówno o usunięcie ewidentnych błędów (w Hucie przykładem tego może być program LONGRECOVERY), jak i o własne pomysły zwiększające możliwości systemu. Wśród nich można wymienić modyfikację programu TIMER, rozszerzenie funkcji COMPACT z DMU, czy wprowadzenie własnej taśmy śladowej TIPLOG.

Warta omówienia jest modyfikacja programu TIMER, sterującego pracą całego systemu DMR (Data Management Routine). Gdy system ten pracował wieloprogramowo (ang. *multithread*), DMR był cały czas przechowywany w pamięci operacyjnej, blokując ją dla innych celów (np. kompilacji nowych programów). Wprowadzona poprawka polega na sprawdzaniu przez TIMER liczby zgłoszeń do systemu i w zależności od niej powoduje zachowanie bądź zwalnianie DMR z pamięci. Zwiększa to znacznie przepustowość systemu, a przy małej pamięci operacyjnej umożliwia wręcz jego normalne funkcjonowanie.

Mgr Waldemar KAPUSCIK ukończył w 1971 r. studia na Wydziale Matematyki, Fizyki i Chemii Uniwersytetu Śląskiego. W 1974 r. rozpoczął pracę w Zakładowym Ośrodku Informatyki HMN „Szopienice", gdzie obecnie zajmuje stanowisko kierownika Działu Eksploatacji. Od początku pracy w HMN odpowiedzialny za całość spraw związanych z bazami danych eksploatowanych systemów. Uczestnik kilku szkoleń zorganizowanych przez firmę UNIVAC (m.in. tematy: system operacyjny, metodyka projektowania, języki programowe). W 1980 r. otrzymał nagrodę (zespołową) Sekretarza Naukowego PAN za opracowanie i wdrożenie „Systemu namiarowania i korekty wsadu w piecach topielnych dla Wydziału Wałków i Tulei".

SZKOLENIE PRACOWNIKÓW

Warunkiem prawidłowej eksploatacji baz danych jest wyszkolenie różnych grup korzystających z niej pracowników, a przede wszystkim Administratora Bazy Danych. Jest to bowiem osoba (lub grupa osób) odpowiedzialna za całość spraw związanych z bazą danych, począwszy od jej projektowania, aż po eksploatację. Szkolenie Administratora powinno obejmować następujące zagadnienia:

- język definiowania danych (DDL)
- język manipulacji danych (DML)
- wewnętrzna struktura SZBD
- metodyka projektowania sytemów opartych na bazie danych.

Z powyższego zakresu tematów wynika, że organizatorem szkoleń Administratora powinien być producent sprzętu. Z kolei tak przygotowany Administrator potrafi już samodzielnie organizować szkolenia projektantów, programistów i operatorów. Istotne jest również zapewnienie przez producenta możliwości stałej konsultacji, zwłaszcza w pierwszym okresie testowania i wdrażania bazy. Należy podkreślić, że w przypadku Huty producent (tj. firma UNIVAC) wywiązała się dobrze ze swych w tym zakresie zobowiązań. Oprócz ww. szkoleń specjaliści firmy przeprowadzili szkolenie programistów i operatorów.

Szkolenie projektantów polegało głównie na zapoznaniu ich z nowym narzędziem pracy — bazą danych, jej możliwościami i ograniczeniami. Duży nacisk położono na minimalizację czasu pracy programów już na etapie projektowania. Na potrzeby operatorów opracowano szereg instrukcji obsługi oraz przekazywano im dokumentacje eksploatacyjne gotowych systemów. Należy też podkreślić konieczność ciągłego nadzoru ich pracy przez Administratora. Nadzór ten polega m.in. na kontroli dziennika komputerowego i analizie wydruków z konsoli systemowej, organizowaniu spotkań, na których omawiane są typowe przypadki błędów, oraz organizowaniu dalszych szkoleń i instruktaży.

METODYKA PROJEKTOWANIA

W Hucie eksploatowanych jest obecnie pięć baz danych dla różnych systemów użytkowych. O ile w projekcie pierwszej bazy starano się wykorzystać jak najwięcej standardowych możliwości SZBD (co jest naturalne i zrozumiałe), to w miarę nabierania doświadczenia, przeprowadzania konsultacji i wysuwania wniosków wynikających z eksploatacji, zmieniono diametralnie stosunek do projektowania. W istniejącym systemie komputerowym pierwszoplanowym zadaniem było skrócenie czasu transakcji, co w znacznym stopniu uzyskano już na etapie projektowania bazy.

Cechą nowego podejścia do projektowania jest także lepsze dopasowanie struktury danych — i to zarówno struktury logicznej jak i fizycznej — do funkcji systemu. Narzędziem pracy projektanta stała się metoda obliczania liczby logicznych dostępów do bazy [1]. Warte podkreślenia jest szerokie stosowanie wygodnych standardów dokumentujących pracę, jak np. „Tablica pól rekordów i częstotliwości ich użycia", „Tablica kosztów pracy transakcji systemu", tablice służące do obliczeń struktury fizycznej.

Projektowanie bazy, wykorzystywaną w Hucie metodą, można podzielić na następujące etapy:

- zebranie żądań użytkownika, zdefiniowanie i uszeregowanie funkcji Systemu (wraz z nazwami pól i częstotliwościami ich użycia)
- opracowanie dla każdej funkcji schematu powiązań logicznych między poszczególnymi polami
- opracowanie zbiorczej struktury powiązań pól w całym systemie
- obliczenie częstotliwości używania wszystkich pól w systemie
- konsolidacja pól w rekordy na podstawie wielkości pól i częstotliwości ich użycia
- konstrukcja wstępnego projektu struktury logicznej bazy
- obliczenie kosztów obsługi poszczególnych transakcji systemu (kosztu w sensie liczby dostępów do bazy)
- wyznaczenie „wąskich gardeł" systemu i poprawienie wstępnej struktury logicznej bazy (wybór właściwych wskaźników w grupach, redundancja danych, konsolidacja rekordów w większe jednostki, tworzenie nowych grup logicznych, wprowadzenie rekordów zmiennej długości zamiast grup itp.).

ŁADOWANIE POCZĄTKOWE

Po zdefiniowaniu struktury bazy danych, przygotowaniu schematu i zainicjowaniu wszystkich obszarów następuje ładowanie bazy rekordami nagłówkowymi, kontrolnymi, stałymi tablicami i katalogami oraz rekordami obrazującymi początkowy stan obiektów systemu. Nosi to nazwę ładowania początkowego bazy danych. Istotna jest w nim kolejność ładowania w poszczególnych obszarach oraz zdefiniowanie punktów restartowych w tej procedurze. Pierwszymi rekordami zapamiętanymi w bazie muszą być rekordy o dostępie bezpośrednim, aby uniknąć zajęcia ich adresów. Następnie należy załadować wszystkie rekordy nagłówkowe (właściciele grup), potem rekordy im podporządkowane (członkowie).

Należy pamiętać o możliwości tylko częściowego, a nie całkowitego załadowania poszczególnych stron (np. w 70%), gdyż znacznie ułatwi to późniejszą eksploatację, a dokładniej mówiąc — dopisywanie nowych rekordów do bazy. Można również stosować interwałową metodę rozmieszczania rekordów właściciela, polegającą na rozdzielaniu stron (zawierających dany typ rekordu) określoną liczbą stron pustych. Metoda ta jest przydatna wtedy, gdy o zapamiętywaniu rekordów członkowskich decyduje tzw. strategia najbliższego miejsca, polegająca na tym, że rekord członkowski umieszczany jest tak blisko rekordu właściciela, jak jest to możliwe, a więc na tej samej co on lub na najbliższych stronach.

W przypadku wprowadzania do bazy dużej liczby rekordów, istotnym problemem jest skrócenie czasu trwania tej operacji. Można to osiągnąć przez użycie do ładowania specjalnie przygotowanego schematu, innego niż używany potem w eksploatacji. Oczywiście różnice między tymi schematami mogą być tylko w sferze logicznej, gdyż fizyczny wygląd rekordów i ich wszystkich wskaźników musi być zachowany bez zmian. Podstawą przyspieszenia ładowania jest:

- wstępne przesortowanie rekordów danych
- wstępne wybranie i odrzucenie rekordów — duplikatów (o ile takie są zdefiniowane)
- zmiana sposobu alokacji rekordów (np. wszędzie na „direct")
- zmiana metody wyboru właściwego wystąpienia grupy logicznej
- zmiana metody wstawiania rekordu do wybranego już wystąpienia grupy (np. wstawianie rekordu nowego jako rekord pierwszy w grupie).

ZABEZPIECZENIE BAZY DANYCH

Specyfika eksploatacji w czasie rzeczywistym systemów z bazą danych polega na konieczności utrzymania ciągłej funkcjonalnej gotowości bazy oraz zapewnienia zawartych w niej informacji. W razie powstania błędu w bazie eksploatacja programów jest ograniczona lub w ogóle niemożliwa — aż do momentu powrotu do stanu poprawnego. Stąd wynika wielkie znaczenie ochrony danych przed zniszczeniem [4]. Standardowe możliwości DMS w tym zakresie są następujące:

- **zbiór QLF** (Quick-Look File), w którym zapisywane są strony przed dokonaniem na nich zmian; pozwala on na usunięcie skutków działania programu (ang. *rollback*) oraz szybkiego odtwarzania bazy (ang. *quick recovery*) do stanu sprzed awarii;
- **taśma śladowa ATT** (Audit-Trail Tape), na której zapisywane są wszystkie dokonane w bazie zmiany. Taśma ta wraz z „dumpem" bazy może służyć do tzw. długiego odtwarzania bazy (ang. *long recovery*), procedury pozwalającej na doprowadzenie stanu bazy do określonego momentu w przeszłości;
- **„dump" bazy danych**, będący statystycznym obrazem skopiowanej na taśmie magnetycznej bazy w określonym czasie.

Aparat ten, teoretycznie wystarczający, okazał się jednak zawodny w warunkach ciągłej eksploatacji taśm (24 godz. na dobę). Okazało się bowiem, że w wyniku fizycznego uszkodzenia taśmy z „dumpem" lub ATT nie jest możliwe wykonanie długiego odtwarzania. Zwiększenie niezawodności uzyskano przez wprowadzenie do systemu jeszcze jednej taśmy śladowej. Jest nią taśma systemu komunikacyjnego TIPLOG, na której zapisywane są kopie wszystkich transakcji aktualizujących bazę. Taśma ta oraz specjalnie opracowane programy pozwalają na ponowne automatyczne uruchomienie przebiegu transakcji za ustalony ubiegły okres. Może to mieć znaczenie przy błędnym zakończeniu procedury długiego odtwarzania. Metodę ponownego uruchomienia przebiegu transakcji można użyć (nawet po poprawnym zakończeniu procedury) do doprowadzenia aktualnego stanu bazy od ostatniego punktu czasowego do momentu wystąpienia awarii. Oprócz tego taśmy TIPLOG można używać do analizy obciążenia systemu transakcjami aktualizującymi oraz do kontroli wejściowych monitorów ekranowych celem wychwycenia ewentualnych błędów (rys. 1).

UŻYCIE DMU

DMU (Data Management Utility) jest wygodnym narzędziem pracy Administratora Bazy, pozwalającym na szybką jej kontrolę [2]. Jest to specjalnie uprzywilejowany program, realizujący pewne, niedostępne poprzez DML operacje na bazie danych. Szczególnie użytecznymi funkcjami DMU są:

- PRINT — wydruk dowolnej strony bazy w postaci rozpakowanej, z opisem rekordów i wskaźników
- PATCH — zmiana dowolnego słowa na dowolnej stronie bazy
- VERIFY — kontrola powiązań w określonej grupie logicznej lub łańcuchu obliczeniowym.

Istotnym z punktu widzenia prawidłowej eksploatacji jest regularne sprawdzanie wszystkich powiązań w bazie (poprzez VERIFY), gdyż pozwala to na szybkie wykrycie ewentualnych błędów. W Hucie procedura ta wykonywana jest codziennie na zmianie nocnej. Z kolei funkcja PATCH używana jest do szybkiego poprawiania drobnych błędów w bazie (błędów typu zły wskaźnik lub zły nagłówek rekordu). W uzasadnionych przypadkach może ona służyć także do usunięcia lub przepięcia błędnych rekordów. Należy podkreślić, że w Hucie opracowano specjalne procedury ułatwiające używanie tych funkcji. Procedury te opierają się na zastosowaniu własnego programu CONSREAD, służącego do wprowadzania danych z konsoli operatorskiej. Pozwalają one Administratorowi na konwersacyjną pracę DMU.

WYKORZYSTANIE ATT DO ANALIZY PRACY PROGRAMÓW

Na taśmie śladowej DMS zapisywane są wszystkie informacje o pracy całego systemu, takie jak: początek i koniec programów aktualizujących, zmiany w bazie — wykonane przez te programy (ang. *after-look*) oraz punkty czasowe (ang. *checkpoint*). Znając dokładnie układ bloków taśmy oraz wewnętrzną strukturę schematu bazy można opracować szereg programów pozwalających na przeglądanie taśmy ATT. Najprostszymi z nich będą programy tworzące wydruki wszystkich bądź wybranych bloków z taśmy.

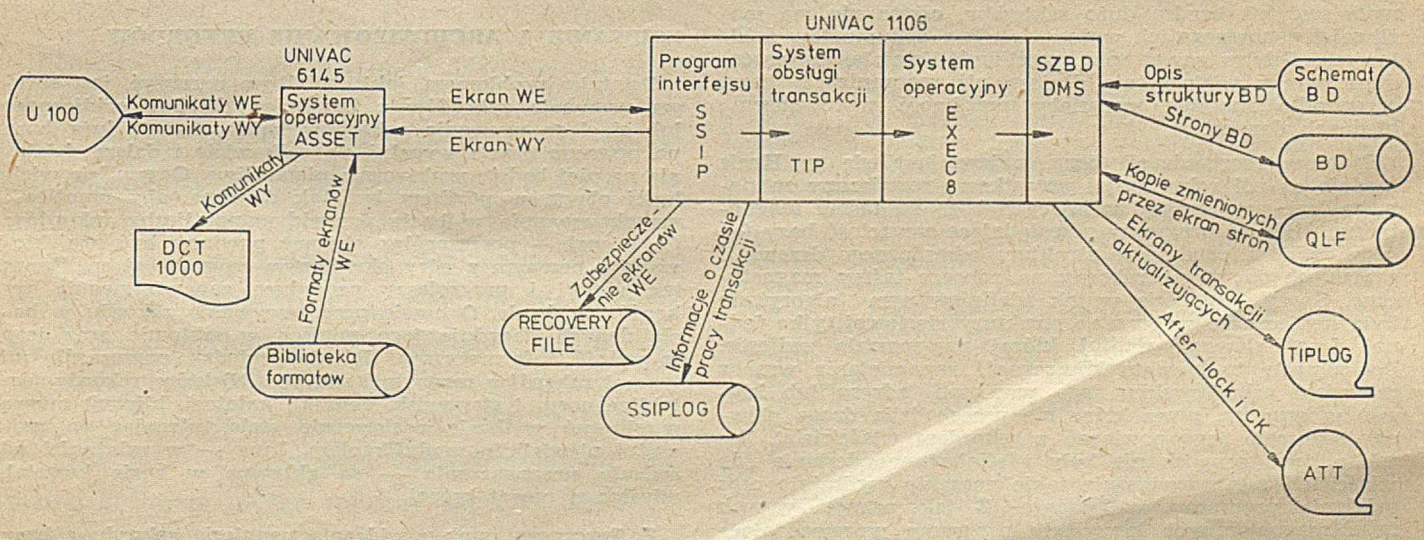
Z taśmy tej można również wyciągnąć szereg szczegółowych informacji, jak np. wydruki wszystkich zmian wykonanych przez określony program, historię zmian określonej strony bazy czy nawet historię modyfikacji wskazanego rekordu. To ostatnie jest szczególnie przydatne do śledzenia przyczyn powstania błędu w bazie. Praktycznie jest to jedyna możliwość uzyskania odpowiedzi na pytania: kto i kiedy zmienił dany rekord czy usunął rekord wskazany. Oczywiście — pytania dotyczą programów, a nie osób. W systemie zawierającym ok. 150 różnych transakcji startowanych asynchronicznie z 32 terminali programy śledzące taśmę ATT niejednokrotnie przyczyniły się do wykrycia błędów w eksploatowanych już programach. Co więcej, niejako przy okazji wykryto szereg formalnych błędów w zapisie na ATT. Został o tym poinformowany producent poprzez specjalne formularze zgłoszeń SUR (Software User Report).

SYSTEM KONTROLI I MODYFIKACJI BAZ DANYCH

Podczas eksploatacji systemów z bazą danych zachodzi często potrzeba szybkiej kontroli lub modyfikacji wskazanych rekordów, a nawet niektórych ich pól [5]. Czynności te można częściowo wykonać za pomocą systemu DMU (VERIFY, PATCH), ale przeważnie w tym celu pisano proste, jednorazowe programy. Wykonywały one elementarne czynności na rekordach bazy (odczyt, usunięcie, przepięcie). Liczba takich programów po rocznej eksploatacji bazy urosła do ok. 50, lecz nie usuwano ich po użyciu — ze względu na możliwość ich przyszłego wykorzystania.

Do wykonania szybkiej kontroli i modyfikacji bazy danych (bez potrzeby pisania programów) producent przewidział system QLP (Query Language Processor). Z uwagi na stosunkowo małą pamięć operacyjną komputera w Hucie, nie można było wygenerować i używać tego systemu. Stało się to powodem opracowania własnego pakietu programów BADACZ, dla każdej bazy danych systemu. Programy tworzące ten pakiet zostały wygenerowane automatycznie poprzez procesor SSG (Symbolic Stream Generator) z przygotowanych uprzednio programów bazowych i danych, zawartych w każdym schemacie.

