Nowe zasady prenumeraty!
Informacje wewnątrz numeru

6 1982

informatyka

WYDAWNICTWO NOT SIGMA

Perspektywa

Czasopismo naukowo-techniczne, chociaż jest na uboczu spraw publicznych, ogólnych, choć pozostaje na marginesie procesów społecznych — zależne jest od sytuacji w kraju. Także na jego kształt wpływa aktualny stan umysłów, zamierzeń, potrzeb. Dzisiaj — wobec stagnacji gospodarki i — jak na razie — braku optymistycznych perspektyw — coraz wyraźniejsza jest (także wśród informatyków) niechęć do podejmowania wysiłku i ryzyka. Poczucie bezradności jest coraz dosadniej manifestowane.

Stan ten wiąże się dla redakcji przede wszystkim ze zmniejszoną aktywnością autorów, szczególnie w sferze spraw ponadspecjalizacyjnych. Jakoś nikt nie próbuje się już wypowiadać na temat perspektyw informatyki, jej atutów i możliwości, wszyscy niemal starają się ominąć kontekst społeczny i gospodarczy swoich dokonań. Dominują więc problemy wąskospecjalistyczne.

Niepokojący jest fakt, iż brak zaufania do obecnych organizacji informatycznych powoduje zupełne niemal zahamowanie rozwoju, także — przepływu myśli technicznej (która wszak nie jest niezależna od ogólnej sytuacji). Niewydolność dotychczasowych struktur zdaje się obezwładniać, zniechęcać do odważnego myślenia, zamiast — co byłoby oczywiste — inspirować do poszukiwania nowych, lepszych form samostanowienia.

Czas informatycznej gigantomanii minął. Lepszym rozwiązaniem byłyby firmy nieduże, obrotne, szybko i efektywnie reagujące na zapotrzebowanie rynku. Wprawdzie nie ma u nas jeszcze mechanizmów ułatwiających życie takim firmom, niemniej ekonomia dobija się powoli swoich praw i niedługo już zapewne fachowość i samodzielność staną się podstawowym atutem.

Aby pomóc także początkującym inicjatywom, ogłosiliśmy w czerwcu br. uruchomienie na naszych łamach rubryki pod nazwą GIEŁDA INFORMACJI. Cały zebrany do połowy września plon przedstawia niniejszy numer (s. 27—28). Na jego podstawie można by uznać, iż wszystko w informatyce rozwija się pomyślnie, że nikt nie cierpi na brak bądź nadmiar informacji. Nie trzeba chyba pisać jak bardzo złudny byłby to wniosek.

Proponujemy informatykom bardziej zdecydowaną, ekspansywną walkę o przyszłość. Instynkt samozachowawczy powinien podpowiadać metody obrony. Łamy INFORMATYKI są do dyspozycji.

Redakcja

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

KOLEGIUM REDAKCYJNE

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, prof. doc. dr hab. Tadeusz WALCZAK

Materiałów nie zamówionych Redakcja nie zwraca.

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

Zakł. Graf. „Tamka". Zam. 188. Obj. 4,5 ark. druk. Nakład 5000 egz. Z-44.

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

Nosowski W., Zajączkowski P.: Zajęcia z podstaw informatyki

INFORMATYKA 1982, nr 6, s. 4.

Rozwiązanie zajęć dydaktycznych z podstaw informatyki, realizowane na Wydziale Mechanicznym Technologicznym Politechniki Warszawskiej. Scharakteryzowano tematykę wykładów oraz ćwiczeń audytoryjnych i laboratoryjnych, a także podano interesujące wyniki badań efektów nauczania i wynikające z nich wnioski.

Носовски В., Зайончковски П.: Занятия по основам вычислительной техники

Информатика 1982, № 6, стр. 4

Решение дидактических занятий по основам вычислительной техники, которые проводятся в Механическом Технологическом Отделении Варшавской Политехники. Охарактеризовано тематику лекций, лабораторных и аудиторийных практических занятий, а также представлено интересные результаты исследований эффектов преподавания и отсюда следующие предложения.

Żebrowski J.: Komputery osobiste

INFORMATYKA 1982, nr 6, s. 6.

Próba określenia cech komputerów osobistych. Podano klasyfikację oraz charakterystykę sprzętową i programową podstawowych grup komputerów osobistych, naświetlając perspektywy ich dalszego rzwoju, a także uruchomienia produkcji krajowej.

Жебровски Й.: Личные эвм

Информатика 1982, № 6, стр. 6

Попытка определения характерных черт личных эвм. Представлено классификацию, конфигурацию оборудования, а также программное обеспечение основных групп личных эвм, указывая перспективы их дальнейшего развития и введение в производство отечественной промышленности.

Zalewski J.: ADA — nowy język programowania (5). Krytyka języka

INFORMATYKA 1982, nr 6, s. 10.

Zakończenie cyklu z charakterystyką języka ADA. Na podstawie literatury przedstawiono usystematyzowany przegląd krytycznych opinii na temat tego języka.

Залевски Я.: ADA — новый язык программирования (5). Критика языка

Информатика 1982, № 6, стр. 10

Заключение цикла характеризующего язык ADA. На литературной основе указан систематизирован обзор критических мнений на счёт этого языка.

Starzyk H.: OSIRIS III w badanich statystycznych

INFORMATYKA 1982, nr 6, s. 14

Ogólna charakterystyka pakietu programów do obliczeń statystycznych OSIRIS III. Podano sposób użytkowania pakietu, ilustrując go przykładem zastosowania w dziedzinie badań spożycia leków.

Стажык Г.: OSIRIS III в статистических исследованиях

Информатика 1982, № 6, стр. 14

Общая характеристика пакета программ для статистических вычислений OSIRIS III. Указан способ пользования пакетом, иллюстрирован примером применения в области исследований потребления медикаментов.

Wieczorkowski K.: REKRUTACJA — system kwalifikowania kandydatów na studia

INFORMATYKA 1982, nr 6, s. 17.

Zwięzła charakterystyka systemu kwalifikowania kandydatów na studia wyższe, opracowanego i eksploatowanego na Uniwersytecie Mikołaja Kopernika w Toruniu. Podano orientacyjne koszty eksploatacji systemu oraz uzyskane efekty.

Вечорковски К.: REKRUTACJA — квалификационная система кандидатов в вузы

Информатика 1982, № 6, стр. 17

Краткая характеристика системы квалифицирования кандидатов в вузы, разработанная и применяемая Университетом им. Миколая Коперника в Торуне. Представлено ориентировочные расходы эксплуатации системы и полученные эффекты.

Gerwin K., Sukiennik J.: Zastosowanie systemu operacyjnego R 800 na standardowym zestawie MERA 9150

INFORMATYKA 1982, nr 6, s. 20.

Możliwość rozszerzenia zakresu zastosowań systemu MERA 9150 w wyniku zastosowania systemu operacyjnego R 800. Scharakteryzowano podstawową strukturę oprogramowania systemu R 800 oraz sposób jego wykorzystania w warunkach ograniczeń sprzętowych systemu MERA 9150.

Гервин К., Сукенник Й.: Применение операционной системы R 800 на стандартном комплекте MERA 9150

Информатика 1982, № 6, стр. 20

Возможность расширения сферы употребления системы MERA 9150 вследствие применения операционной системы R 800. Определено основную структуру программного обеспечения системы R 800 и способ его использования в условиях ограниченного оборудования системы MERA 9150.

Nosowski W., Zajączkowski P.: Teaching of data processing foundations

INFORMATYKA 1982, No. 6, p. 4.

A solution of data processing foundations teaching, realized at the Mechanic Technological Faculty of the Warsaw Technical University. Content of lectures, auditorial and laboratorial exercises, as well as interesting research results and conclusions are discussed.

Nosowski W., Zajączkowski P.: Informatikgrundlagenunterricht

INFORMATYKA 1982, Nr. 6, S. 4.

Eine Lösung des Informatikgrundlagenunterrichtes in der mechanisch-technologischen Fakultät der Warschauen Technischen Universität. Es wurden die Vorlesungen und Auditorium- und Laborübungen charakterisiert, sowie interessante Forschungsergebnisse der Unterrichtseffekte und daraus resultierende Beschlüsse angegeben.