Programy wchodzące w skład pakietu pozwalają na wykonanie na rekordach bazy takich funkcji, jak: ich odczyt, kontrola grup, zapamiętanie i modyfikacja rekordu, obliczanie liczby rekordów w bazie oraz określanie wielkości wolnego miejsca. Dzięki tym ostatnim funkcjom Administrator może śledzić stopień zapełnienia bazy i decydować o ewentualnym usuwaniu rekordów najstarszych. Programy pakietu BADACZ działają w trybie konwersacyjnym z konsoli operatorskiej, a ich obsługa jest prosta i wygodna.



Rys. 1. Obieg informacji w systemie komputerowym UNIVAC 1106/6145

KONTROLA OBCIĄŻENIA SYSTEMU TRANSAKCJAMI

W systemach wielodostępnych czasu rzeczywistego poważnym zadaniem jest śledzenie zmiennego — w czasie — obciążenia systemu. Pozwala ono na wykrycie „wąskich gardeł" systemu oraz transakcji najbardziej obciążających system. Początkowo do realizacji tej funkcji używano taśmy śladowej systemu komunikacyjnego TIPLOG, na której jednak były zapisywane jedynie ekrany wejściowe transakcji aktualizujących. Pozwalało to co prawda na kontrolę samych ekranów wejściowych (co mogło mieć znaczenie przy szukaniu przyczyn błędów), ale nie mogło służyć do oceny obciążenia całego systemu. Dużą część transakcji stanowią bowiem transakcje odczytujące zawartość bazy. W związku z tym założono specjalny zbiór dyskowy o nazwie SSIPLOG, do którego zapisywane są informacje o realizacji każdej transakcji w systemie. Informacjami tymi są: kod transakcji, a także czasy rozpoczęcia oraz zakończenia jej realizacji.

Programy przetwarzające zbiór SSIPLOG dają ogólne pojęcie o liczbie transakcji i średnim czasie jej trwania. Mogą więc one służyć do badania zmian działania systemu w wyniku reorganizacji bazy lub modyfikacji programów.

REORGANIZACJA BAZY DANYCH

Konieczność reorganizacji bazy danych jest problemem, z którym wcześniej czy później musi zetknąć się każdy Administrator Bazy [3]. Powodów reorganizacji można wymienić wiele. Przykładowo mogą to być: zmiany projektowe wynikłe z niemożności przewidzenia wszystkich tendencji rozwojowych systemu, błędy w projektowaniu lub naturalna tendencja do zwiększania się stopnia nieuporządkowania rekordów w bazie.

Nieuporządkowanie rekordów w bazie charakteryzuje się dużą liczbą rekordów na stronach nadmiarowych (obcych), co powoduje wydłużenie czasu opracowania transakcji. Chcąc poprawić sytuację należy wykonać tzw. reorganizację fizyczną, polegającą na zmianie rozmieszczenia rekordów w bazie bez zmiany ich zawartości. Z kolei reorganizacja logiczna polega na zmianie zawartości rekordów bądź ich połączeń między sobą. Najbardziej ogólnie problem reorganizacji logicznej można zdefiniować następująco: mając bazę danych z odpowiadającym jej schematem oraz nowy, zmieniony schemat, należy zmodyfikować dotychczasową bazę danych w myśl tego schematu. Tak postawiony problem jest jednak bardzo trudny do rozwiązania. Producent zapewnia co prawda możliwość użycia procesora DRU (Data Reorganization Utility) lub specjalnego programu REORG, opracowanego niedawno w BELL LABORATORIES (USA), ale po pierwsze — nie rozwiązują one jeszcze tak zdefiniowanego problemu, a po drugie — są to programy bardzo duże, na ogół nie mieszczące się w istniejącej małej pamięci operacyjnej. Nie bez znaczenia jest również koszt tych programów.

Oczywiście do zagadnienia reorganizacji bazy można podejść zupełnie inaczej (i tak właśnie początkowo uczyniono w Hucie), a mianowicie w taki sam sposób jak do zakładania początkowego. Jeżeli jednak w bazie zapamiętanych jest już bardzo dużo rekordów, często nie ma możliwości ponownego ich wprowadzenia z dokumentów źródłowych. Gdyby nawet taka możliwość istniała, to procedura byłaby wielokrotnie dłuższa niż reorganizacja bazy, a ponadto znacznie wzrosłaby możliwość wprowadzenia błędnych danych.

Problem reorganizacji bazy, pojawiający się w Hucie średnio co pół roku, został początkowo rozwiązany za pomocą tzw. programów zwijających bazę na taśmy magnetyczne. Były to programy przeglądające zawartość bazy (od góry do dołu, zgodnie ze strukturą odwróconego drzewa) i zapisujące odczytane rekordy na robocze taśmy magnetyczne. Sporą trudnością w tak zdefiniowanym algorytmie była konieczność wyboru ze struktury sieciowej tylko jednej drogi głównej, wzdłuż której następowało zwijanie. Powodowało to bowiem konieczność zapamiętania wraz z rekordem członkowskim jego powiązań (adresy lub klucze) w grupach, przez które nie przechodziła droga główna (zwłaszcza jego powiązań z rekordem właściciela). Jeżeli w tych grupach nie były zdefiniowane odesłania „do właściciela", to rozwiązanie takie znacznie wydłużało czas działania programu zwijającego. Z tak przygotowanych taśm magnetycznych korzystały programy wprowadzające nową już bazę danych.

Aby przyspieszyć proces reorganizacji, zamiast taśmy roboczej zaczęto używać zbiory dyskowe, a ściślej mówiąc — roboczą bazę danych. Z uwagi na to, że jeden program nie może jednocześnie obsługiwać dwóch baz danych, zastosowano rozwiązania polegające na tym, że do istniejącej bazy i schematu dopisano nowy obszar, równy całej nowej bazie danych (oczywiście z obiektami o zmienionych nazwach). Robocza podbaza nie posiadała jeszcze pełnej struktury nowej bazy, jednak same rekordy odpowiadały już w pełni nowym rekordom, a także istniały wszystkie niezbędne ich powiązania logiczne. Zastosowano tu szereg rozwiązań przyspieszających programy zwijające i ładujące, jak np. likwidacja dostępu bezpośredniego w rekordach członkowskich, zdefiniowanie wszystkich możliwych wskaźników w roboczych grupach, użycie interwałowej metody pamiętania rekordów właściciela i inne. Z kolei programy ładujące działały znów na podwójnej bazie, której jedną częścią była już nowa zreorganizowana baza, a drugą — uprzednio przygotowany obszar roboczy z rekordami przeznaczonymi do przekopiowania. Programy ładujące bazę w porównaniu z programami korzystającymi z taśm magnetycznych działały kilkakrotnie szybciej (czas zmniejszył się z 3—4 godz. do 15—20 min). W przypadku programów zwijających różnica ta była mniejsza. Ostatnią czynnością w tak przeprowadzonym procesie było przygotowanie nowego schematu bazy i fizyczne usunięcie obszaru roboczego.

SORTOWANIE REKORDÓW

Baza danych, aczkolwiek opracowana pod kątem wykorzystania jej przez programy transakcyjne, może również zawierać definicje ułatwiające pracę programom wsadowym. Typowym tego przykładem są grupy sortowane. Grupy takie znacznie jednak wydłużają czas pracy transakcji wstawiających do bazy nowe rekordy. Również w przypadku zdefiniowania bezpośredniego dostępu do rekordów członkowskich lub wysokiego stopnia nieuporządkowania bazy, korzyści ze zdefiniowania takich grup są problematyczne. Wtedy bowiem, mimo że grupa będzie posortowana, odczyt wszystkich jej rekordów będzie wymagał praktycznie wykonania tylu fizycznych operacji we/wy, ile jest rekordów w grupie. Spowoduje to — wbrew początkowym oczekiwaniom — znaczne wydłużenie czasu pracy programów. Stąd rodzi się konieczność sekwencyjnego przeglądania całych obszarów, celem wybrania z nich rekordów przeznaczonych do sortowania, a samo sortowanie — poza nielicznymi wyjątkami — winno być operacją zewnątrzną w stosunku do bazy danych. Do tego celu można używać oczywiście wielu rozwiązań standardowych dostarczonych przez producenta (np. czasownik SORT z języka COBOL, procesor SORT/MERGE dla komputerów serii UNIVAC-1100 itp.).

W Hucie opracowano własny program sortujący, który jest prostszy, a przez to znacznie szybszy niż programy standardowe. Algorytm tego programu opiera się na sortowaniu w pamięci operacyjnej dużych bloków zbioru, a następnie ich scalaniu. Sortowanie w pamięci operacyjnej polega na podziale bloku na grupy czteroelementowe oraz ich uporządkowaniu i kolejnym scalaniu — aż do osiągnięcia całego bloku.

USUWANIE I ARCHIWIZOWANIE REKORDÓW

Ostatnim problemem, jaki chciałbym poruszyć, jest sprawa usuwania rekordów z bazy danych. Bez cyklicznego lub ciągłego usuwania rekordów każda, nawet duża baza po pewnym czasie wypełni się całkowicie i dalsze działanie na niej będzie praktycznie niemożliwe. Oczywiście wielkości obszarów powinny być tak dobrane, aby umożliwić przechowywanie w bazie przewidywanej liczby rekordów. Co więcej — poszczególne obszary powinny być projektowane z pewnym z góry określonym zapasem stron. Zapas ten ułatwi i przyspieszy procedurę zapamiętywania rekordów w bazie. O czasie przechowywania rekordu w bazie danych decyduje jego miejsce w systemie oraz jego zawartość informatyczna. Różnice między poszczególnymi typami rekordów mogą być znaczne. Niektóre rekordy (np. tablice norm i stanowisk, cenniki, katalogi, klienci) muszą przebywać w bazie praktycznie stale, ulegając co najwyżej niewielkim modyfikacjom. Inne — w zależności od rodzaju przetwarzania — są aktywne w bazie kwartał, miesiąc, a nawet krócej.

Z fizycznego punktu widzenia usunięcie rekordu z bazy polega jedynie na zaznaczeniu tego rekordu jako nieaktyw-

nego i jego eliminacji ze wszystich grup logicznych, w których partycypował jako właściciel bądź jako członek. Faktyczne usuwanie rekordów nieaktywnych następuje w momencie próby zapisywania nowego rekordu na stronie zawierającej rekordy nieaktywne. Zwiększa to jednak czas trwania wprowadzenia rekordu bazy. Aby uniknąć tego, można — po usunięciu dużej partii rekordów — odpowiednio upakować wszystkie strony właściwych obszarów. Funkcję tę spełnia DMU poprzez instrukcję COMPACT. Powoduje ona fizyczną eliminację rekordów zaznaczonych do usunięcia oraz przesunięcia pozostałych rekordów na stronie ku górze celem utworzenia na stronie jednego ciągłego bloku wolnego miejsca.

Usuwanie rekordów z bazy nie jest problemem trudnym, aczkolwiek o jego prawidłowym, a zwłaszcza szybkim przebiegu decyduje poprawnie zaprojektowana struktura danych. Daleko istotniejsze wydaje się być jednak archiwizowanie usuwanych informacji celem ewentualnego późniejszego ich wykorzystania.

Do zagadnienia archiwacji można podejść dwojako. Pierwszy sposób to kopiowanie wszystkich usuwanych rekordów (w dotychczasowej czy nawet prostszej strukturze) do odpowiedniego zbioru, zdefiniowanego najczęściej na taśmach magnetycznych. Zbiór ten można wykorzystać w dowolnym czasie, celem dokładnego prześledzenia historii operacji na wybranym obiekcie systemu. Drugim sposobem jest przechowywanie odpowiednio skumulowanych wielkości w specjalnych rekordach bazy. Przykładowo rekordy te mogą zawierać takie informacje, jak: sumy przychodów i rozchodów, produkcja za ustalony okres, stany kont itp.

Często zdarza się, że z przeszłości użytkownika interesują jedynie tego rodzaju zagregowane wielkości, przy czym dostęp do tych informacji w bazie jest prawie natychmiastowy. W Hucie prace nad stworzeniem systemu archiwizacji są dopiero w toku. Jedynie w „Systemie Planowania i Kontroli Produkcji na Wydziale Wałków i Tulei" wprowadzono rekordy kontrolne, zawierające skumulowane wielkości produkcji w różnych przekrojach za okres miesiąca, kwartału i roku.

LITERATURA

[1] Kapuścik W.: Liczba logicznych dostępów w bazach danych typu CODASYL. Podstawy Sterowania, tom 10, z. 2, 1980

[2] Kapuścik W., Mrowiec Z.: DMU narzędziem pracy Administratora Baz Danych. Problemy Postępu Technicznego, nr 2/80

[3] Kapuścik W., Mrowiec Z.: Problemy reorganizacji baz danych. Problemy Postępu Technicznego, nr 4/80

[4] Praca zbiorowa: System zabezpieczenia i odzyskiwania informacji w HMN „Szopienice". Dokumentacja eksploatacyjna

[5] Praca zbiorowa: System kontroli i modyfikacji bez danych BADACZ. Dokumentacja eksploatacyjna.

JANUSZ ZALEWSKI
Instytut Badań Jądrowych
Warszawa

ADA — nowy język programowania (2)

Jednostki programowe i instrukcje

Poniższe omówienie oparto na podręczniku języka ADA [1], który zastąpił wstępne opracowanie opublikowane przed dwoma laty [2] i jest obecnie najdokładniejszym źródłem informacji o języku. W niniejszym artykule przedstawiając budowę języka zrezygnowano z formalizacji, zakładając, że przyczyni się to do łatwiejszego zrozumienia różnych konstrukcji językowych. Uproszczenie to jest zgodne z duchem podręcznika i celowe również dlatego, aby z językiem zapoznać jak najszerszą grupę informatyków. Czytelnikom zainteresowanym formalną definicją języka radzimy zapoznać się z dostępną literaturą, np. [3, 4].

JEDNOSTKI PROGRAMOWE

Podstawowymi elementami języka są tzw. jednostki programowe (ang. program units), zwane niekiedy modułami. Jednostkami programowymi są: podprogramy (procedury i funkcje), pakiety (ang. packages) i zadania (ang. tasks). W większości przypadków stanowią one jednostki kompilacji, a więc moduły, które można oddzielnie kompilować.

Wszystkie jednostki programowe mają zbliżoną budowę, mianowicie, składają się z części deklaracyjnej (która może być częściowo niewidoczna) i z ciała (ang. body).

Widoczna część deklaracji funkcji, zwana specyfikacją, może wyglądać następująco:

function BIN (argumenty oraz ich typy i wartości początkowe)
return (typ wartości funkcji); -- nagłówek deklaracji funkcji.

Zwróćmy uwagę, że średnik oznacza zakończenie instrukcji a podwójny myślnik — początek komentarza.

Specyfikację procedury zawierającej część niewidoczną (kompilowaną oddzielnie) oraz ciało procedury — przedstawiono poniżej.

```
procedure CFSA (ARG1 : INTEGER, ARG2 : INTEGER, ...) is
-- deklaracje
procedure WRT (W1 : INTEGER) is separate;
begin
-- ciało procedury
X : = 16 # 3D #; -- szesnastkowo
Y : = 61; -- dziesiętnie
end CFSA;
```

Znak : = oznacza instrukcję przypisania (ang. assignment), a znak # służy do oznaczenia podstawy systemu liczbowego, w którym zapisana jest liczba.

Parametry wejściowe podprogramów mogą mieć wartości początkowe domniemane (ang. default), np.

```
function BIN (ARGUMENT : INTEGER : = 8 # 177776 #;
procedure CFSA (ARG1 : INTEGER : = 0, ...);
```

Wywołanie procedury polega na użyciu jej nazwy z parametrami aktualnymi, np.

```
begin
  -- przykład użycia funkcji
  Y : = X+BIN (32);
  -- wywołanie procedury
  CFSA (AKT1, AKT2,...);
  -- inne instrukcje
end;
```

Kolejność parametrów aktualnych można zmienić przyporządkowując im wprost nowe wartości, np.

```
CFSA (ARG2 = > AKT2, ARG1 = > 0, ...);
```

Tak więc, parametry aktualne można podać w zapisie pozycyjnym (ang. positional), jak w poprzednim przykładzie lub — w zapisie bezpośrednim (ang. **named**).

Wszystkie użyte dotąd parametry są parametrami wejściowymi, co można zaznaczyć jawnie za pomocą klauzuli **in**. W przypadku parametrów wyjściowych lub wejściowo-wyjściowych użycie klauzuli **out** lub **in out** jest obowiązkowe, np.

```
CFSA (ARG1 : in : INTEGER : = 0,)
      ARG2 : in : INTEGER : = 16 # 3D #,
      ARG3 : out : INTEGER,
      ARG4 : in out : BOOLEAN);
```

Podstawowym modułem programowym jest pakiet, który definiuje zbiór danych, zmiennych, typów i podprogramów powiązanych logicznie w całość.

Część deklaracyjna pakietu stanowi jego sprzężenie z innymi używanymi programami, np. bibliotecznymi. Jeżeli specyfikacja zawiera podprogramy, to ich ciała muszą się znajdować w ciele pakietu. Jest możliwe ukrycie ciała przed użytkownikiem za pomocą klauzuli **separate**, np.

```
package NAZWA is
  -- specyfikacje
end;
package body NAZWA is separate;
```

Ciało pakietu musi być zdefiniowane oddzielnie, np.

```
separate
package body NAZWA is
  -- ciało pakietu
end NAZWA;
```

Warto zwrócić uwagę, że pakiet nie musi mieć ciała. Definiuje wtedy obszar, pole danych przeznaczone do wykorzystania przez inne programy.