Żebrowski J.: Personal computers

INFORMATYKA 1982, No. 6, p. 6.

An attempt of personal computer features defining. Classification as well as hardware and software characteristics of personal computer main groups, showing prospects of their future development and possibility of domestic manufacturing, are presented.

Żebrowki J.: Die Personalrechner

INFORMATYKA 2982, Nr. 6, S. 6.

Eine Definierungsprobe von Kennzeichen des Personalrechners. Es wurde die Klassifikation, sowie Hard- und Softwarecharakteristiken grundlegender Gruppen von Personalrechnern, mit Betonung der Weiterentwicklungsaussichten und Inbetriebnahme der einheimischen Produktion, angegeben.

Zalewski J.: ADA — a new programming language (5). Criticism against the language

INFORMATYKA 1982, No. 6, p. 10.

Termination of the publication cycle containing the ADA-language characteristics. A systematized, elaborated on the literature base, review of critical opinions concerning the language is presented.

Zalewski J.: ADA — eine neue Programmiersprache (5). Kritik der Sprache

INFORMATYKA 1982, Nr. 6. S. 10.

Beendigung des Publikationszyklus über die Charakteristik der ADA-Sprache. Es wurde auf Grund der Fachliteratur eine systematisierte Vorschau der kritischen Meinungen über diese Sprache vorgestellt.

Starzyk H.: OSIRIS III in statistical research

INFORMATYKA 1982, No. 6, p. 14.

General characteristics of the OSIRIS III program package for statistical computations. The way of the package use, illustrated with an application example in the field of medicaments consumption research, is presented.

Starzyk H.: OSIRIS III in statistischen Forschungen

INFORMATYKA 1982, Nr. 6, S. 14.

Eine allgemeine Charakteristik des OSIRIS III Programmpakets für statistische Berechnungen. Es wurde die Benutzungsweise des Pakets, illustriert mit einem Anwendungsbeispiel aus dem Bereich der Arzneiverbrauchforschung, angegeben.

Wieczorkowski K.: REKRUTACJA — a data processing system for high school candidates estimation

INFORMATYKA 1982, No. 6, p. 17.

Concise characteristics of the data processing system for high school candidates estimation, elaborated and operated at the Nicolas Copernicus University in Toruń. Evaluated costs of the system operation and observed effects are presented.

Wieczorkowski K.: REKRUTACJA — ein EDV-System für Qualifizierung der Hochschulkandidaten

INFORMATYKA 1982, Nr. 6, S. 17.

Eine zusammengefasste Charakteristik des EDV-Systems für Hochschulkandidatenqualifizierung, das in der Nikolaus Copernikus Universität erarbeitet wurde und laufend angewendet wird. Es wurden die Schätzungskosten des Systembetriebes und die erzielten Effekte angegeben.

Gerwin K., Sukiennik J.: Application of the R 800 operating system on the MERA 9150 standard configuration

INFORMATYKA 1982, No. 6, p. 20.

Extension possibility of the MERA 9150 application using R 800 operating system. Basic software structure of the R 800 system and the way of its use in hardware limitation conditions of the MERA 9150 are characterized.

Gerwin K., Sukiennik J.: Anwendung des R 800 Betriebssystems auf der MERA 9150 Standardkonfiguration

INFORMATYKA 1982, Nr. 6, S. 20.

Eine Erweiterungsmöglichkeit der MERA 9150 Anwendungen bei Ausnutzung von R 800 Betriebssystem. Es wurden die Grundsoftwarestruktur des R 800 Systems und seine Nutzung in Bedingungen der Hardwarebegrenzung von MERA 9150 charakterisiert.

Informatyka

zastosowania w gospodarce, technice i nauce

Nr 6
MIESIĘCZNIK
1 9 8 2
ROK XVII
Wrzesień

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

W NUMERZE: Strona

WIESŁAW NOSOWSKI
PIOTR ZAJĄCZKOWSKI

Wydział Mechaniczny Technologiczny
Politechnika Warszawska

Zajęcia z podstaw informatyki

W programie zajęć wielu uczelni technicznych występuje przedmiot pod nazwą „Podstawy informatyki". Zajęcia te mają stanowić wprowadzenie w tematykę informatyczną dla wszystkich studentów (poza specjalnościami informatycznymi). Dla większości studentów jest to pierwsze i jedyne zetknięcie się z informatyką podczas studiów. Świadomość tego faktu nakazuje zwrócenie szczególnej uwagi na treść i organizację tych zajęć.