Dostęp do zmiennych zadeklarowanych w ciele pakietu nie jest możliwy z zewnątrz, natomiast elementy specyfikacji są dostępne (pośrednio) przy użyciu zapisu z kropką, np.

```
X : = NAZWA.ZMIENNA; -- zmienna z pakietu NAZWA
```

Można również wykorzystać klauzulę **use**, którą omówimy później i instrukcję **renames**, np.

```
NOWA : REAL renames NAZWA.ZMIENNA;
```

Zadania stanowią jednostki programowe służące do realizacji procesów współbieżnych. Synchronizacja następuje za pomocą tzw. spotkania (ang. rendezvous), w którym biorą udział dwa zadania, oraz — instrukcji specjalnych. W zadaniu wywoływanym należy zadeklarować tzw. wejście, do czego służy instrukcja **entry**, np.

```
entry NAME; -- deklaracja w zadaniu wywoływanym
```

Zewnętrzny dostęp do zadania jest realizowany wywołaniem podobnym do wywołania procedury, np.

```
ZADANIE.NAME; -- użyte w zadaniu wywołującym
```

Przyjęcie wiadomości przesłanej wskutek użycia (w zadaniu wywołującym) nazwy NAME następuje przez wykonanie instrukcji **accept**, która musi znajdować się w ciele zadania wywoływanego i zapewnia synchronizację obu zadań.

```
accept NAME do -- w zdaniu wywoływanym
  -- pełna instrukcja accept
end;
```

Spotkanie (rendezvous) rozpoczyna się od instrukcji **accept** i trwa przez czas jej wykonania (od **do** do **end**). Wtedy następuje przekazanie parametrów, jeżeli takie są. Natomiast, zadanie wywołujące jest przez ten czas zawieszone.

Należy zaznaczyć, że spotkanie może nastąpić dopiero wtedy, gdy oba zadania są gotowe, tzn. zadanie wywołujące osiągnęło instrukcję ZADANIE.NAME, a zadanie wywoływane — instrukcję **accept** NAZWA. Jeżeli jedno z zadań nie jest gotowe, to drugie na tę gotowość oczekuje.

Ogólna postać zadania jest zbliżona do postaci innych jednostek programowych.

```
task ZADANIE is
  -- jedyną dopuszczalną deklaracją jest entry
  entry NAME;
end;
task body ZADANIE is
begin
  -- w tym miejscu następuje uaktywnienie zadania
  accept NAME (ARG1,...) do
    -- przekazanie parametrów
  end;
  -- inne instrukcje
end ZADANIE;
```

Bardziej szczegółowo omówimy zadania po przedstawieniu instrukcji. Dodajmy tylko, że wszystkie zadania zadeklarowane w danej jednostce programowej muszą być zakończone przed wyjściem z tej jednostki.

Bardziej szczegółowo omówimy zadania po przedstawieklarowania i obsługi, w każdej jednostce programowej, tzw. wypadków (ang. exceptions), tj. sytuacji szczególnych, takich jak błędy, pułapki, przerwania itp. Istnieje specjalny mechanizm umożliwiający reakcję na te sytuacje, zaistniałe podczas wykonywania programu. Wypadek (np. o nazwie ERROR) powinien być zadeklarowany w części specyfikacyjnej, np.

```
package PAKIET is
  ERROR: exception;
  -- inne deklaracje
end PAKIET;
```

Jeżeli zostanie stwierdzony, uprzednio zadeklarowany wypadek, to sterowanie jest przekazywane natychmiast do programu obsługi, który powinien określać działania zapobiegawcze.

```
begin
  -- instrukcje programu
  exception
    when ERROR = > INSTRUKCJA; -- działanie zapobiegawcze
end;
```

INSTRUKCJE

Część wykonawczą każdej jednostki programowej musi zawierać instrukcje. Dotychczas poznaliśmy jedynie niektóre, tj. instrukcję przypisania (: =), instrukcję bloku (**begin...end**), instrukcję wywołania procedury, a także instrukcje **return, renames** oraz dwie — odnoszące się do zadań: **entry** i **accept**. Obecnie przedstawimy dalsze, dzieląc je na instrukcje proste i złożone.

Instrukcje proste

Najczęściej używaną instrukcją jest bez wątpienia instrukcja przypisania, która przypisuje nazwie wartość wyrażenia. Instrukcja wywołania procedury stanowi nazwę procedury uprzednio zadeklarowanej. Działanie instrukcji pustej **null** polega na przejściu do wykonywania instrukcji następnej.

Istnieją trzy proste instrukcje sterujące: **return**, instrukcja wyjścia **exit** i instrukcja skoku **goto**, który może nastąpić jedynie wewnątrz modułu w ograniczonym zakresie. Choć na ogół instrukcje są wykonywane w takiej kolejności, w jakiej występują w programie, to trzy wymienione instrukcje sterujące (oraz instrukcja **raise**, którą omówimy razem z instrukcjami złożonymi) mogą spowodować zmianę tej kolejności.

Instrukcje złożone

Do instrukcji złożonych zalicza się m.in. instrukcje sterujące: **if, case** i **loop.**

Instrukcja **if** ma postać:

```
if WARUNEK
  then INSTRUKCJA;
  elsif WARUNEK1
  then INSTRUKCJA1;
  ...
  else INSTRUKCJAN;
end if;
```

na przykład,

```
if X < 0
  then X : = —X; - - wartość bezwzględna
end if;
```

Instrukcja **case** może mieć postać następującą:

```
case WYRAŻENIE is
  when WARTOSC = > INSTRUKCJA;
  when WARTOSCI/WARTOSC2 = > INSTRUKCJA1;
  when 1..30 = > INSTRUKCJA2;
  when others = > INSTRUKCJA3;
end case;
```

Znak / (kreska pionowa) oznacza alternatywę wartości wyrażenia, a zapis A..B — zakres wartości od A do B. Warunek zwany **others,** stanowiący zbiór warunków nie wymienionych uprzednio, musi wystąpić jako ostatni.

Instrukcja **loop** ma kilka postaci:

```
loop
  - - instrukcje
end loop;
```

lub

```
while WARUNEK
loop
  - - instrukcje
end loop;
```

lub

```
for INDEKS in ZAKRES
loop
  - - instrukcje
end loop;
```

Wyjście z pętli może nastąpić ponadto wskutek wykonania instrukcji

```
exit PETLA when WARUNEK;
```

na przykład,

```
PETLA
while NEXT / = null
loop
  - - instrukcje
  exit PETLA when WARUNEK;
end loop;
```

Użycie pętli w programie jest podobne jak w innych znanych językach,

```
SUM : = 0;
for I in 1..100
loop
  SUM : = SUM+X (I);
end loop;
```

Do instrukcji złożonych należy także blok, który w języku ADA spełnia kilka funkcji, m.in. służy do deklarowania zmiennych lokalnych, ukończenia zadań (jak pamiętamy, nie można wyjść z bloku przed zakończeniem wszystkich zadań) oraz obsługi wypadków (program obsługi może znajdować się tylko w bloku stanowiącym część funkcji, procedury, pakietu lub zadania).

```
BLOK - - blok może mieć nazwę
declare
  - - nieobowiązkowa część deklaracyjna
begin
  - - ciąg instrukcji
end;
```

Instrukcje odnoszące się do wypadków powinny być umieszczone na końcu bloku, np.

```
begin
  - - instrukcje
  exception
  when NUMERIC ERROR/CONSTRAINT_ERROR = >
  INSTRUKCJA;
  when others = > raise MY_FAILURE;
end;
```

Znak _ (podkreślenie) symbolizuje odstęp i jest pełnoprawnym znakiem alfabetu podstawowego, obejmującego łącznie 56 znaków.

Instrukcja **raise** służy do programowania obsługi wypadków. Obsługa może odbywać się na tym samym poziomie, gdzie powstał wypadek lub na poziomie wyższym, tzn. można zaprogramować obsługę wypadków przez blok wyższej warstwy, np. jeżeli w bloku, gdzie powstał wypadek, nie ma programu obsługi. Wypadki, które nie zostały obsłużone, podlegają tzw. propagacji (wstecz) przez kolejne warstwy jednostki programowej aż do miejsca istnienia programu obsługi.

Sposób reagowania na wypadki ilustrują poniższe przykłady.

```
- - przykład 1
if I > LIMIT
then
  raise ERROR; - - ogłoszenie wypadku (bez obsługi)
end if;
- - przykład 2
function „%" (U, V : REAL) return REAL is
begin
  return U/V;
  exception
    when NUMERIC_ERROR = > return REAL'LARGE;
    - - obsługa
    end „%";
```

Zapis REAL'LARGE oznacza największą liczbę reprezentowaną w komputerze i jego interpretacja jest zdefiniowana systemowo (p. omówienie typów).

Niektóre wypadki (np. użyte powyżej NUMERIC ERROR, CONSTRAINT_ERROR) przewidziano w definicji języka, a ich obsługa następuje automatycznie (implicit **raise**). Jednak, wypadki obsługiwane systemowo mogą być definiowane przez użytkownika.

Ogłoszenie wypadku można porównać ze zwykłym wywołaniem procedury bezparametrowej i — z powrotem do wykonania programu wywołującego. Jednakże, przy obsłudze wypadków, zarówno moduł, w którym wypadek nastąpił, jak i moduł, gdzie wypadek został obsłużony, nie kończą się normalnie lecz sterowanie powraca do wyższej warstwy — nie do miejca, gdzie powstał wypadek.

Nie omówiliśmy jeszcze kilku instrukcji związanych z zadaniami.

Może się zdarzyć, że aktywnych jest kilka zadań wywołujących jedno wejście **entry.** Zatem, z każdym wejściem zadania wywoływanego musi być związana kolejka zadań wywołujących (obsługiwanych). Kolejka jest obsługiwana na zasadzie FIFO.

W takich wypadkach używa się instrukcji **select,** która zawiera rozgałęzienia zaczynające się od **accept** i połączone przez **or,** tzn.

```
select
  accept ... do
  ...
  end;
or
  accept ... do
  ...
  end;
end select;
```

Przed instrukcją **select** można użyć tzw. dozoru (ang. guard), który jest wyrażeniem uzależniającym wykonanie instrukcji, np.

```
select
  when DOZOR = > WARTOSC
  accept ... do
  ...
  end;
or
  when DOZOR1 = > WARTOSC1
  accept ... do
  ...
  end;
or
  terminate;
end select;
```

W takim przypadku najpierw następuje zbadanie wszystkich warunków klauzuli **when.** Warunki spełnione umożliwiają dostęp do instrukcji **accept** zwanych z tego powodu otwartymi (ang. open **accept**). Jeżeli nie ma wywołania dotyczącego otwartych instrukcji **accept,** to następuje oczekiwanie oraz obsługa pierwszego przychodzącego zgłoszenia. Jeżeli jest kilka zadań oczekujących na obsługę przez otwarte **accept,** to wybierane jest jedno we-

dług zasady niedeterministycznej. Należy to rozumieć tak, że sposób szeregowania nie jest przypadkowy, lecz nie ma wpływu na programy użytkowe.

Instrukcja **terminate** w powyższym przykładzie oznacza zakończenie zadania w przypadku wyjścia z bloku, w którym ono się znajduje. Inną instrukcją często używaną w zadaniach jest **delay**, która oznacza oczekiwanie, np.

```
delay 1.0; -- odczekaj 1s
```

Warto wiedzieć, że instrukcja **select** może występować także w zadaniu wywołującym, np.

```
select
ZADANIE.NAME;
else
  N := NAME'COUNT; -- liczba zadań w kolejce do
  wejścia NAME
end select;
```

Jednakże mechanizm spotkań jest niesymetryczny, tzn. zadanie wywołane nie ma możliwości takich jak wywołujące — i na odwrót.

• • •

W następnym numerze przedstawimy podstawowe w języku ADA pojęcie typu, które w dużym stopniu decyduje o możliwościach języka. Nie jest to zwykła kolejność omawiania budowy języka, gdyż najczęściej autorzy zaczynają nauczanie właśnie od przedstawienia struktur danych. Jednak, uczyniliśmy tak celowo, aby umożliwić czytelnikom, nieco inne od upowszechnionego, praktyczne spojrzenie na język programowania.

W przyszłym numerze omówimy ponadto nowoczesne konstrukcje językowe, takie jak moduły generyczne, przeciążanie operatorów, które różnią język ADA od innych — na bazie których on powstał.

LITERATURA

[1] Reference Manual for the Ada Programming Language Proposed Standard Document, US Department of Defense, Washington, DC, July 1980

[2] Ichbiah J., et al.: Preliminary Ada Reference Manual, SIGPLAN Notices, Vol. 14, No. 6, Pt. A, June 1979

[3] Bjoerner D., Oest O. N., eds.: Toward a Formal Description of Ada, Lecture Notes in Computer Science, Vol. 98, Springer-Verlag, Berlin, 1980

[4] Donzeau-Gouge V., Kahn G., Lang B.: On the Formal Description of Ada, Proc. Workshop Semantics-Oriented Complier Generation, Aarhus, Denmark, 14—18 January 1980, N. D. Jones, ed., Lecture Notes in Computer Ccience, Vol. 94, Springer-Verlag, Berlin, 1980.

ALGORYTMY

Procedura wyszukująca dany wzorzec w tekście

Problem wyszukiwania wzorca $W = w_1...w_m$ w tekście $T = t_1...t_n$ (ang. pattern matching) polega na znalezieniu takiego indeksu j, dla którego fragment $t_j...t_{j+m-1}$ tekstu T jest równy wzorcowi W, przy czym j jest najmniejszym takim indeksem. Jeśli takie j nie istnieje, to definiujemy $j = 0$.

MOTYWACJA

Procedura wyszukująca dany wzorzec w tekście jest podstawową częścią programu służącego do poprawiania i redagowania tekstów (*text editora*). Program taki daje z reguły szeroką gamę możliwości. Na przykład — użytkownik może zażądać zastąpienia słowa s_1 słowem s_2 w pewnym tekście, wyszukania wszystkich wystąpień słowa s_1 w tym tekście itp. Ponieważ wiele systemów operacyjnych nie posiada podobnego programu, programista, chcąc w sposób istotny ułatwić sobie pracę, zmuszony jest do pisania go samodzielnie. W związku z tych chcemy mu dostarczyć bardzo efektywny algorytm Knutha, Morrisa i Pratta rozwiązujący ten problem.

Procedura może znaleźć również zastosowanie wszędzie tam, gdzie niezbędne jest wyszukiwanie w tekście wyróżnionych identyfikatorów, a więc — na przykład — w systemach konwersacyjnych z dostępem w języku (quasi) naturalnym.

OPIS ALGORYTMU

Rozważmy najpierw tzw. problem *Prefix matchingu* (por. Aho A., Hopcroft E., Ullman J.: The Desigr and Analysis of Computer Algorithms. Addison-Wesley 1976, algorytm 9.2):
dla danego wzorca $W[1:m] = w_1...w_m$ obliczyć wartości funkcji f zdefiniowanej następująco:

$f(j)$ = największa liczba s mniejsza od j, taka, że $w_1...w_s = w_{j-s+1}...w_j$, jeżeli takie s nie istnieje, to 0.

Jeżeli na przykład wzorzec jest równy *aabbaab* to funkcja f opisana jest w tabeli.

i	1	2	3	4	5	6	7
f (i)	0	1	0	0	1	2	3

Oznaczmy przez $f^k(j) = \underbrace{f(f(...f}_{k \text{ razy}}(j)...))$.

Algorytm działa następująco:

Definiujemy $f(1) = 0$.

Załóżmy, że obliczyliśmy już $f(1), f(2),...,f(j)$. Niech $f(j) = i$.

Aby obliczyć $f(j+1)$, porównujemy w_{j+1} oraz w_{i+1}. Jeśli $w_{j+1} = w_{i+1}$, to $f(j+1) = i+1$ (ponieważ $w_1w_2...w_{i+1} = w_{j-i+1}w_{j-i+2}...w_jw_{j+1}$), w przeciwnym przypadku znajdujemy najmniejsze q dla którego:

$fq(j) = u_iw_{j+1} = w_{u+1}$, albo $fq(j) = 0$ i $w_{f+1} \neq w_i$.

W pierwszym przypadku $f(j+1) = u+1$, w drugim $f(j+1) = 0$.

Algorytm ten zapisany jest jako procedura *Prefix matching*. Może być ona z powodzeniem użyta do budowy interesującego nas algorytmu (wprowadzenie funkcji f służy do uniknięcia zbędnych porównań). W celu ułatwienia czytelnikowi zrozumienia tego rozwiązania, załóżmy, że wzorzec i tekst zostały połączone w jedną całość w następujący sposób:

$w_1...w_mXt_1...t_n$, gdzie X-znak nie występujący wśród $w_1,...,w_m$, $t_1...,t_n$.
Zakładając, że tak uzyskany ciąg znaków potraktowalibyśmy jako wzorzec w procedurze *Prefix matching*, uzyskalibyśmy:
$f(j) = m$ wtedy i tylko wtedy, gdy j jest takim indeksem, dla którego $w_1...w_m = t_{j-m+1}...t_j$.
Stąd wynika, że $(j-2*m)$ jest rozwiązaniem wyżej postawionego zadania wtedy i tylko wtedy, gdy j jest najmniejszą taką liczbą, przy której $f(j) = m$. Jeżeli takie j nie istnieje, to dany wzorzec nie występuje w rozpatrywanym tekście.

Ponieważ zapamiętywanie wszystkich wartości $f(k)$ dla $k > m$ oraz konkatenacja wzorca i tekstu są zbędne, poniższa procedura napisana jest tak, aby przy zachowaniu koncepcji uniknąć strat pamięci.

STRUKTURA DANYCH

Tekst oraz wzorzec pamiętane są w tablicach znakowych *T [1:n], W [1:m]*, gdzie n jest długością tekstu, m — długością wzorca.

Ponieważ teksty spotykane w praktyce najczęściej podzielone są na linijki, wystarczy, aby rozmiar tablicy T był równy długości linijki. Po przeszukaniu każdej linijki należy wczytać następną i postępowanie powtórzyć, oczywiście nie wliczając ponownie wartości funkcji f.
Używana będzie także pomocnicza tablica F [1 : m], służąca do zapamiętania wartości funkcji f.