W niniejszym artykule przedstawiony zostanie sposób, w jaki przeprowadzone zostały tego typu zajęcia na Wydziale Mechanicznym Technologicznym Politechniki Warszawskiej. Cel zajęć można sformułować następująco: czynne opanowanie podstawowego poziomu umiejętności programowania. Programowanie jest tu rozumiane jako technika rozwiązywania problemów przy wykorzystaniu maszyn cyfrowych. Przy takim ujęciu celu, do zajęć weszły następujące główne zagadnienia:

- **podstawy budowy i użytkowania komputerów**
- **metodyka programowania**
- **język programowania**
- **praktyka programowania.**

Zagadnienia te zostały rozłożone na okres 75 godzin zajęciowych, w tym: 15 godz. wykładów i 30 godz. ćwiczeń audytoryjnych w semestrze drugim oraz 30 godz. laboratorium w semestrze trzecim.

WYKŁADY

Tematyka wykładów objęła podstawy budowy i użytkowania komputerów oraz część teoretyczną metodyki programowania. Szczegółowy układ tematów był następujący:

- **pojęcia podstawowe informatyki**
- **struktura systemu liczącego**
- **urządzenia peryferyjne**
- **pamięci zewnętrzne**
- **języki programowania**
- **systemy operacyjne**
- **korzystanie z systemu liczącego**
- **metodyka programowania.**

Niewielka liczba godzin pozwoliła tylko na bardzo ogólne omówienie wymienionych tematów. Na pierwszych siedem tematów, które stanowią standardowy wstęp do informatyki, przeznaczone było 10 godzin, na metodykę programowania — 5 godz. W tym ostatnim temacie, szczególnie ważnym dla dalszego toku zajęć, poruszane były następujące zagadnienia:

- **podstawowe zasady programowania**
- **projektowanie zstępujące**
- **programowanie strukturalne**
- **osiąganie cech dobrego programu — niezawodności, sprawności, dobrego stylu.**

ĆWICZENIA AUDYTORYJNE

Celem ćwiczeń było opanowanie podstawowych zasad poprawnego programowania oraz opanowanie wybranego języka programowania. Językami nauczanymi były ALGOL 1900 i FORTRAN 1900. Do ich wprowadzenia użyto metody zaproponowanej przez K. L. Bowlesa [1]. Istota metody polega na posługiwaniu się w początkowym okresie nauczania języka maszyną wyposażoną w urządzenie kreślące. Z braku takiego urządzenia, trzeba było je sobie wyobrazić i wykonywać rysunki na tablicy. Sama jednak idea metody jest na tyle ciekawa, że warto ją pokrótce omówić. Jej główną zaletą jest możliwość obejrzenia wyników programów w postaci rysunków, co najsilniej oddziałuje na wyobraźnię studentów i ułatwia zrozumienie działania kolejno wprowadzanych konstrukcji językowych.

Pierwszymi wprowadzanymi instrukcjami języka są cztery instrukcje sterujące pisakiem w prostokątnym układzie współrzędnych:

MOVE TO (X,Y) — ustaw pisak na pozycji o współrzędnych (X,Y), bez kreślenia

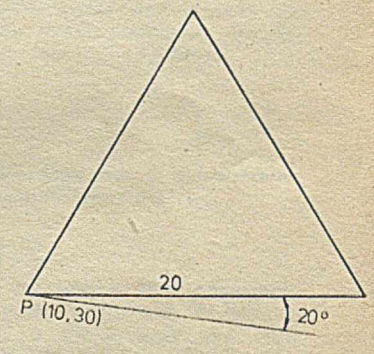
TURN TO (A) — ustaw kierunek kreślenia na kąt A w stosunku do osi x

MOVE (D) — wykreśl odcinek prostej o długości D

TURN (B) — zmień kierunek kreślenia o kąt B w stosunku do ostatniego kierunku.