OPIS PROCEDURY PATTERN MATCHING

Parametrami procedury funkcyjnej Pattern matching są dwie tablice znakowe *T [1 : n]* (tekst), *W [1 : m]* (wzorzec) oraz ich zakresy n i m. Wynikiem funkcji jest j, zdefiniowane w opisie problemu. Liczba instrukcji wykonywanych podczas działania procedury jest wprost proporcjonalna do $(m+n)$.

```
integer procedure Pattern matching (T, n, W, m); integer n, m;
character array W, T;
comment procedura znajduje pierwsze wystąpienie wzorca W w
tekście T. Jeżeli W nie występuje w T, to wartością funkcji jest
0;
begin integer array F [1:m]; integer i, j, fj;
 procedure Prefix matching;
 comment Prefix matching oblicza wartości funkcji f;
 begin
 F [1]: = 0;
 for j: = 2 step 1 until m do
 begin
 i: = F[j—1];
 while W [j] ≠ W [i + 1] & i > 0 do i: = F [i];
 if W [j] ≠ W [i + 1] & i = 0 then F [j]: = 0 else F [j]: = i + 1
 end
 end Prefix matching;
Prefix matching;
Pattern matching: = i: = 0;
for j: = 1 step 1 until n do
begin
 while T [j] ≠ W [i + 1] & i > 0 do i: = F [i];
if T [j] ≠ W [i+1] & i = 0 then fj: = 0 else fj: = i+1;
 if fj = m then
 begin Pattern matching: = j—m+1;
 go to exit
 end;
 i: = fj
end;
exit:
end Pattern matching;
```

Procedurę tę można zmodyfikować tak, aby znajdowała wszystkie wystąpienia wzorca w tekście. W tym celu należy w trakcie wykonywania procedury wyprowadzać wszystkie j, dla których $fj = m$ oraz instrukcję go to *exit* zastąpić instrukcją *fj: = F [m]*.

ZBIGNIEW ŚWIRSKI
ANDRZEJ SZAŁAS

Informatyka na studiach pielęgniarskich

Od kilku lat informatyka jest przedmiotem nauczania na studiach medycznych, w tym także na studiach pielęgniarskich. Wydziały pielęgniarskie akademii medycznych istnieją: od 1969 r. w Lublinie, od 6—7 lat — w Katowicach, Poznaniu i Krakowie, a od niedawna także we Wrocławiu. Program studiów obejmuje przedmiot „Podstawy informatyki" na IV (ostatnim) roku w wymiarze 30 godzin wykładów. Przewidziana liczba godzin zajęć (na obecnym etapie kształcenia wystarczająca) pozwala na realizację przedmiotu, którego nazwa winna brzmieć: „Propedeutyka informatyki" lub „Propedeutyka informatyki medycznej". Żadna z tych dwóch nazw nie przyjęła się dotychczas w ministerialnym programie studiów, chociaż może być uwzględniona w nowym programie.

Jeżeli chodzi o formy zajęć, to w realizowanym przez Zakład Organizacji Pracy Pielęgniarskiej AM w Krakowie programie nauczania na Wydziale Pielęgniarskim, w ramach 30 godzin zajęć uwzględnia się cztery godz. ćwiczeń, prowadzonych bezpośrednio w ośrodku obliczeniowym. Nasuwa się przypuszczenie, że pozostałe wydziały podobnie „radzą" sobie z nauczaniem przedmiotu.

Brak jednolitego, obowiązującego wszystkie wydziały pielęgniarskie programu nauczania powoduje, że każdy wydział realizuje własny program. Zebrane obserwacje i doświadczenia pozwolą zapewne na wypracowanie jednolitego programu i przygotowanie skryptu, a może nawet podręcznika (opracowany w Zakładzie Informatyki Medycznej AM w Krakowie pod redakcją prof. J. Trąbki skrypt „Propedeutyka informatyki medycznej" dość dobrze, choć nie w pełni „przystaje" do realizowanego przez nas programu).

Zachęcając do dyskusji nad nauczaniem informatyki na studiach medycz-

nych, przedstawiamy program przedmiotu opracowany przez Zakład Organizacji Pracy Pielęgniarskiej i realizowany na Wydziale Pielęgniarskim AM w Krakowie. W programie można wyróżnić dwie zasadnicze części: propedeutyka informatyki oraz zastosowania informatyki w medycynie.

I. Propedeutyka informatyki:

- Definicja i przedmiot informatyki. Zarys rozwoju maszyn cyfrowych. Mała informatyka i komputerowa technika obliczeniowa.
- Pojęcie informacji i jej pomiar.
- Kodowanie, przechowywanie i odczytywanie informacji. Nośniki informacji. Techniki i urządzenia wprowadzania i wyprowadzania danych (2 godziny zajęć praktycznych z tego tematu prowadzi się w ośrodku obliczeniowym).
- Algorytm — opis procesu przetwarzania danych.
- Arytmetyka dwójkowa.
- Struktura i zasady działania maszyny cyfrowej. System operacyjny.
- Wprowadzenie do programowania maszyn cyfrowych. Pakiety programów użytkowych.
- Języki programowania. Kompilacja programów.
- Ośrodek obliczeniowy — struktura i funkcje. Pokaz pracy maszyny cyfrowej i wykonania programu obliczeń statystycznych (2 godziny ćwiczeń w ośrodku obliczeniowym).

II. Zastosowania informatyki w medycynie:

- Medyczne systemy informatyczne.
- Systemy informatyczne zarządzania.
- Systemy komputerowego wspomagania diagnostycznego i decyzyjnego (na przykładzie DOLMEDU — Dolnośląskiego Centrum Diagnostyki Medycznej we Wrocławiu). Zastosowania komputerów w diagnozowaniu pacjenta (najwięcej uwagi poświęca się specjalizowanym maszynom cyfrowym, systemom monitorowania funkcji fizjologicznych organizmu w połączeniu z komputerową analizą danych, komputerowej analizie sygnałów kardiofaktograficznych, komputerowej analizie głosu przy użyciu intonografu oraz tomografii komputerowej).
- Organizacja ankiety i statystyczne opracowanie wyników badań.

Na każdy z powyższych tematów przeznacza się ok. dwóch godzin wykładów. Przekazywane treści ilustruje się planszami, schematami (np. struktura maszyny cyfrowej) i wykresami. Nośniki informacji omawia się na konkretnym materiale 5- i 8-ścieżkowych taśm dziurkowanych, kart dziurkowanych oraz taśm magnetycznych. Uwzględnienie w programie przedmiotu systemów informatycznych zarządzania oraz zagadnienia statystycznego opracowania wyników badań ma na celu stworzenie dodatkowej motywacji do nauczania oraz powiązanie tego przedmiotu z „Podstawami organizacji i kierowania" i „Statystyką", nauczanymi na II roku studiów.

Uwzględnienie czterech godzin wykładów na programowanie maszyn cyfrowych ma na celu poznanie istoty i znaczenia programowania komputerów raczej z punktu widzenia użytkownika niż programisty. Ogólnie, program zajęć był układany z zamiarem przygotowania absolwentów studiów pielęgniarskich do roli użytkowników systemów informatycznych. W pełni zatem podzielamy pogląd wyrażony przez M. Paprockiego odnośnie do nauczania informatyki na studiach farmaceutycznych (INFORMATYKA, nr 6/1979). Minimum wiedzy o charakterze encyklopedycznym ma na celu nie tylko poszerzenie horyzontów wiedzy u przyszłych magistrów pielęgniarstwa, ale przede wszystkim ma stanowić podstawę do wdrażania i wykorzystania systemów informatycznych, rozumienia istoty i znaczenia informatyki, celowości jej stosowania oraz ogromnej przydatności w badaniach medycznych i praktycznej działalności służby zdrowia. Dlatego nie wydaje się celowe poświęcanie większej liczby godzin na programowanie, a w tym nauczanie konkretnych języków programowania. Płytka wiedza i umiejętności w tym zakresie mogłyby okazać się gorsze niż niewiedza. Słuszniejszym wydaje się zwrócenie szczególnej uwagi na zastosowania informatyki w medycynie. Porywający jest dla studentów przykład tomografii komputerowej, jej wartości, niepowtarzalne możliwości i zalety.

Cel tego krótkiego opracowania zostanie w intencji autora spełniony, jeżeli zachęci do dyskusji oraz przedstawienia własnych poglądów i doświadczeń odnośnie do nauczania informatyki na studiach pielęgniarskich i — ogólnie — na wszystkich kierunkach studiów medycznych.

ZBIGNIEW KABAT
Akademia Medyczna
Zakład Organizacji
Pracy Pielęgniarskiej
Kraków

Zastosowanie informatyki w projektowaniu budownictwa

W dniach 10—12 maja 1981 r. odbyła się w Kudowie trzecia konferencja „Zastosowanie informatyki w projektowaniu budownictwa — INFOPRO'81". Konferencje INFOPRO organizowane w cyklu dwuletnim przeznaczone są głównie dla pracowników biur projektów, przedstawicieli uczelni technicznych i ośrodków obliczeniowych. Wylania się z tych obrad aktualny stan metod komputerowych wykorzystywanych w kraju w projektowaniu budownictwa.

Według analiz BISTYPU — 174 biura projektów (68% wszystkich biur) posiadają różnego rodzaju sprzęt komputerowy. Jeżeli weźmiemy przy tym pod uwagę fakt, że wiele z nich ma dogodne warunki korzystania z obcego sprzętu komputerowego, będzie można przyjąć, że zdecydowana większość projektantów ma dzisiaj możliwość łatwego korzystania z elektronicznej techniki obliczeniowej.

Biura projektów posiadają obecnie 25 komputerów, w tym 14 zainstalowanych w ostatnim pięcioleciu (R-32 — 3 szt., R-10 — 1 szt., ODRA 1305 — 5 szt., ODRA 1325 — 2 szt., ICL 2903 — 3 szt.), 10 maszyn stopniowo wycofywanych z eksploatacji (ODRA 1204, ZAM i ODRA 1003), a także jeden unikalny UNIVAC 90/60. Drugą grupę sprzętu stanowi 91 minikomputerów: MERA 400 — 35 szt. (wszystkie instalowane w ostatnich trzech latach), WANG 2200 — 46 szt., NOVA 840, 3D, 1200 — 5 szt., SM-3 — 1 szt., VARIAN — 2 szt., K-202 — 2 szt., a także 153 mikro- i minikomputerów (COMPUCORP — 116 szt., HEWLETT PACKARD — 7 szt., MERA 300 — 30 szt.). Do tego dochodzą jeszcze przestarzałe kalkulato-

ry programowane o niewielkich możliwościach obliczeniowych. Do potrzeb biur projektów najbardziej są dostosowane minikomputery, które obecnie stanowią 28% ogółu sprzętu.

Znacznie gorzej jest z urządzeniami do automatyzacji prac graficznych. Używa się zaledwie dziesięciu autokreślarek oraz pojedynczych grafoskopów i czytników rysunków (digitizery). W ostatnim roku rozpoczęto wprawdzie import autokreślarek DIGIGRAF z CSRS, są one jednak dla biur projektów relatywnie bardzo drogie.

Niezły jest natomiast stan oprogramowania. W dziedzinie konstrukcji budowlanych istnieje znaczna liczba programów, które w pełni zaspokajają potrzeby obliczeń statycznych oraz wymiarowania żelbetu i stali; natomiast stosunkowo nieliczne są programy do obliczeń dynamiki konstrukcji oraz zagadnień stateczności. W dziedzinie konstrukcji budowlanych istnieje grupa dobrych systemów automatyzacji projektowania (tj. obejmujących obliczenia, rysunki, zestawienia itp.). W dziedzinie instalacji przemysłowych i sanitarnych, istnieje szereg dobrych i powszechnie stosowanych programów do projektowania instalacji c.o., wodociągów i wentylacji. Prowadzone są prace dotyczące innych instalacji. Część tych programów ma już wyjścia graficzne.

W pełni zaspokajają potrzeby projektantów programy przeznaczone do obliczeń służących ochronie atmosfery. Obejmują one obliczenia i wykresy opadów i stężeń. Programy do projektowania instalacji elektrycznych są niestety mniej liczne i rzadziej stosowane.

W dziedzinie kosztorysowania zastosowanie komputerów jest dość ograniczone, gdyż aktualna metoda kosztorysowania i baza normatywna znacznie utrudniają szerszy rozwój komputeryzacji.

Oprogramowanie ODRA 1305, R-32, WANG 2200 i COMPUCORP jest obszerne i dalej rozbudowywane. ODRA 1204 ma duży zasób programów, który jednak ze względu na wycofanie tych komputerów z użytkowania nie jest rozwijany. MERA 400 ma — jak dotąd — jedynie część niezbędnego oprogramowania.

W prowadzonej przez BISTYP Bibliotece Programów i Systemów dla Projektowania zarejestrowano już 752 programów (bez programów dla minikomputerów WANG i COMPUCORP). W ostatnim roku użytkowano 239 programów (32%). Następuje bowiem pewnego rodzaju koncentracja zastosowań. Jest to zjawisko korzystne, ułatwiające dalsze upowszechnianie i szkolenie.

W biurach projektów informatyką zajmuje się na stałe ok. 2 tys. osób (ok. 2—3% łącznego stanu zatrudnienia). Podniósł się także poziom znajomości metod komputerowych wśród projektantów.

Na tle istniejącego poziomu informatyki w projektowaniu można próbować wyznaczyć dalsze kierunki działania. Jednym z istotnych zjawisk, które trzeba przy tym uwzględnić, jest przywrócenie działania praw ekonomicznych w gospodarce. Należy się więc liczyć z koniecznością dokonywania gruntownej analizy ekonomicznej zastosowań informatyki także w biurach projektów. Będzie to wpływać na politykę inwestycyjną i dlatego należy się liczyć z przejściowym ograniczeniem zakupów sprzętu komputerowego przez biura.

W najbliższym czasie nacisk powinien być położony na uzupełnienie i rozbudowę istniejących konfiguracji sprzętu (a zwłaszcza MERY 400). Zestawy powinny być rozbudowane o moduły pamięci operacyjnej, pamięci dyskowych i taśmowych, drukarki, a ponadto urządzenia grafiki komputerowej (przede wszystkim autokreślarki). Rozbudowywane powinny być także istniejące konfiguracje mikrokomputerów COMPUCORP. Stałą troską biur powinno stać się wykorzystanie posiadanego sprzętu, mimo oczywistego — w wyniku ograniczenia całości inwestycji — zmniejszenia zapotrzebowania na projektowanie.

Pożądane byłoby włączenie się informatyków w prace związane z planowaniem i rozliczaniem produkcji w biurze projektów, gospodarką finansową, listami płac, ewidencją kadr oraz wyszukiwaniem informacji w zbiorach INTE. Zespoły informatyków — złożone w wielu przypadkach z osób o dużym doświadczeniu — powinny się szerzej włączać w procesy projektowania, przy czym zespoły silniejsze powinny podejmować się wykonania prac eksportowych.

Zależnie od częstotliwości obliczeń, stopnia trudności korzystania z poszczególnych programów, a także od możliwości przełamania oporów psychicznych przy posługiwaniu się komputerem — powinny być stosowane elastycznie systemy: usługowy (zespół informatyków wykonuje obliczenia) lub samoobsługowy (wyszkoleni projektanci sami korzystają ze sprzętu komputerowego).

Metody komputerowe powinny być w znacznie większym niż dotychczas stopniu związane z innymi metodami usprawniającymi projektowanie, np. z techniką mikrofilmową, z nowocześnie zorganizowanymi zbiorami informacji czy projektowaniem katalogowym.

Oczywiście, niezbędne są również prace badawczo-rozwojowe, które w latach 1976—1980 dotyczyły głównie wprowadzenia metod informatycznych do projektowania i to w znacznej mierze w oparciu o metody wcześniej stosowane w projektowaniu tradycyjnym. Prace zamierzone na lata 1981—1985 mają doprowadzić do osiągnięcia nowej, wyższej jakości automatyzacji projektowania. Są to prace wyprzedzające i przygotowujące zastosowania powszechne. Głównymi kierunkami działań powinny być:

- **Problemy trudne,** tzn. takie, które bez użycia komputerów stwarzały zbyt duże trudności w obliczeniach. Planuje się prace nad obliczeniami uwzględniającymi nieliniowość geometryczną i fizyczną oraz szersze uwzględnienie dynamiki konstrukcji. We wszystkich dziedzinach i specjalnościach projektowania powinien nastąpić rozwój metod optymalizacyjnych.

- **Łączenie systemów projektowania i realizacji,** tzn. tworzenie związków pomiędzy systemami informatycznymi projektowania i systemami stosowanymi w organizacji, zarządzaniu i sterowaniu produkcją budowlaną. Jednym z podstawowych warunków postępu w tej dziedzinie jest uporządkowanie (w skali krajowej) bazy cen, norm i klasyfikacji.

- **Rozwój grafiki komputerowej** (wejść i wyjść graficznych). Szersze efekty w tej dziedzinie są zależne od upowszechniania się sprzętu graficzwego.

- **Prace podstawowe nad systemami** projektowania wspomaganego przez komputery. Istniejące systemy na ogół rozwiązują skomplikowane zagadnienia obliczeniowe, jednakże są jeszcze dość prymitywne z punktu widzenia metodologii wspomagania komputerowego.

- **Organizacja i zarządzanie biurem projektów** jest obszarem, w którym zastosowanie metod informatycznych dopiero się zaczyna.

- **Komputerowe wspomaganie zbiorów informacji naukowej i technicznej,** ewentualnie połączone z zastosowaniem techniki mikrofilmowej. W tej dziedzinie nie należy jednak oczekiwać szybkich efektów.

* * *

W konferencji INFOPRO'81 — podczas której omówiono powyższe zagadnienia — wzięło udział ok. 200 osób z całego kraju. Organizatorem były: Komisja Koordynacji Ogólnobranżowej Projektowania w Budownictwie, Centralny Ośrodek Badawczo-Projektowy Budownictwa Przemysłowego (COB-PBP) „BISTYP", Główna Komisja Informatyki Zarządu Głównego Polskiego Związku Inżynierów i Techników Budownictwa oraz Oddział Wrocławski PZITB. Obrady prowadzone były w sesjach tematycznych poświęconych:

- projektowaniu graficznemu
- systemom łączącym projektowanie, realizację i kosztorysowanie
- konstrukcjom budowlanym
- instalacjom sanitarnym i przemysłowym
- instalacjom elektrycznym
- organizacji i zarządzaniu biurem projektów
- zagadnieniom wdrażania.

W czasie trwania Konferencji odbyły się zebrania Klubu Użytkowników MERA 400 w projektowaniu i Klubu Użytkowników kalkulatorów programowanych COMPUCORP. Warszawskie Przedsiębiorstwo Informatyki Przemysłu Budowlanego „ETOB" zorganizowało pokaz działania urządzeń końcowych systemu wielodostępnego i teletransmisji (za pomocą których wykonywano obliczenia projektowe na komputerach w Warszawie i Gdańsku).

Zwrócono uwagę władz na wysoce niepokojący stan przemysłu komputerowego, który nie jest w stanie zapewnić nawet utrzymania istniejącego potencjału sprzętowego, podczas gdy powinien umożliwiać szybki i wszechstronny postęp w informatyzacji życia gospodarczego.

MACIEJ ROBAKIEWICZ
COB-PBP „BISTYP"
Warszawa

Oczami studenta

W lipcu 1980 r. spędziliśmy (my, tzn. grupa studentów z Koła Naukowego Informatyków) dwa tygodnie w Stanach Zjednoczonych. Odwiedziliśmy tam kilka wyższych uczelni: Uniwersytet Maryland koło Waszyngtonu, prywatny Uniwersytet Carnegie-Mellon w Pittsburgu oraz stanowy uniwersytet w Buffalo. Zwiedziliśmy również filię firmy IBM — Thomas J. Watson Research Center pod Nowym Jorkiem. Warto tu może wspomnieć, że trzy lata temu nasi starsi koledzy, (Wydział Informatyki na Uniwersytecie Warszawskim), w tym samym celu (zaliczenie zawodowych praktyk studenckich) przez trzy tygodnie podróżowali po Danii, RFN i Szwajcarii (por. INFORMATYKA nr 4—7/1978) oraz że młodsi koledzy planowali na ten rok również ciekawe spotkanie ze światową informatyką.

Uniwersytet jako ośrodek żywej myśli powinien umożliwić swoim studentom kontakt z tym, co najnowsze. Nasze zagraniczne praktyki zawodowe spełniły w pewnym stopniu to zadanie, a w każdym razie uświadomiły nam czym w istocie jest uczelnia otwarta, doceniająca nowości. Na amerykańską informatykę cały świat patrzy z uznaniem. Nic dziwnego więc, że uczucie zazdrości nie było nam zupełnie obce.

Silne wrażenie zrobiły na nas wydziały informatyczne na uczelniach amerykańskich, a w szczególności ich wyposażenie. Nie jesteśmy przyzwyczajeni do takiej ilości i jakości sprzętu komputerowego. Przede wszystkim wrażenie robiła naturalność, z jaką Amerykanie go przyjmują i używają. Przywykli do techniki, akceptują ją, starając się wykorzystać wszystkie jej zalety, a ujemne wpływy wyeliminować. Nie ma tam komputerów na pokaz. Duże znaczenie ma przy tym wysoki poziom kultury technicznej, jaki reprezentuje przeciętny obywatel. Już dzieci w szkole przyzwyczajane są do technicznych osiągnięć cywilizacji. Przykładem — choć nie najbardziej reprezentatywnym — może być fakt, że zdobywają one tam cenną umiejętność pisania na maszynie, co bardzo przydałoby się studentom w Polsce — szczególnie tym, którzy godzinami okupują dziurkarki na pierwszym roku studiów.

Czego spodziewa się w Polsce absolwent szkoły średniej, który wybrał kierunek Informatyka? Jeśli nie ma bujnej wyobraźni, to trudno mu chyba sprecyzować, a często nawet określić z grubsza, czemu posłuży jego przyszła wiedza, jak będzie mógł wykorzystać zdobyte doświadczenie. Codziennym życiem w Polsce nie zawładnęła jeszcze informatyczna gorączka (która notabene wcale nie musi być świadectwem choroby), nie widać na każdym kroku dobrodziejstw informatyki. Stąd pytanie: czy taki człowiek ocenia studia pod kątem przyszłej pracy, przydatności dla społeczeństwa? Czy ma po temu jakieś dane? Spróbujemy na te pytania odpowiedzieć. Oczywiście — nie z socjologicznego punktu widzenia, ale z pozycji tych, którzy przed takimi problemami parę lat temu stanęli. Odpowiedzi te są tym ważniejsze, że absolwenci wydziału Informatyki napotykają coraz większe trudności w znalezieniu odpowiedniego miejsca pracy.

Niewielu zdających na ten kierunek ma pojęcie, co konkretnego można po tych studiach robić. Słyszał jedynie, że to rozwijająca się dziedzina, niezgłębiona studnia wszelkich zastosowań, a na fachowców istnieje ciągłe zapotrzebowanie. Wszystko prawda, tylko tych zastosowań u nas prawie nie widać. Skąd więc wiara, że informatyk w Polsce jest tak potrzebny? Bo przecież każdy kandydat na nasz wydział tę wiarę żywi. I jest wśród nich wielu takich, którzy idąc za głosem serca trafiliby raczej na matematykę lub fizykę teoretyczną. To oczywiste, że w tej sytuacji istnieje spore prawdopodobieństwo pomyłki w wyborze kierunku, a także późniejszej zmiany profilu studiów. W tym kontekście system studiów amerykańskich, który daje studentom pełną swobodę rozwoju, rozszerzania i pogłębiania zainteresowań trudnych do przewidzenia przez młodego człowieka tuż po szkole, jawi się modelem godnym naśladowania.

Informatyka zaś jako nauka interdyscyplinarna doskonale do takiego modelu pasuje, co zobaczyliśmy na uczelniach amerykańskich. Uczelniach, które nie tylko pozwalają, ale też zachęcają do łączenia jej z innymi dziedzinami nauki i techniki. Traktują informatykę jako narzędzie, nie pozbawiając jej jednocześnie statusu odrębnej nauki. Uczą jak tym narzędziem fachowo władać — i w praktyce, i w naukach na pozór zupełnie abstrakcyjnych.

Powie ktoś, że w Polsce też są takie próby. Są, ale nie zdają egzaminu. Wiadomo, że na wielu wydziałach istnieje wykład mający coś wspólnego z informatyką. Prawdopodobnie jego celem jest wskazanie studentom, jak technika wkracza w ich dziedzinę i w jaki sposób można ją wykorzystać (a studenci ci nierzadko nie mają nawet pojęcia czym jest komputer). Efekty są na ogół żałosne. Wiemy o tym z przeprowadzonych rozmów z kolegami, skazanymi na tego rodzaju spotkania z informatyką, a często też z praktyk zawodowych, podczas których mamy kontakt z ludźmi pracującymi w tzw. zastosowaniach (o ich niekompetencji krążą na Wydziale anegdotki i legendy).

Być może właśnie Wydział Informatyki powinien być bardziej otwarty, elastyczny. Spójrzmy na taki wydział na uniwersytecie stanowym w Buffalo. Prowadzone tam są m.in. badania nad przetwarzaniem obrazów w medycynie. Łączą one, na bazie informatyki, specjalistów z różnych dziedzin, dają szanse pracy czysto nauko-

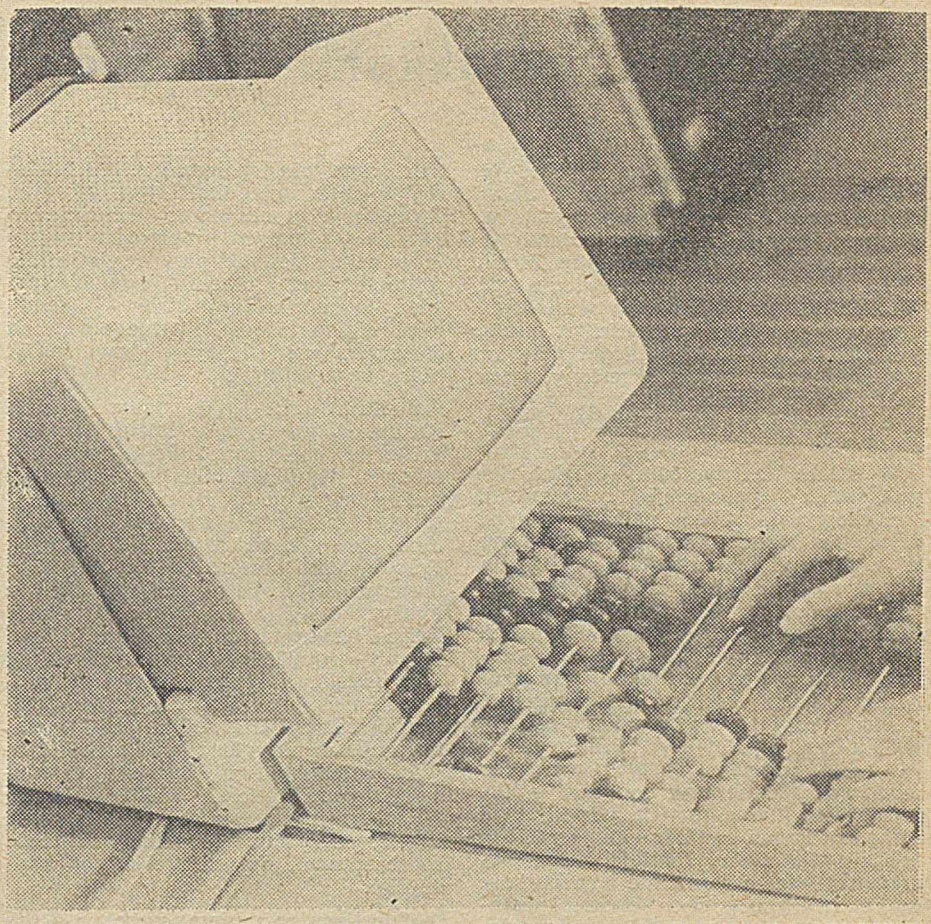
wej i praktycznej. Aktualnie prowadzone badania związane są przede wszystkim z tomografią komputerową. Jest to proces ujawniania wewnętrznej struktury ciała z wielu zdjęć rentgenowskich. Nowa metodologia i teoria, rozwijana przez grupę badawczą z Buffalo, zrewolucjonizowała diagnostykę radiologiczną ostatnich kilku lat. Badania nad tomografią komputerową i problemami jej pokrewnymi zazębiają się z różnymi dziedzinami informatyki: wewnętrzną strukturę wylicza się używając metod numerycznych, wielka liczba danych (średnio ok. 100 tys. słów do jednego obliczenia) wymaga umiejętności organizacji i zarządzania, do wyświetlania wyników używa się grafiki komputerowej, a do dużej ilości obliczeń — konieczne są mikroprogramowalne urządzenia specjalistyczne.

Drugi znaczący przykład współistnienia kilku dziedzin znaleźliśmy także w Buffalo. Grupa studentów powzięła ciekawą decyzję zbudowania całego komputera od początku własnymi siłami, i to zarówno jego części sprzętowej, jak i oprogramowania. Jeden z członków tej grupy zademonstrował opracowywany przez siebie program. Użytkownik wprowadzał opis pewnych faktów, potrzebne definicje i twierdzenia w języku naturalnym. Następnie zadawał pytania również w języku naturalnym. Jeśli coś było dla maszyny niezrozumiałe, to pojawiała się prośba o wyjaśnienie.

Taka forma studiowania — wynikająca z powyższych przykładów — wydaje się bardzo interesująca. Daje studentom możliwość autentycznej konfrontacji tego **czego się uczą** z tym **po co się uczą**. W ten sposób mogą sprawdzić swoje umiejętności i doznać satysfakcji z posiadanej wiedzy. Akademicka informatyka może w takim wydaniu wychować sobie prawdziwych i pożytecznych miłośników. W stanowym uniwersytecie w Buffalo jest ich wielu, ale pod tym względem — chyba jednym z niewielu — Wydział Informatyki UW mu nie ustępuje.

Rodzi się więc pytanie: czy nie można byłoby poprzez stworzenie bardziej elastycznej formy studiów dać szansę również polskim studentom? Niech by mogli jeszcze przed podjęciem pracy zdecydować, czy pragną zajmować się czystą informatyką, czy też wolą od razu kształcić się pod pewnym kątem — połączyć wiedzę informatyczną z medycyną, biologią, elektroniką, chemią, socjologią, językoznawstwem czy muzyką lub plastyką.

O mariażu z muzyką — na przykład — słyszeliśmy na Uniwersytecie Carnegie-Mellon (CMU). Była to najbogatsza i najciekawsza uczelnia, jaką zwiedziliśmy w USA. Wśród tamtejszych studentów popularne są różne formy uprawiania muzyki. Jeden z nich, pracujący w grupie zajmującej się systemem operacyjnym dla maszyny powstałej na CMU (mającej w ubiegłym roku 50 procesorów) urządził na uczelni osobliwy koncert. Chcąc osiągnąć efekt otaczania słuchaczy przez muzykę rozmieścił członków zespołu w różnych odległych od siebie miejscach, wśród budynków Uniwersytetu. Z powodu dzielących ich dużych odległości grający nie mieli ze sobą bezpośredniego kontaktu. Rolę dyrygenta przejął minikomputer z zestawem terminali.



Fot. JERZY SZCZĘSNY

Takie studiowanie byłoby chyba ciekawsze, a z całą pewnością kształciłoby większą samodzielność w zdobywaniu wiedzy, dawałoby dużo satysfakcji i zmniejszało prawdopodobieństwo pomyłki w wyborze kierunku. Półśrodkiem do osiągnięcia tego celu mogłyby być wykłady o charakterze interdyscyplinarnym dla studentów informatyki. Nie jest bowiem łatwo uzyskać teraz zgodę na zaliczanie przedmiotów na innym wydziale. Wykłady te, jako monograficzne, mogłyby być zresztą dostępne nie tylko studentom informatyki, ale również zainteresowanym słuchaczom innych kierunków. Dobrą ideą wydają się też seminaria tematyczne i tzw. grupy robocze pracujące nad konkretnymi problemami. Grupy te stanowiłyby pole współdziałania studentów, kadry naukowej i ludzi zawodowo zajmujących się danymi zagadnieniami.

Ten sposób realizowania interdyscyplinarności informatyki byłby bardzo korzystny nie tylko z punktu widzenia rozwoju współpracujących nauk. Studentom pozwoliłby wyjść poza sztywne ramy programu studiów, po ich ukończeniu byliby na pewno lepiej przygotowani do zawodu; kadrze naukowej uświadomiłby dokładniej oczekiwania i potrzeby zarówno studentów, jak i ludzi stosujących swą wiedzę w praktyce poza uczelnią; tym ostatnim zaś udostępniłby m.in. najnowsze badania, rozwiązania i eksperymenty prowadzone w uczelni.

Efektem i przykładem takiego podejścia do informatyki w Ameryce jest np. firma IBM. Przygarnia ona naukowców wszystkich specjalności, daje im sprzęt, wspaniałe warunki pracy, zapewnia doskonałe warunki socjalne. W zamian za to żąda jedynie rzetelnej pracy i prawa do firmowania każdego pomysłu, który powstanie w ich laboratoriach.

Gdyby podczas studiów dano nam możliwość współpracy z konkretnymi instytucjami, placówkami badawczymi, gdyby bardziej indywidualnie traktowano studenta i tok jego studiów, z pewnością okazałoby się to korzystne dla obu stron. Istnieją przecież instytucje w niewielkiej mierze wykorzystujące sprzęt komputerowy, albo takie, w których informatycy potrafiliby usprawnić pracę (np. wydawanie paszportów!), odciążyć pracowników przejmując wiele żmudnych automatycznych czynności. Nam zaś potrzebny jest dostęp do sprzętu.

Wydaje się też wielce wskazane, aby tzw. praktyki zawodowe mogły przeobrazić się w kształcący kilkumiesięczny pobyt na uczelniach lub w innych placówkach w krajach, gdzie informatyka rozwija się prężniej niż u nas, gdzie jest sprzęt i wyższa kultura techniczna, w krajach które etap oswajania społeczeństwa z informatyką mają już za sobą.

Joanna GUTT

Co dalej ze związkami zawodowymi

Najwyższy już czas, by zastanowić się nad sposobem wznowienia działalności związków zawodowych w Polsce. Zgodnie z zapewnieniami w „Propozycjach" Komitetu Rady Ministrów ds. Związków, wznowienie tej działalności ma przebiegać w pełnym poszanowaniu samorządności i niezależności związków. Ich model ma być ponadto p o n o w n i e określony przez ludzi pracy. Powstają pewne wątpliwości, jak właściwie ludzie pracy mają ten model określić.

W stanie zawieszenia działalności wszystkich organizacji i instancji związkowych, wobec nadzwyczajnych wymogów stanu wojennego — nie ma w tej chwili możliwości wyrażania kolektywnych opinii załóg pracowniczych. A sadzę, że tylko takie opinie odzwierciedlają poglądy większości. Nie oznacza to jednak, że trzeba czekać na decyzje. Pracownicy poszczególnych grup zawodowych winni się z konieczności wypowiadać i n d y w i d u a l n i e. Dotyczy to szczególnie — moim zdaniem — tych grup zawodowych, które (tak jak my — informatycy) nie stanowią większości i są rozproszone w ramach resortów, branż i zakładów. Istnieje niebezpieczeństwo, że ewentualne przemiany w ruchu związkowym znowu postawią naszą grupę zawodową poza sferą zainteresowań. Łączy nas wykonywanie podobnej pracy — tu rodzą się konflikty i problemy zawodowe i tylko my sami powinniśmy je rozwiązywać — nikt inny (nie rozumiejący naszych problemów) ich za nas nie rozwiąże. Istotną rolę powinny odegrać n a s z e związki zawodowe.

Rola związków nie może być ograniczona do a p r o b o w a n i a, p o p i e r a n i a i m o b i l i z o w a n i a, ponieważ w takim przypadku spełniałyby rolę pomocniczą, stałyby się organizacjami fikcyjnymi, pozbawionymi autorytetu — organizacjami, które nie będą społecznie akceptowane. Nie można do tego dopuścić. Autonomia związków zawodowych jest fundamentalnym warunkiem ich właściwego miejsca w państwie.