Pojęcie programu zostaje wprowadzone na przykładach mających na celu rysowanie regularnych figur geometrycznych, np.:

```
      MASTER TROJKĄT
C     KREŚLENIE TRÓJKĄTA RÓWNOBOCZNEGO
C     USTAWIENIE POCZĄTKOWE
      MOVE TO (10,30)
      TURN TO (20)
C     KREŚLENIE BOKÓW
      MOVE(20)
      TURN (120)
      MOVE (20)
      TURN (120)
      MOVE (20)
      STOP
      END
```



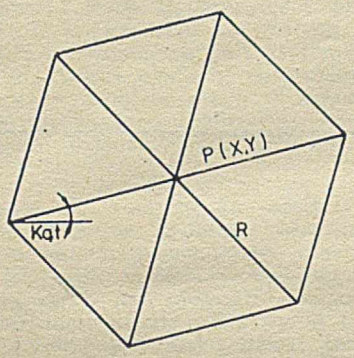
Bezpośrednio po tym wprowadzone zostaje pojęcie procedury i te same zadania można wykonywać w postaci procedur początkowo bez parametrów, a następnie z para-

[1]) BOWLES K. L.: MICROCOMPUTERS PROBLEM SOLVING USING PASCAL. Springer-Verlag, 1977

metrami. Od procedur z prostym układem parametrów, np. PROSTOKĄT (A,B), można tu stopniowo przechodzić do coraz bardziej uniwersalnych procedur, np.:

```
      SUBROUTINE PROSTOKĄT (A, B, X, Y, KĄT)
C     KRESLENIE DOWOLNEGO PROSTOKĄTA
      REAL A, B, X, Y, KĄT
C     A, B - DLUGOSCI BOKÓW PROSTOKĄTA
C     X, Y - WSPÓŁRZĘDNE WIERZCHOŁKA
C     KĄT - KĄT POŁOŻENIA PODSTAWY
C
C     USTAWIENIE POCZĄTKOWE
      MOVE TO (X,Y)
      TURN TO (KĄT)
C     KRESLENIE BOKÓW
      MOVE (A)
      TURN (90)
      MOVE (B)
      TURN (90)
      MOVE (A)
      TURN (90)
      MOVE (B)
      RETURN
      END
```

Kolejnym krokiem jest wprowadzenie wywoływania procedur przez inne procedury. Temat ten może być wprowadzony na przykładach figur złożonych z figur prostych, np.:




```
      SUBROUTINE SZESCIOKĄT (X, Y, R, KĄT)
C     KRESLENIE SZESCIOKĄTA
      REAL X, Y, R, KĄT
C     INTEGER I
C     X, Y - WSPÓŁRZĘDNE ŚRODKA
C     R    - DŁUGOŚĆ BOKU
C     KĄT - KĄT USTAWIENIA
C     TRÓJKĄT (A) - PROCEDURA KREŚLĄCA TRÓJKĄT RÓWNOBOCZNY
                    O BOKU A
C     USTAWIENIE POCZĄTKOWE
      MOVE TO (X, Y)
      TURN TO (KĄT)
C     KRESLENIE SZESCIOKĄTA
      DO 10 I = 1,6
      CALL TRÓJKĄT (R)
      TURN (180)
10    CONTINUE
      RETURN
      END
```

Równolegle z powyższymi tematami wprowadza się kolejno typowe konstrukcje językowe: instrukcje podstawienia, wyrażenia arytmetyczne, instrukcje cyklu, procedury funkcyjne, wyrażenia logiczne, instrukcje warunkowe. Do ich wprowadzenia można posłużyć się choćby następującymi przykładami:

- kreślenie dowolnego wielokąta foremnego
- kreślenie odcinka pomiędzy dwoma punktami
- kreślenie linii łamanej o losowym przebiegu.

Na tym zakończona zostaje pierwsza część zajęć. Jej celem było opanowanie procedur jako podstawowych jednostek wykonawczych programu i wyrobienie umiejętności myślenia proceduralnego. Interpretacja graficzna rozwiązywanych zadań znakomicie ułatwia osiągnięcie tego celu.