Powstaje więc zasadnicze pytanie, na czym mają polegać zadania związków zawodowych w naszym kraju. Uważam, że ich podstawowym zadaniem jest — przy zaaprobowaniu podstaw ustrojowych — szeroki udział w życiu społecznym. Nie wierzę w deklaracje określające ZZ jako organizacje apolityczne. We wszystkich szczegółowych problemach związki powinny jednak przede wszystkim reprezentować interesy swoich członków, a co za tym idzie — muszą mieć własne zdanie. Zdanie oparte na doświadczeniu grupy zawodowej, którą reprezentują.

Związki działają w sferze pracy i stosunków pracy, a tu także rodzą się działania polityczne. W wyniku doniosłych wydarzeń Sierpnia 1980 powstały nowe nurty ruchu związkowego: „Solidarność" i związki autonomiczne (do których zalicza się NSZZ Pracowników Informatyki). Nie wyobrażam sobie, aby te ugrupowania miały zniknąć z mapy związkowej w Polsce. Uważam, że silny ruch związkowy — gwarantujący kontynuację tych przemian społecznych — to ruch pluralistyczny z jasnymi motywacjami zawodowymi. Odejście od tej idei, to pozbawienie głosu w sprawach ogólnospołecznych i czysto zawodowych przedstawicieli wielu zawodów — w tym i nas, informatyków. Nie wierzę, aby nasze interesy właściwie reprezentował np. energetyk czy nauczyciel.

Pluralizm rozumiem szeroko — to znaczy nie tylko w odniesieniu do nurtów związkowych, ale i poszczególnych zakładów pracy. Sądzę, że przyjęcie zasady działania jednego związku zawodowego w zakładzie pracy stwarzałoby możliwość powrotu do struktur sprzed 1980 roku, co — zgodnie z zapewnieniami — nie jest możliwe. Pluralizm związkowy nie przekreśla jedności działania; z wielości rodzi się jedność. Tak więc — wprost przeciwnie — jest ogromna szansa na porozumienie. Przyjęcie tej zasady stworzy podstawy do rywalizacji poszczególnych związków i ich central w obronie interesów pracowniczych. Owo porozumienie stanowi rację bytu związków zawodowych i uzasadnia konieczność pilnego wznowienia ich działalności. Działalność ta winna być wznowiona jeszcze w czasie trwania stanu wojennego.

Jednym z warunków odnowy społecznej i gospodarczej kraju jest wznowienie działalności związków — na rzecz poprawy etyki i moralności pracy, podniesienia godności zawodowej i racjonalnych zasad polityki kadrowej. Zawieszony obecnie NSZZ Pracowników Informatyki zgłaszał wielokrotnie konstruktywne inicjatywy rozwiązań gospodarczych, uwzględniających interesy państwa oraz naszej grupy zawodowej (model przedsiębiorstwa informatycznego, pośrednictwa w umieszczaniu zleceń, migracje zarobkowe itp.). Dalszy brak działalności związkowej informatyków stanowi poważne zagrożenie naszego bytu.

Związki zawodowe muszą mieć własne zdanie, oparte na doświadczeniu swoich członków. Muszą mieć one odwagę i możliwość poddawania w wątpliwość stanowisk władzy — szczególnie w imię sprawiedliwego podziału dóbr, właściwego miejsca jednostki w społeczeństwie, jej integralności i prawa do społecznego i zawodowego awansu na podstawie rzeczywistych umiejętności i dokonań (nie zaś sztucznych systemów doboru kadr). Z drugiej strony związki zawodowe nie mogą sobie rościć prawa do nieomylności. W negocjacjach powinny prezentować właściwie pojęty umiar, biorąc pod uwagę interes pracowników i możliwości państwa — co nie oznacza, że winny zgadzać się na mniej niż to jest możliwe.

Każdy, komu nie jest obojętny los naszego kraju — powinien się zastanowić nad rolą Związków Zawodowych w naszym społeczeństwie. Jestem przekonany, że jedność działania i porozumienie są niezbędne, szczególnie w sytuacjach kryzysowych. Opowiadałem się i opowiadam za związkami zawodowymi zrzeszającymi grupy zawodowe. Związki zawodowe mogą i powinny odegrać istotną rolę w przezwyciężaniu kryzysu — ale muszą mieć możliwość legalnego działania.

Brak autonomii, pluralizmu, możliwości krytyki i tolerancji niszczy prestiż i atrakcyjność związków zawodowych. Nie można dopuścić do tego, aby większość pracowników nie należała do związków zawodowych.

BOGDAN FIUTOWSKI
Przewodniczący NSZZPI

Okazja!

Pozostały nam jeszcze do rozdania następujące archiwalne egzemplarze: MASZYN MATEMATYCZNYCH — nr 3/69 oraz INFORMATYKI — nr 2—7, 10—12/71; 2, 6, 10/72; 3, 9—12/73; 4—12/74; 1, 3/75; 7—8/76; 1—3, 6/78; 2, 1, 7/79. Prosimy o składanie zamówień do Redakcji. Pierwszeństwo mają biblioteki.

POLSKIE TOWARZYSTWO INFORMATYCZNE

Bieżąca działalność PTI

W okresie ostatniego półrocza działalność Polskiego Towarzystwa Informatycznego była wynikiem głównie inicjatyw Zarządu Głównego. Zarząd Główny PTI podjął w tym okresie szereg przedsięwzięć zmierzających do zaktywizowania środowiska informatyków. Za sprawę najważniejszą uznano rozwinięcie działalności sekcji i komisji PTI, które powinny przejąć większą część działań merytorycznych Towarzystwa.

Niektóre sekcje i komisje odbyły już swoje pierwsze zebrania i ustaliły plany działań na najbliższy rok. Opiekunami poszczególnych sekcji i komisji są:

Sekcja Baz Danych — Włodzimierz Mardal, tel. 21-84-41 w. 274

Sekcja EMC IBM — Lech Janczewski, tel. 28-79-42

Sekcja SM — Irena Matusiak, tel. 48-10-41 (Wrocław)

Sekcja Sieci Komputerowych — Jarosław Deminet, tel. 20-02-11 w. 2103

Sekcja Sprzętu Mikroprocesorowego — Bartłomiej Głowacki

Sekcja Grafiki Komputerowej — Zenon Kulpa, tel. 20-62-21 w. 122

Komisja Stopni Specjalizacyjnych — Jacek Bańkowski, tel. 25-28-09

Komisja Biblioteczna — Janusz Schminda, tel. 21-32-83

Komisja Wydawnictw — Maciej Stolarski, tel. 21-75-17

Komisja Szkoleniowa — Antoni Mazurkiewicz, tel. 20-02-11 w. 2548

Zarząd Główny ustalił, że członkami sekcji mogą być osoby fizyczne (członkowie PTI) oraz instytucje, będące członkami wspierającymi reprezentowane przez swoich delegatów. Składy sekcji i komisji są otwarte i wszyscy zainteresowani mogą się zgłaszać do ich opiekunów bezpośrednio lub pisemnie poprzez sekretariat Zarządu Głównego.

Zarząd Główny wystąpił z inicjatywą zorganizowania, działającego pod opieką PTI, przedsiębiorstwa (o charakterze spółdzielni) produkującego oprogramowanie i świadczącego usługi w zakresie wykorzystania informatyki. Przedsiębiorstwo pracowałoby dla kontrahentów krajowych i zagranicznych, a zatrudniałoby członków PTI. Działaniami zmierzającymi do zorganizowania takiego przedsiębiorstwa kieruje Ryszard Dąbrówka.

Polskie Towarzystwo Informatyczne dąży do jak najszerszego prezentowania spraw polskiej informatyki w organach rządowych. W wyniku odpowiednich starań przedstawiciel PTI, Jerzy Kisielnicki, bierze udział w pracach XI zespołu Komisji ds. Reformy Gospodarczej (Postęp Techniczny).

Rozwijana jest działalność szkoleniowa. W ub.r. odbyło się kilka konwersatoriów i seminariów środowiskowych w Warszawie i Wrocławiu. W bieżącym roku zorganizowane zostanie przez Andrzeja Bliklego stałe, comiesięczne seminarium poświęcone prezentacji najnowszych osiągnięć informatyki. W dalszej perspektywie będą organizowane monotematyczne szkoły letnie i zimowe.

Zarząd Główny podjął uchwałę o przyjmowaniu do PTI członków wspierających. Członkowie wspierający będą mieli prawo do używania przy nazwie jednostki organizacyjnej stwierdzenia, że jest ona członkiem PTI.

W dążeniu do podniesienia rangi rzetelnej wiedzy i umiejętności zawodowych zamierza się wprowadzić — nadawane przez PTI — stopnie specjalizacyjne w poszczególnych działach informatyki. Komisja Stopni Specjalizacyjnych kończy opracowanie regulaminu ich nadawania, który niedługo zostanie opublikowany. Z podobnych przesłanek wychodzi również inicjatywa Komisji Wydawniczej, aby promować publikacje członków PTI do druku w zagranicznych periodykach naukowych oraz by dokonywać doboru prac, które będą reprezentowały dorobek PTI na sesjach międzynarodowych konferencji informatycznych.

Mechanizm promocji prac naukowych przeznaczonych do publikacji w zagranicznych periodykach zapewniłby członkom PTI możliwość prezentacji wyników swoich prac badawczych czy wdrożeniowych na szerokim forum, a zarazem świadczyłoby o wkładzie PTI w rozwój informatyki. Mechanizm ten mógłby działać w oparciu o tych członków Towarzystwa, którzy są członkami kolegiów redakcyjnych lub w oparciu o kontakty członków PTI z członkami kolegiów redakcyjnych odpowiednich periodyków. Prace przeznaczone do promocji za pośrednictwem sekcji wydawniczej byłyby przy tym selekcjonowane i typowane wstępnie przez poszczególne sekcje tematyczne Towarzystwa w ramach organizowanych przez nie seminariów czy konwersatoriów.

Zarząd Główny PTI jest przekonany, że członkowie PTI mogą przyczynić się do rozwijania informatyki polskiej oraz rozwiązywania jej bieżących problemów zawodowych. Dlatego prosimy o nadsyłanie propozycji — bezpośrednio pod adresem Zarządu Głównego — inicjatyw i problemów, w których rozwiązaniu PTI mogłoby pomóc.

Aktywność wszystkich członków PTI w zakresie propagowania w swoim środowisku idei Towarzystwa przyczynia się do intensyfikacji rozwoju informatyki. Służy temu zwłaszcza pozyskiwanie nowych członków PTI oraz członków wspierających.

Przypominamy adres do korespondencji:
Zarząd Główny PTI
ul. Jasna 14/16, pok. 338
00-041 Warszawa

Sekretariat prowadzi w godz. 9—15 p. Anna Pykało
telefon: 26-82-61 w. 122

BOLESŁAW SZYMAŃSKI

Pierwszy egzemplarz R-60 w sieci ZETO

W dniu 2 września 1981 r. w ZETO Katowice odbyła się uroczystość oficjalnego przekazania do eksploatacji pierwszego w kraju komputera typu R-60. Ten największy z produkowanych obecnie modeli serii RIAD pozwoli w istotny sposób wzmocnić dotychczasowe wyposażenie sprzętowe, a w konsekwencji znacznie zwiększyć potencjał obliczeniowy ośrodka katowickiego. Doniosłość faktu podkreślał udział w uroczystości nie tylko przedstawicieli kierownictwa resortu MNSzWiT, władz wojewódzkich i głównych użytkowników ZETO Katowice, ale również reprezentantów kierownictwa władz nadrzędnych producenta R-60, polskich i radzieckich branżowych central handlu zagranicznego oraz wydziału handlowego ambasady ZSRR w Polsce.

W.K.

Diagnostyka raz jeszcze

Prof. E. J. McCluskey w akcji

W drugim zeszłorocznym numerze INFORMATYKI poinformowaliśmy czytelników o tematyce i terminie konferencji FTS&D 81. Konferencja ta jest już za nami. Odbyła się w dniach 28—30 września ub.r. w Brnie w Czechosłowacji w pięknym hotelu „Voroneż". Była to już czwarta konferencja z serii FTS&D organizowanych na zmianę — w Polsce i w Czechosławcji.

Tym razem organizatorami były następujące instytucje z Czechosłowacji: Stowarzyszenie Techniczne i Naukowe w CSSR (ČSVTS), Czechosłowacki Komitet ds. Elektrotechniki przy ČSVTS, Centralna Zawodowa Grupa ds. Diagnostyki w Elektronice przy ČSVTS, Czechosłowacki Narodowy Komitet IMEKO, Podkomitet ds. Diagnostyki Technicznej.

Komitet Programowy Konferencji pracował natomiast w zespole reprezentowanym przez pracowników: Instytutu Maszyn Matematycznych w Pradze, Politechniki w Brnie, Słowackiej Akademii Nauk w Bratysławie, Instytutu Technik Obliczeniowych w Żilinie, Politechniki Śląskiej w Gliwicach, Politechniki Warszawskiej oraz Uniwersytetu Śląskiego.

Do organizatorów Konferencji nadesłano 67 prac poświęconych zagadnieniom testowania i niezawodności cyfrowych systemów na etapie ich projektowania, produkowania i stosowania. Ostatecznie Komitet Programowy zakwalifikował i włączył do materiałów Konferencji 46 referatów, opracowanych przez specjalistów z USA, Japonii, RFN, Francji, Włoch, Finlandii, Jugosławii, ZSRR, Węgier, Bułgarii, NRD, Rumunii, Czechosłowacji i Polski.

Referaty wygłoszono w sesjach poświęconych następującej problematyce:

- projektowanie systemów tolerujących uszkodzenia
- modelowanie i ocena systemów o wysokiej niezawodności
- niezawodność oprogramowania
- generacja testów
- symulacja testów
- projektowanie w celu uzyskania łatwej testowalności
- systemy samotestujące
- diagnostyka systemów i urządzeń.

Gospodarze Konferencji tym razem nie zorganizowali dyskusji okrągłego stołu. Jej brak zrekompensował zespół pięciu tzw. „invited papers", z których najciekawszym — wartym podkreślenia — był referat prof. E. J. McCluskey (z Uniwersytetu w Stanfordzie w USA) związany tematycznie z projektowaniem układów i systemów łatwych do testowania. Zagadnienie to było jednym z najczęściej poruszanych. To zainteresowanie wynika głównie z faktu, że obecne złożone elementy scalone (tzw. chipy LSI i VLSI) wymagają coraz większego nakładu środków na ich testowanie.

Zagadnieniem, które również zostało mocno zaakcentowane była analiza sygnatur. Po raz pierwszy także na konferencji z cyklu FTS&D zagadnienia niezawodności oprogramowania zostały poruszone aż w czterech referatach. W związku z tym poświęcono im jedną sesję.

Na podstawie Konferencji można stwierdzić, że zainteresowania naukowców w dalszym ciągu coraz bardziej przesuwają się z zagadnień czysto teoretycznych, oderwanych od życia, do zagadnień związanych z potrzebami przemysłu. Jest to dobra prognoza dla najbliższej konferencji FTS&D'82, o której pierwszy komunikat zamieszczamy obok.

ANDRZEJ HŁAWICZKA

Uczestnicy konferencji, w pierwszym rzędzie specjaliści z Polski

FTS & D'82

Kolejna międzynarodowa konferencja FTS&D'82 (Fault — Tolerant Systems and Diagnostics) odbędzie się w Katowicach w dniach 21—23 września 1982 r.

Tematyka konferencji:

- mechanizm uszkodzeń i modelowanie
- generowanie testów: metody i programy
- urządzenia i narzędzia diagnostyczne
- wyposażenie do automatycznego testowania
- diagnostyka mikroprocesorów i mikrokomputerów
- diagnostyka systemowa, diagnostyka systemów wieloprocesorowych oraz sieci komputerowych
- projektowanie łatwych do testowania układów i systemów
- układy samokontrolujące, samotestujące i odporne na uszkodzenia
- kody korygujące błędy i ich implementacja
- projektowanie, ocena i programowanie systemów tolerujących uszkodzenia
- systemy o dużej niezawodności dla celów sterowania w czasie rzeczywistym
- aspekty ekonomiczne diagnostyki oraz tolerowania uszkodzeń
- niezawodność oprogramowania.

Szczegółowy program konferencji zostanie przesłany wszystkim zgłoszonym uczestnikom w czerwcu 1982 r. Oficjalnymi językami konferencji będą angielski, polski i rosyjski.

Udział w konferencji, referaty oraz zapytania należy kierować pod adresem Komitetu Organizacyjnego FTSD'82:

Dr Marian Budka
NOT, FTSD'82 Conference
ul.: Podgórna 4, skr. poczt. 468
40-955 Katowice, Poland
tel. 514-558

Instytut Automatyki Systemów Energetycznych

jest zainteresowany zakupem następujących urządzeń:

- pamięci dyskowe EC 5052 — 4 szt.
- jednostkę sterującą do pamięci taśmowych EC 5012 typu MTS 304/2 — 1 szt.
- urządzenia teletransmisji UTD-211
- dalekopisy T-100
- drukarki znakowo-mozaikowe DZM-180
- czytnik taśmy papierowej CT 2101 (logika ujemna)
- perforator taśmy DT 105s (logika ujemna)

Oferty prosimy kierować pod adresem:
ul. Wystawowa 1, 51-618 Wrocław,
Telefon: 48-42-21 w. 238

EO/814/K/81

OŚRODEK OBLICZENIOWY PRZY BIURZE PROJEKTÓW I REALIZACJI INWESTYCJI PRZEMYSŁU SYNTEZY CHEMICZNEJ „PROSYNCHEM"

PRZYJMIE ZLECENIA NA NASTĘPUJĄCE USŁUGI:

- wykonywanie obliczeń programami będącymi w dyspozycji ośrodka, a dotyczącymi zagadnień konstrukcyjnych, technologii chemicznej, budowlanej, ochrony środowiska, obliczeń rurociągowych, automatyki przemysłowej, elektro-energetycznych, przetwarzania danych,
- udostępnianie informacji z banków danych własności wytrzymałościowych stali konstrukcyjnych, własności fizykochemicznych, mediów technologicznych,
- programowanie problemów klienta w zakresie obliczeń technicznych, finansowych i przetwarzania danych,
- wykonywanie obliczeń programami klienta (sprzedaż czasu EMC),
- projektowanie i wdrażanie systemów informatycznych,
- pomoc we wdrażaniu systemów operacyjnych,
- szkolenie w zakresie programowania,
- szkolenie w zakresie wykorzystywania systemu GEORGE-3,
- inne usługi informatyczne i projektowe.

Ośrodek dysponuje:

- procesorem ODRA 1305 o pamięci 256 k słów,
- pamięciami taśmowymi i dyskowymi,
- czytnikami kart i taśmy papierowej,
- siecią teletransmisji zdalnej łączącej biura i instytuty z terenu Śląska.

Instalacja pracuje pod kontrolą systemu operacyjnego GEORGE-3. Istnieje możliwość podłączenia zdalnych końcówek klientom.

INFORMACJI W POWYŻSZYCH SPRAWACH UDZIELA I ZLECENIA PRZYJMUJE: BIURO PROJEKTÓW „PROSYNCHEM", PRACOWNIA INFORMATYKI
ul. Konstytucji 11, 44-101 Gliwice
Telefon: 31-37-14, 31-17-41

EO/866/K/81

Komputery w nauczaniu

W dniach 27—31 lipca 1981 r. odbyła się w Lozannie (Szwajcaria) trzecia Międzynarodowa Konferencja nt. Zastosowań Informatyki w Nauczaniu (3rd World Conference on Computers in Education), organizowana przez Komitet TC3 Międzynarodowej Federacji Przetwarzania Informacji (IFIP).

Tematyka konferencji obejmowała przede wszystkim problemy komputeryzacji dydaktyki w szkołach wyższych i średnich oraz związane z tym zagadnienia planowania, programowania i zarządzania komputeryzacją szkolnictwa. Obrady konferencji były prowadzone w następujących sesjach:

- Informacja i informatyka a inne dyscypliny nauki (nauki społeczne, językoznawstwo, nauki przyrodnicze, matematyczne, projektowanie techniczne, projektowanie w sztuce, muzyka, nauki medyczne).
- Nauczanie wspomagane komputerem oraz inne sposoby bezpośredniego wykorzystania komputerów w nauczaniu (systemy nauczania dla szkolnictwa wyższego i średniego).
- Zastosowanie nowych technologii komputerowych (rozwój technik audiowizualnych, pomocy dydaktycznych, systemów mikroprocesorowych).
- Aspekty społeczne wprowadzania informatyki, w tym zmiana roli nauczyciela.
- Strategie i modele komputeryzacji nauczania w poszczególnych krajach, ze szczególnym uwzględnieniem potrzeb krajów rozwijających się (polityka komputeryzacji w szkolnictwie średnim i wyższym).
- Cele, polityka i programy nauczania informatyki (nauczanie informatyki w szkolnictwie średnim, szkolenie nauczycieli, zaawansowane techniki nauczania, matematyka w programie nauczania informatyki, doświadczenie).

W konferencji wzięło udział ponad 1250 przedstawicieli z 67 krajów. Zgłoszono ogółem ponad 900 referatów, spośród których przyjęto do prezentacji i publikacji 114 referatów (odpowiednio w sesjach: 34, 14, 13, 17, 15 i 21).

Niezależnie od sesji zorganizowano dyskusje panelowe, w których omawiane były najnowsze pomysły, rozwiązania i osiągnięcia (może niezbyt precyzyjnie sformułowane, ale wskazujące na istniejące trendy). Były one pogrupowane w pięciu zespołach tematycznych:

- systemy nauczania wspomaganego komputerem (CAL) — osiągnięcia przy stosowaniu systemów LOGO, PLATO, projektowanie CAL, przygotowywanie nauczycieli do CAL
- najnowsze techniki stosowane w CAL — zastosowanie teleinformatyki, videodysków i mikrokomputerów
- zastosowanie CAL w odległych od informatyki dziedzinach (nauczanie języka, nauczanie wymowy, nauczanie użytkownika nieinformatyka, nauczanie zastosowań przemysłowych, skutki społeczne)
- CAL w krajach rozwijających się — przegląd nauczania informatyki, współpraca międzynarodowa
- nauczanie informatyki.

Ze strony polskiej zaprezentowano dwa referaty:

— „Program nauczania przedmiotu „Inżynieria komputerowa" na Politechnice Warszawskiej" (Jan Zabrodzki)
— „Metodologia i doświadczenia w dziedzinie komputeryzacji szkół wyższych w Polsce" (Mieczysław Bazewicz i Leopold Misiaszek).

Pierwszy z nich omawiał program nauczania projektantów sprzętu oraz oprogramowania systemów komputerowych — z uwzględnieniem metod łączenia aspektów teoretycznych i praktycznych programu nauczania, a także sposobów prowadzenia programów indywidualnych. Drugi natomiast — przedstawiony w grupie referatów poświęconych polityce komputeryzacji szkolnictwa wyższego (obok dwóch referatów USA) — prezentował założenia, metody realizacji i wyniki merytoryczne w Problemie Resortowym MNSzWiT RI-14 pn. „Rozwój komputeryzacji szkół wyższych" oraz dalsze prognozy i plany rozwoju zastosowań informatyki w latach osiemdziesiątych. Problemy te były także poruszone w dyskusji panelowej na temat nauczania informatyki, zwłaszcza studentów w specjalnościach technicznych.

W czasie trwania konferencji zorganizowana była wystawa specjalistyczna sprzętu komputerowego przeznaczonego do wspomagania nauczania, prezentująca m.in. terminale uniwersalne i specjalne, autonomiczne systemy mikroprogramowane, systemy przystosowane do obsługi użytkowników niepełnosprawnych, systemy wspomagające opracowywanie zadań, urządzenia ułatwiające dostęp do baz danych dla celów nauczania i badań, systemy do indywidualnej obsługi użytkowników niepełnosprawnych, systemy wspomagające opracowywanie zadań, urządzenia ułatwiające dostęp do baz danych dla celów nauczania i badań, systemy do indywidualnej obsługi użytkowników. Wśród wystawców należy wymienić m.in. następujące firmy: ACORN, COMPUTER, APPLE COMPUTER, ATARI COMPUTER DIVISION, COMMODORE COMPUTER, CONTROL DATA, COSENDAI COMPUTER PRODUCTS, NORSK DATA, RC COMPUTER, SYMAG INFORMATIQUE, TEXAS INSTRUMENTS.

W programie konferencji wyraźnie dominowała problematyka zastosowań informatyki w dydaktyce, a nie zagadnienia rozwoju samych środków informatyki. Oznacza to konieczność zmiany dotychczasowego sposobu podejścia użytkowników, którzy zastosowania powinni podporządkować wyłącznie wymaganiom rozwiązywanego problemu, a nie wymaganiom i ograniczeniom komputera.

MIECZYSŁAW BAZEWICZ

W ślad za Andrzejem Wajdą ruszyłem we wrześniu do Cannes. W Pałacu Festiwali nie pokazywano już jednak filmów, lecz slajdy z wykresami. Od 9 do 11 września odbywała się tam siódma międzynarodowa konferencja, poświęcona Bardzo Wielkim Bazom Danych (Very Large Data Bases), na którą zjechało niemal 500 „bazodanowców" z całego świata [1]). Wśród nich sławy: Chris Date — autor najbardziej znanego (trzy wydania angielskie, polskie właśnie się ukazało) podręcznika baz danych, Moshe Zloof — twórca jedynego dostępnego na rynku systemu relacyjnego „Query-By-Example", Jim Gray — jeden z architektów Systemu R, David Hsiao, Dennis Tsichritzis, John i Diana Smith. Tylko niektórzy z nich wygłosili referaty, pozostali ograniczyli się do wystąpień w dyskusjach i rozmów w kuluarach. Wskutek tego dominowały, niestety, referaty przyczynkarskie, a i dyskusja — mimo prób ożywienia jej, m.in. przez dr. Stniszkisa — była raczej niemrawa.

Spośród 21 tematów sesji trzeba wyróżnić następujące:

TEORIA BAZ DANYCH

Eksplozja odkryć rozmaitych typów zależności w relacyjnych bazach danych została wreszcie uwieńczona pierwszymi syntezami: pracami Beeriego, Maiera czy Sagiva o logicznych podstawach teorii zależności. W Cannes Yannakakis i Chase przedstawili koncepcję a c y k l i c z n y c h baz danych. Mają one dość prostą postać, wystarczającą jednak do odzwierciedlenia większości występujących w praktyce sytuacji. W acyklicznych bazach danych istnieją wyłącznie zależności funkcyjne oraz jedna zależność złączeniowa (ang. *join dependency*), odpowiadająca naturalnemu rozkładowi schematu na sensowne obiekty. Wiele trudnych (ang. *intractable*) problemów daje się wówczas rozwiązać przez algorytmy o złożoności wielomianowej. W dalszym ciągu wielu autorów zajmuje się „niczym", czyli w a r t o ś c i a m i n i e i s t n i e j ą c y m i (ang. *null values*). Referat Imielińskiego i Lipskiego, formułujący warunki sensowności dla operacji alge-

[1]) W Konferencji wzięło udział sześciu uczestników z Polski, oprócz mnie: dr Mieczysław Muraszkiewicz z Politechniki Warszawskiej (referat), dr Tomasz Imieliński i dr Witold Lipski z Instytutu Podstaw Informatyki PAN (referat) oraz mgr inż. Jan Popiel (Centrum Projektowania i Zastosowań Informatyki, obecnie University of Maryland) i dr Witold Staniszkis (również CPiZI, obecnie Uniwersytet Kalabryjski — Włochy). Głównym organizatorem był francuski Institut National de la Recherche en Informatique et en Automatique (INRIA), a współorganizowały: ACM i IEEE Computer Society, natomiast pomocy finansowej udzieliło organizatorom 11 instytucji z Belgii, Francji, Anglii, RFN, Włoch i USA. W przyszłym roku ósma Konferencja VLDB będzie organizowana w Meksyku pod auspicjami IFIP, z tym że instytucje amerykańskie zgłosiły już wycofanie się z uczestnictwa w tej imprezie.

Bardzo Wielkie Bazy Danych'81

bry relacji w przypadku wartości nieistniejących, został dobrze przyjęty. Również i inni autorzy podejmowali problem semantycznych rozszerzeń podejścia relacyjnego. Są już nawet implementacje; niewątpliwie najciekawszy — to tworzony w laboratoriach Bella system DSIS. Zapytania są w nim realizowane przez sieć wyspecjalizowanych procesorów strumieni, pełniących funkcje filtrów, czytników, kalkulatorów, sorterów itp. Dzięki temu możliwy jest wysoki stopień równoległości.

MODELE BAZ DANYCH

Najbardziej kontrowersyjnym tematem jest tu niewątpliwie reprezentacja dynamiki w bazach danych. Dotychczas sytuacja wyglądała tak: na pasywną bazę danych napuszczało się aktywne programy — bądź użytkowe, bądź zarządzające. Chodzi o to, by baza „ożyła" i sama generowała zdarzenia i akcje. Osiągnięto na razie ciekawe wyniki w dziedzinie specyfikacji dynamiki, np. stosując formalizm sieci PETRIEGO. W jaki sposób jednak w samej bazie reprezentować dynamikę — nie wiadomo. Kłopoty z dynamiką mają głębsze korzenie, jako że *logika formalna dotychczas pomijała zdania rozkazujące, rozwijając bogatą teorię zdań twierdzących* (Melkanoff). Z doświadczeń W. Kenta wynika natomiast, że na podstawie modelu pojęciowego przedsiębiorstwa nie sposób analitycznie zaprojektować bazy danych. W każdym kolejnym kroku analizy występuje niedeterminizm, który może zostać zredukowany tylko poprzez uwzględnienie kontekstu użytkowania bazy. Byłby więc to jeszcze jeden argument za integracją danych i operacji.

SYSTEMY ZARZĄDZANIA BAZĄ DANYCH

Rozwijają się dwie mutacje teorii współbieżności: pesymistyczna i optymistyczna. Pierwsza, zakłada znaczną częstotliwość konfliktów między transakcjami i konstruuje rozmaite protokoły zamykania (Fussell, Kedem i Silberschatz). Druga, puszcza transakcje „na żywioł" i tylko wówczas, gdy po zakończeniu stwierdzony zostanie konflikt, transakcja jest cofana (Schlageter). Ponadto Beeri i Obermarck przedstawili w Cannes najogólniejszy znany dotąd algorytm wykrywania zakleszczenia.

Do najciekawszych należały referaty Gray'a i Borr o systemach ciągłej obsługi w komputerach TANDEM. Bardzo długi, mierzony w latach średni czas między awariami osiągnięto tam dzięki konsekwentnemu stosowaniu zasady dublowania na wszystkich poziomach architektury systemu. Każdy proces ma swój duplikat. Odtwarzanie pozostaje dla użytkownika niewidoczne. Z kolei Söderlund analizując eksperyment symulacyjny pokazał, że opłaca się stosować reorganizację fizycznej bazy danych współbieżnie z przetwarzaniem bieżących zapytań.

W dziedzinie ochrony baz danych dzieje się chyba niewiele. N. Minsky w Cannes opisywał sytuacje, gdy możliwości, jakie daje suma uprawnień są większe niż suma możliwości, jakie dają osobno te uprawnienia. Żeby móc — na przykład — prowadzić samochód, trzeba posiadać prawo jazdy oraz być właścicielem samochodu (uzyskać zgodę właściciela). Wniosek: pojęcie „uprawnienia" w komputerowym świecie funkcjonuje w zawężonym sensie i nie odzwierciedla wielu ważnych faktów prawnych. Minsky zaproponował rozwiązanie tego problemu poprzez uogólnienie pojęcia „dojścia" (ang. *capability*).

Stosunkowo mało słychać było w Cannes o rozproszonych bazach danych. Dzieje się tak chyba dlatego, że większość problemów teoretycznych w tej dziedzinie została już rozwiązana, a na ocenę użytkową rozmaitych systemów trzeba jeszcze poczekać. Natomiast sporo miejsca poświęcono ocenie maszyn baz danych. De Witt sugerował (na przyszłość) odwrócenie podejścia: najpierw zaprogramować równoległe algorytmy dla operacji relacyjnych, wyodrębnić w nich operacje pierwotne i dopiero wówczas zbudować maszynę, która wykonywałaby możliwie efektywnie te operacje.

W dalszym ciągu rozpatruje się rozmaite strategie optymalizacji zapytań relacyjnych. Dość obiecująco wygląda metoda semantyczna (J. J. King). Przykładem heurystyki stosowanej w tym podejściu jest np. rozszerzenie kwerendy, pozwalające zastąpić przeglądanie całej relacji wyszukiwaniem poprzez indeks. Jako podstawa dla takiego rozszerzenia służy zbiór ogólnych faktów (reguł) z dziedziny, której dotyczy dana baza. Natomiast S. Cammerata zaproponowała odkładanie modyfikacji aż do momentu, gdy zmieniona wartość będzie potrzebna. Menon i Hsiao pokazali optymalną realizację układową relacyjnej operacji złączenia.

Zainteresowanych pozostałymi referatami odsyłam do materiałów z Konferencji, zredagowanych przez C. Zaniolo i C. Delobela. Pora na wnioski.

• Rozwój zastosowań minikomputerów i sieci komputerowych stawia problem integracji systemów baz danych z systemami przetwarzania tekstów typu biurowego i systemami transmisji danych. Pierwsze projekty są już w toku: na Uniwersytecie w Toronto i oczywiście w IBM.

• W dziedzinie baz danych obecnie występują już wszystkie problemy, wyróżniające dotychczas dziedzinę systemów operacyjnych, takie jak synchronizacja, ochrona czy ocena efektywności. Co więcej — wymagają one nowych, specyficznych rozwiązań.

• Wielce owocne okazać się może wzajemne oddziaływanie baz danych i języków programowania. Do opisu modeli danych — na przykład — wykorzystuje się znane formalizmy, jak: logika Hoare'a, logika dynamiczna, abstrakcyjne typy danych. W programowaniu natomiast niewątpliwie przydatne byłyby używane w dziedzinie baz danych pojęcia transakcji, perspektywy (ang. view) czy rozmaite rodzaje abstrakcji.

• To samo można powiedzieć o wzajemnym wpływie baz danych i sztucznej inteligencji. Semantyczne modele danych wzięły się przecież z tego oddziaływania. Natomiast rozwiązania z dziedziny baz danych dałyby się wykorzystać do zwiększenia efektywności rozmaitych systemów inteligentnych.

• Przeszkodą w integracji baz danych z innymi dziedzinami jest specyficzna, osobno się dotychczas rozwijająca terminologia techniczna. Zaczyna ona jednak zanikać — nie wiadomo dokładnie, czym różni się schemat od bazy danych, baza danych od systemu zarządzania nią ... Jednak pełna integracja możliwa będzie dopiero wówczas, gdy powstaną nowe syntezy teoretyczne.

• Można sądzić, że w przyszłości osobna dziedzina baz danych zniknie, wchłaniając inne fragmenty informatyki.

Czy można mówić o polskim wkładzie w teorię baz danych? Pojedyncze osiągnięcia naukowe nie łączą się niestety w tej dziedzinie w Polsce w żadną sensowną całość, czego dowodem może być choćby brak projektów badawczych. Nic dziwnego, gdyż rozwój każdej informatycznej teorii jest stymulowany zastosowaniami. W naszym kraju większość problemów przetwarzania danych może być rozwiązana za pomocą technologii „COBOL + taśmy magnetyczne". W praktyce potrzeba operatywnych systemów bankowych, bibliotecznych, biurowych itp.