Dalszą część zajęć można przeznaczyć na omówienie podstawowych struktur danych oraz instrukcji wejścia i wyjścia. W wykonywanych tu zadaniach można już swobodnie posługiwać się procedurą jako naturalnym narzędziem programowania.

W całym ciągu zajęć należy stopniowo wdrażać poprawną metodykę programowania, uczyć formułowania problemu, wydzielania podproblemów składowych, kształtowania funkcji procedur, które będą je rozwiązywać, stosować dobry styl programowania. W drugiej części zajęć można wprowadzić elementy niezawodności i sprawności programu.

Zastosowanie przedstawionej metody dało bardzo zachęcające rezultaty. W oryginalnym wydaniu została ona zastosowana do nauki programowania w PASCALU. Jej adaptacja do potrzeb i możliwości innych języków wymaga pewnego nakładu pracy i przemyślenia całej koncepcji zajęć.

ĆWICZENIA LABORATORYJNE

Celem tych zajęć było pogłębienie i skonkretyzowanie — poprzez praktykę — znajomości języka i metodyki programowania.

Zadania studenckie wykonywane były na maszynie ODRA 1305 pod kontrolą systemu operacyjnego GEORGE 3 w trybie wsadowym. Zajęcia były przeprowadzone przy następujących założeniach:

- każdy student pracuje indywidualnie i wykonuje w semestrze dwa zadania, w tym jedno przewidujące wykorzystanie biblioteki podprogramów
- techniczna obsługa zadań wykonywana jest całkowicie przez personel Laboratorium Informatyki (perforowanie, wykonanie)
- Laboratorium Informatyki pracuje w trybie „open shop", a więc każdy student wykonuje swe zadania niezależnie od innych uczestników zajęć
- zasadnicza część zajęć ma charakter projektowania, podczas którego asystent prowadzi grupę kilku studentów.

W trakcie zajęć studenci zostali zapoznani również z podstawowymi właściwościami systemu operacyjnego GEORGE 3. Niewielka liczba godzin, jaką można było przeznaczyć na ten temat, nie pozwoliła na takie opanowanie języka zadań, aby można było wykonywać złożone zadania wsadowe. Dlatego też dla obsługi zadań studenckich zostały opracowane trzy makrokomendy, które obejmowały cały zakres obsługi zadań:

LETOEDIT — poprawa treści programu przechowywanego w pamięci zbiorów systemu za pomocą edytora systemowego; Makrokomenda ta utrzymywała w PZS tylko dwie ostatnie wersje poprawianego programu
LETOALGOL — kompilacja i (lub) wykonanie programu w języku ALGOL 1900.
LETOFORTRAN — jw. dla FORTRANU 1900.

Parametry dwu ostatnich makrokomend pozwalały m. in. na korzystanie z bibliotek, utworzenie programu binarnego i późniejsze jego wykonywanie oraz sterowanie czasem obliczeń i objętością wydruku.
Obydwie makrokomendy zawierały silnie rozbudowany aparat badania poprawności parametrów połączony z czytelną diagnostyką błędów (30 komunikatów).
W celu zbierania danych statystycznych o przebiegu wykonywania zadań, do wszystkich makrokomend włączono instrukcje powodujące rejestrowanie każdego ich użycia. W przeznaczonym do tego celu zbiorze rejestrowane były więc dane ewidencyjne użytkownika, rodzaj użytej makrokomendy i skutki jej użycia (ocena poprawności parametrów, kompilacji i wykonania). Po zakończeniu zajęć dane zawarte w tym zbiorze pozwoliły na uzyskanie następujących interesujących rezultatów:

	Średnio	Min.	Maks.
Liczba dni od początku zajęć do wykonania pierwszego zadania studenta	50	24	105
Liczba dni pomiędzy wykonaniem kolejnych zadań tego samego studenta	4,5	1	63
Liczba wywołań makrokomend w zadaniach studenta w całym okresie pracy	23	1	79

Ponadto stwierdzono, że:

- **95% wywołań makrokomend było poprawnych pod względem parametrów**
- **poprawianie programów za pomocą edytora było poprawne w 80%**
- **kompilacje programów ALGOLU były poprawne w 49%, a wykonanie tych programów — poprawne w 55%**
- **kompilacje programów w FORTRANIE były poprawne w 62%, a wykonania poprawne w 48%**
- **koszt wykonania jednego zadania wyniósł 100 zł (cena 1 s pracy arytmometru wynosi 1,5 zł)**
- **35% całości obciążenia systemu liczącego przypadła na ostatnie dwa tygodnie zajęć (15% czasu).**

Od strony metodycznej, na zajęciach położony był nacisk na dobre zaprojektowanie programu z uwzględnieniem zasad dobrego programowania, a w szczególności utworzenia poprawnej struktury programu, wprowadzenia mechanizmów podwyższających niezawodność i przestrzeganie dobrego stylu.

* * *

Główne spostrzeżenia i wnioski z zajęć są następujące:

- **Próba zastosowania do nauki języka programowania sposobu zaproponowanego przez Bowlesa dała bardzo zachęcające rezultaty. Przede wszystkim cenne jest tu szybkie opanowanie procedur, co pozwala uczynić bardziej efektywną dalszą część zajęć. Ponadto metoda ta pozostawia szerokie pole do wprowadzania własnych pomysłów przez prowadzących. Z tych też względów jest ona warta szerszego rozpowszechnienia.**
- **Główne zasady dobrego programowania mogą być wyjaśnione w ramach sześciogodzinnego wykładu. Wdrażanie tych zasad wymaga natomiast właściwego rozłożenia materiału na zajęcia ćwiczeniowe i laboratoryjne.**
- **Położenie nacisku na dobre zaprojektowanie programów wydłuża okres ich realizacji (średnio do 50 dni), lecz przynosi pozytywne skutki w postaci lepszej jakości programów.**
- **Przechowywanie programów w pamięci pomocniczej i ich poprawianie za pomocą edytora spotyka się początkowo z dużą nieufnością. Po jej pokonaniu edytor jest prawidłowo używany.**

JACEK ŻEBROWSKI
Łódź

Komputery osobiste

Personal computer — jak to pojęcie zręcznie przełożyć na język polski? Komputer osobisty? Nie brzmi to może najładniej, ale dość dobrze oddaje sens angielskiego określenia.

Takie urządzenia zdobywają od kilku lat coraz większą popularność w krajach zachodnich. Wielu specjalistów uważa, że cywilizacyjne skutki upowszechnienia komputerów osobistych mogą być tak doniosłe, jak przemiany spowodowane wprowadzeniem na szeroką skalę telefonów lub radia. Jeżeli dalszy rozwój komputerów osobistych będzie odbywał się w dzisiejszym tempie, to mogą one poważnie wpłynąć nie tylko na sposób myślenia i życie codzienne jednostek, ale również na organizację pracy i formy życia społecznego. Jest prawdopodobne, że obserwujemy początek kolejnego etapu „rewolucji informatycznej". Tym razem szalony postęp technologii półprzewodników dał do rąk przeciętnego człowieka jego własny, osobisty komputer. Podkreślmy: komputer coraz lepiej przystosowany do potrzeb użytkownika.

PO CZYM POZNAĆ KOMPUTER OSOBISTY

Nie ma jeszcze powszechnie uznawanej definicji komputera osobistego. Aby rozstrzygnąć, jaki komputer jest, a jaki nie jest „osobisty" — najczęściej bierze się pod uwagę następujące czynniki:

niskie koszty zakupu: (od 300 do 5000 $), które decydują o możliwości nabycia takiego sprzętu prywatnie lub przez przedsiębiorstwo z przeznaczeniem dla konkretnego pracownika

zarządzanie zasobami: użytkownik (często będący posiadaczem) sam decyduje o sposobie i czasie wykorzystania sprzętu

komunikacja z człowiekiem: wymagana jest łatwość posługiwania się komputerem przez osobę nie będącą informatykiem; obsługa sprzętu powinna być jak najprostsza, a oprogramowanie — łatwe do zrozumienia.