Sądzę, że wychodząc od naszych własnych doświadczeń możemy jednak dojść do ciekawych wyników teoretycznych. Weźmy choćby problem „podwójnej sprawozdawczości": jednej dla wykazania wyników, drugiej do codziennego użytku. Założę się, że żadnemu amerykańskiemu informatykowi nie przyjdzie do głowy, że dwie wzajemnie sprzeczne perspektywy tej samej bazy danych mogą być jednocześnie potrzebne. Albo — jakże ciekawe może być zaprogramowanie odtwarzania prawdziwych danych na podstawie zdeformowanych statystyk. Odnowa (nie tylko) w tej dziedzinie mogłaby nas cofnąć z własnej, choćby i karykaturalnej drogi i wprowadzić na tor światowy w miejscu, gdzie Amerykanie byli 15 lat temu.

JAN CHOMICKI

Terminologia języka Ada

W podręczniku języka Ada (a także w jego definicji) użyto wielu nowych określeń, które nie są zbyt często używane w języku angielskim i które nie mają odpowiedników w języku polskim. Ponieważ do podręcznika dołączono słownik najczęściej używanych pojęć (nie będący częścią normy), wydaje się, że warto przedstawić czytelnikom, jak one są rozumiane przez twórców języka, a także — w niektórych przypadkach — zaproponować odpowiedniki polskie.

Jednym z podstawowych pojęć języka Ada jest pojęcie typu. Zacytujmy określenie podane w podręczniku: *Typ charakteryzuje zbiór wartości i zbiór operacji mających do nich zastosowanie.* Mówiąc o typie nie można zatem mieć na myśli — przykładowo — tylko liczb, lecz także określone na nich operacje. Pojęcie typu jest szersze od pojęć wartości i zmiennej, ponieważ obejmuje zbiór wartości (określonych w definicji typu) i zbiór zmiennych określonych w deklaracjach mogących przybierać te wartości.

W literaturze polskojęzycznej nie spotkałem dotąd precyzyjnego określenia pojęcia typu. Według J. Bieleckiego i M. Suchenka (Fortran dla zaawansowanych, PWN, Warszawa, 1981) — na przykład — typem zmiennej nazywa się funkcję przyporządkowującą wartość konkretnej informacji (inaczej: sposób interpretowania ciągu bitów wchodzących w skład zmiennej).

Typy liczbowe są jednymi z możliwych typów w języku Ada. Ogólnie — istnieją typy skalarne (ang. *scalar*), złożone (ang. *composite*) i typ dostępowy (ang. *access* type). Typy skalarne dzielą się na rzeczywiste (stało- i zmiennoprzecinkowy, ang. *fixed* i *floating point*) i dyskretne (całkowitoliczbowy — ang. *integer*, i wyliczeniowy — ang. *enumeration*). Natomiast typy złożone dzielą się na typ tablicowy (ang. *array type*) i rekordowy (ang. *record type*).

Przejdźmy do określeń podanych w podręczniku Ady. Typ skalarny jest typem, którego wartości nie mają składowych. Typ dyskretny jest typem skalarnym, który ma uporządkowany zbiór wartości dyskretnych. Typ wyliczeniowy jest typem dyrektywnym, którego wartości są przedstawione jawnie w definicji typu. Typ złożony jest typem, którego obiekty (tzn. stałe lub zmienne) zawierają składowe. Typ tablicowy jest typem złożonym, którego wszystkie składowe są tego samego typu i podtypu. Typ rekordowy jest typem złożonym, którego składowe mogą być różnych typów. Typ dostępowy jest typem, którego obiekty są utworzone przez alokację (dosłownie: wykonanie alokatora, ang. *execution of an allocator*). Alokator tworzy nowy obiekt typu dostępowego i zwraca tzw. wartość dostępu oznaczającą utworzony obiekt. Zbiór obiektów (wartości) typu dostępowego (zwany kolekcją, ang. *collection*) jest więc tworzony dynamicznie.

Obecnie uzasadnimy stosowany sposób nazywania typów. Otóż w języku angielskim nazwy typów mają następującą budowę: określenie + wyraz *type*, np. *record type*, *access type*. Wiadomo, że w języku polskim konstrukcję tę można tłumaczyć dwojako, tzn. przy użyciu rzeczownika (przydawka dopełniaczowa), np. typ rekordu, typ dostępu (jak to najczęściej robiono dotąd) lub — przy użyciu przymiotnika (przydawka przymiotnikowa), np. typ wyliczeniowy, typ tablicowy.

Wydaje się, że w informatyce można ściśle zdefiniować pojęcie typu — tak, aby było jasne, że różni się ono od pojęcia typu w języku potocznym. W związku z tym — w celu uniknięcia nieporozumień co do znaczenia wyrazu typ w przypadku jego użycia (czyli — w którym z dwóch znaczeń jest użyty) — proponuję przyjąć zasadę, że w języku polskim przydawka przymiotnikowa + wyraz *type* dotyczy typu jako konstrukcji języka programowania, natomiast — przydawka dopełniaczowa + wyraz *typ* dotyczy typu w języku potocznym.

Wtedy wyrażenie „zmienna typu tablicy" należy rozumieć potocznie, a wyrażenie „zmienna typu tablicowego" — jako odnoszące się do definicji określonego typu (w tym przypadku — typu tablicowego) w języku programowania. Jest to bardzo ważne rozróżnienie, albowiem wyrażenie „zmienna typu tablicowego" zmniejsza prawdopodobieństwo skojarzenia wyrazu typ w informatyce z wyrazem typ w języku potocznym i przywodzi na myśl, że typ (w informatyce) to nie tylko zbiór wartości, lecz także — zgodnie z definicją — zbiór związanych z nimi operacji.

Wartość typu złożonego, a dokładniej — pisemna postać oznaczająca tę wartość nazywa się agregatem. Agregat tablicowy oznacza wartość typu tablicowego, natomiast agregat rekordowy — wartość typu rekordowego. Składowe agregatu można przedstawić w zapisie pozycyjnym (ang. *positional*) lub bezpośrednim (ang. *named*).

Podtyp (ang. *subtype*) otrzymuje się przez ograniczenie zbioru dopuszczalnych wartości typu, przy czym operacje określone w tym zbiorze nie ulegają zmianie. Do ograniczeń (ang. *constraint*) należą: ograniczenie zakresu, dokładności, indeksu i ograniczenie dyskryminacyjne (dotyczące rekordów). Natomiast typ pochodny (ang. *derived type*) jest typem, którego zbiory wartości i operacji są podzbiorami innego typu zdefiniowanego.

Typem prywatnym (ang. *private type*) jest typ, którego zbiór wartości jest określony, lecz znany użytkownikowi jedynie przez tzw. dyskryminanty.

Bardzo ważnym pojęciem jest tzw. przeciążanie nazw (ang. *overloading*). Według określenia z podręcznika Ady przeciążalność jest właściwością literałów, identyfikatorów i operatorów, które mogą mieć kilka różnych znaczeń w tym samym zakresie widoczności. Zakres widoczności nazwy (zakres deklaracji, ang. *scope*) jest obszarem tekstu programu, w którym określona deklaracja wywiera skutek. Natomiast literał (wartość literalna) oznacza wartość określonego typu daną jawnie.

Mówiąc ogólnie — konstrukcja językowa nazywa się przeciążalną, jeżeli istnieje kilka jej realizacji w języku i w przypadku użycia wybiera się realizację zgodną z kontekstem zawartym w programie. Proces wyboru realizacji nazywa się rozwiązaniem przeciążenia, ang. *resolution of the overloading* (P. J. L. Wallis, B. W. Silverman, Efficient Implementation of the Ada Overloading Rules, Information Processing Letters, Vol. 10, No. 3, p. 120, 1980).

Przykładowo — literał wyliczeniowy jest przeciężony, jeżeli występuje co najmniej w dwóch różnych typach wyliczeniowych. Podprogram jest przeciążony, jeżeli jego nazwa oznacza dwa lub więcej różnych tekstów podprogramów.

Dalszą część terminologii języka Ada przedstawimy w jednym z kolejnych numerów INFORMATYKI.

JANUSZ ZALEWSKI

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

Jestem przekonany, że zastosowanie informatyki dla celów informacyjnych, tj. gromadzenia informacji, jej przetwarzania i przekazywania w odpowiednio krótkim czasie stanie się w Polsce nie tylko techniczną koniecznością, ale znajdzie również ekonomiczne uzasadnienie. Nastąpi to w wyniku szybkiego postępu techniki informatycznej na świecie, a także ogólnego — dzięki reformie gospodarczej — ożywienia naszej gospodarki.

O błędach popełnionych w informatyce mówiono już na wielu konferencjach i pisano także w prasie fachowej. Dlatego postaram się wymienić tylko te — według mnie — najważniejsze, które w sposób katastrofalny zaciążyły na rozwoju informatyki. Do nich zaliczam:

- Centralistyczny system sterowania informatyką.
- Centralny rozdzielnik sprzętu informatycznego i resortowy rozdzielnik środków na zakup tego sprzętu — często całkowicie rozbieżne.
- Bardzo wysoką amortyzację sprzętu informatycznego, wynikającą z nieproporcjonalnie wysokich cen sprzętu w stosunku do kosztów jego produkcji i cen światowych podobnego sprzętu (nt. o daleko wyższej jakości).
- Monopolistyczną pozycję producenta sprzętu i wynikające stąd konsekwencje: wysokie ceny sprzętu i usług, nieterminowość dostaw, długi czas oczekiwania na zamówiony sprzęt i opóźnienia w jego instalowaniu, niewywiązywanie się z obowiązków serwisu, sprzedaż tzw. kadłubowych konfiguracji i permanentny brak części zamiennych, bardzo niska jakość sprzętu i związana z tym — duża zawodność, nie reagowanie na uwagi użytkownika, a nawet wprost lekceważący do niego stosunek.
- „Dzikie", nie poddane analizie ekonomicznej zakupy sprzętu informatycznego i oprogramowania z II obszaru płatniczego, zgodnie z partykularnymi, resortowymi interesami. Część tego sprzętu jest do tej pory niewykorzystana lub wykorzystywana niewłaściwie, a bywają też przypadki, że sprzęt w ogóle nie został zainstalowany (np. IRIS 80 — koszt ok. 7 mln $).
- Tworzenie różnych centralnych instytucji kierujących informatyką, a także różnego rodzaju instytutów i ośrodków (głównie z myślą o konkretnych prominentach, często całkowicie niekompetentnych w branży informatycznej). Instytucje te i ludzie je prowadzący potrafili sprytnie zdezorganizować naszą informatykę, wytępić elementy twórcze, obstawić się miernotami i zlikwidować w zalążku każdy przejaw słusznej krytyki.
- Niewłaściwą politykę zakupów licencyjnych, która — z jednej strony — doprowadziła do zahamowania własnej myśli konstruktorskiej, zaś z drugiej — spowodowała znaczne straty gospodarcze.
- Egalitarne, pozbawione ekonomicznego uzasadnienia zastosowania informatyki.
- Błędną politykę zatrudnieniową i niewłaściwe planowanie środków.
- Partykularyzm grupowy i resortowy.

Błędy te należy jak najszybciej usunąć. Trzeba rozwiązać wszelkiego rodzaju sztuczne twory w postaci KI, SKI,

Całkowita decentralizacja

wszelkie zjednoczenia, centra itd. Należy zlikwidować centralny rozdzielnik na sprzęt informatyczny i obniżyć jego amortyzację, a nawet wprowadzić formę dzierżawy sprzętu. Należy złamać monopolityczną pozycję producenta i stworzyć możliwość egzekwowania swoich uprawnień przez użytkowników. I wreszcie — sprzyjać powstawaniu konkurencji na rynku informatycznym. Konieczne jest stworzenie dla wszystkich użytkowników i producentów sprzętu informatycznego jednakowych warunków rozwoju, respektujących twarde prawa ekonomiczne i prawo konkurencji, a także — samorządność, samodzielność i samofinansowanie.

Najwyższy już czas, by wyeliminować z informatyki wszelkie nieprodukcyjne ogniwa pośrednie. Zastosowania i rozwój informatyki powinny wynikać z konkretnych potrzeb i uwarunkowań ekonomicznych. Znieść trzeba wszelkiego rodzaju zbiurokratyzowaną koordynację. Wszystkie większe przedsięwzięcia informatyczne o charakterze międzygrupowym lub międzyresortowym powinny być podejmowane przez te grupy lub resorty na zasadzie dobrowolności, włącznie z podziałem zadań i środków. Informatyka musi być traktowana na równi z innymi dziedzinami techniki i nauki, takimi jak: telekomunikacja, transport, energetyka itd. Należy zrezygnować z jakichkolwiek przywilejów dla informatyki, gdyż mogą jej one tylko zaszkodzić. Informatyka jest i będzie potrzebna gospodarce, jest zdolna nie tylko przetrwać, ale też rozwijać się, zgodnie z ogólnymi prawami rozwoju — niezależnie czy komuś się to podoba, czy nie.

Reforma gospodarcza musi zmienić dzisiejszą kryzysową sytuację. W informatyce przy tym należy:

- skończyć z karuzelą stanowisk ludzi uzurpujących sobie prawo reprezentowania informatyki i informatyków w sposób samozwańczy (w różnych instytucjach, organizacjach, radach, stowarzyszeniach naukowych i zawodowych).
- sprzyjać wdrażaniu informatyki tylko w tych dziedzinach, które stwarzają możliwość efektywnego jej zastosowania
- domagać się rozliczenia ludzi za błędy i nadużycia w informatyce
- wyegzekwować od przemysłu produkcję części niezbędnych do uzupełnienia kadłubowych konfiguracji komputerowych oraz odpowiednich usług w zakresie serwisu i konserwacji
- ożywić działalność klubów użytkowników komputerów i utworzyć federację użytkowników
- zlikwidować „problemy" rządowe, resortowe i inne finansowane centralnie prace badawcze i wdrożeniowe — na rzecz niskoprocentowego, kredytowego finansowania tego typu prac przez banki na określony czas (z koniecznością spłacania kredytów drogą sprzedaży opracowań i wdrożeń)
- zlikwidować wszelkiego rodzaju koordynację w zakresie informatyki
- utworzyć centralny bank informacji o zbędnym sprzęcie informatycznym, wolnych mocach obliczeniowych oraz systemach oprogramowania (wykorzystać np. łamy INFORMATYKI do publikowania wszelkiego rodzaju ogłoszeń na ten temat)
- określić właściwy stan prawny informatyki i jej zastosowań
- zrewidować programy kształcenia w zakresie informatyki i zrezygnować z ilościowego na rzecz jakościowego kształcenia, zgodnie z konkretnymi potrzebami w tym zakresie
- stymulować w okresie kryzysu — poprzez odpowiednią politykę kredytową oraz wszelkiego rodzaju fundusze rozwoju — te kierunki zastosowań informatyki, które sprawdziły się u nas i w innych krajach i które wynikają z konkretnych potrzeb i uwarunkowań społecznych i ekonomicznych; np.:

— systemy komputerowej obsługi rent, emerytur, kredytów bankowych, klientów poczty, PKO itp.

— usługi teleinformatyczne, wykorzystujące sieć telegraficzną

— systemy informacji kierownictwa przedsiębiorstw, zrzeszeń, itp.

— ogólnodostępne usługi teleinformatyczne

— kompleksowe usługi EPD.

Szybkie efekty wychodzenia z kryzysu w zakresie informatyki można by w krótkim stosunkowo czasie uzyskać w następujących dziedzinach:

— produkcji mini- i mikrokomputerów w oparciu o struktury mikroprocesorowe i własne urządzenia peryferyjne

— produkcji sprzętu teleinformatycznego

— produkcji urządzeń peryferyjnych

— produkcji urządzeń redagujących teksty i mikrokomputerowych maszyn do pisania, elektronicznych dalekopisów itp.

— produkcji inteligentnych terminali dla handlu, miernictwa, diagnostyki w medycynie, rolnictwie, geologii i górnictwie

— oprogramowania specjalistycznego

— baz danych

— systemów rozproszonych

— systemów sieciowych (wielokomputerowych)

— inteligentnych sieci transmisji danych.

Te kierunki działania winny być wsparte pracami naukowo-badawczymi i potencjałem produkcyjnym i kadrowym informatyki. Reforma gospodarcza powinna stworzyć ku temu odpowiednie bodźce ekonomiczne, a administracja państwowa poprzez odpowiednią politykę może przyczynić się do rozwoju wymienionych dziedzin. Środowisko informatyczne musi być jednak świadome swoich celów i stojących przed nim zadań.

CZESŁAW SYC

robotron

Automat księgująco-fakturujący
wszechstronny specjalista

Wszędzie poszukiwani są specjaliści, szczególnie ci obrotni.
Nasz A 5120 dotrzymuje tego, co obiecujemy.
Na przykład w handlu (opracowywanie zleceń, dyspozycje towarowe, zbieranie danych rachunkowych i dotyczących dostaw), w przemyśle (bilansowanie robocizny i płac, dostawy i ekspedycja towarów, zbieranie i śledzenie realizacji zamówień, rachunkowość), w centralach obliczeniowych (gromadzenie danych), w komunikacji (opracowywanie informacji i danych, naprawa pojazdów samochodowych) i w finansach. To jednak jeszcze nie wszystko.
Automat księgująco-fakturujący przy połączeniu telekomunikacyjnym pracuje jako urządzenie końcowe. Składa się z monitora i zawiera (alternatywnie) pamięć dyskową lub kasetową. Jeszcze wydajniejszy jest nasz komputer po połączeniu z drukarką i dalszymi urządzeniami dyskowymi lub kasetowymi.

Zebrane dane mogą być przekazywane na peryferyjny nośnik danych lub bezpośrednio na centralną maszynę matematyczną. I jeszcze jedno: koncepcja pola ekranu czyni A 5120 szczególnie przydatnym do gromadzenia danych masowych. Prefabrykowane zespoły programowe lub kompletne systemy programowe codzienną pracą udowodnią jak sprawny jest nasz specjalista również w Waszym biurze.
Znajdźcie miejsce dla naszego automatu księgująco-fakturującego A 5120.

robotron

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

Automat
księgująco-fakturujący

A 5120