Komputer osobisty powinien być, oczywiście, mały i lekki. Musi być wysoce niezawodny, zawsze gotów do użycia bez udziału specjalistycznej obsługi technicznej oraz łatwy w naprawie. Jedną z firm, która rozwiązała zagadnienie serwisu w skali ogólnoświatowej jest RADIO-SHACK. Krok dalej zrobiła firma APPLE, która dostarcza specjalne oprogramowanie („*teachware*") ułatwiające użytkownikowi samodzielne opanowanie eksploatacji komputera.

Jest rzeczą charakterystyczną, że wiele modeli komputerów osobistych wyposażono w systemy graficzne, które pozwalają uzyskać w prosty sposób na ekranie monitora (często kolorowego) rysunki, wykresy, tabele itp. Jest również możliwe generowanie dźwięków sterowane programem. Wszystko to znacznie ułatwia przepływ informacji od maszyny cyfrowej do człowieka. Komputer jest coraz bliżej użytkownika.

Warto dodać, że komputery osobiste zazwyczaj nie są sprzedawane bezpośrednio przez wytwórców, lecz w zwykłych sklepach.

PRZYKŁADY ZASTOSOWAŃ

Pierwsze komputery osobiste (w obecnej postaci) zaczęto produkować w 1977 roku. Dziś można je znaleźć niemal wszędzie: w gabinetach naukowców, pracowniach inżynierów, w laboratoriach naukowych i przemysłowych, w domach, szkołach, klubach, w małych przedsiębiorstwach. W bardziej rozbudowanych systemach pojawiają się małe bazy danych i systemy informacyjno-wyszukiwawcze. Komputery osobiste mogą pracować jako końcówki w lokalnych sieciach teleinformatycznych. Szybko wkraczają w takie dziedziny, jak przetwarzanie tekstów czy obrazów.

Upowszechnienie się komputerów osobistych złagodziło co najmniej dwa drażliwe problemy klasycznej informatyki. Myślę tu o mitach „kosztownego czasu maszynowego" i „kapłańskiej roli informatyków". Czas maszynowy komputerów osobistych jest niemalże darmowy ze względu na szczególnie małe koszty amortyzacji i bieżącej eksploatacji. Amortyzacja jest niska dzięki temu, że sprzęt i oprogramowanie są relatywnie tanie. Natomiast eksploatacja nie jest obciążona płacami personelu, gdyż użytkownik sam obsługuje komputer, będący dla niego narzędziem pracy (opłacanej z innego tytułu).

Zmieniła się również rola informatyka, ponieważ w niespotykanym wcześniej stopniu komputer dostosował się do możliwości człowieka. Dotąd informatyk oddzielał użytkownika od maszyny cyfrowej. Dziś komputer osobisty jest wspólnym celem ich wysiłków.

SPRZĘT

Komputery osobiste można podzielić na trzy klasy. Nazwijmy je umownie: amatorską, uniwersalną i zawodową. Amatorskie są najprostsze i najtańsze (200—300 $). Natomiast zawodowe są to stosunkowo wszechstronne systemy o dość dużej mocy obliczeniowej; są zatem dość drogie (ok. 4—5 tys. $). Klasa uniwersalna (cena rzędu 1000 $), zdobyła największą popularność i dalej szybko się rozwija [1]).

[1]) Większość podanych w artykule cen dotyczy rynku USA w 1980 r.

Jednostka centralna komputera osobistego opiera się na układach scalonych wielkiej skali integracji. W istocie rzeczy jest ona mikrokomputerem, charakteryzującym się dużą pojemnością pamięci RAM, często zawierającym wyspecjalizowane układy scalone, ułatwiające współpracę z klawiaturą i monitorem ekranowym czy innymi urządzeniami zewnętrznymi.

Jest bardzo charakterystyczne, że do budowy jednostek centralnych wykorzystuje się zaledwie kilka typów mikroprocesorów, choć do wyboru jest ich kilkadziesiąt. Najczęściej używany 6502, opracowany jest przez firmę MOS TECHNOLOGY [2]). Istnieją dwa powody tej popularności: architektura 6502 pozwala na stosunkowo łatwą implementację języków wyższego poziomu, a ponadto istnieje możliwość dostępu do pamięci operacyjnej w czasie cyklu mikroprocesora, co bardzo ułatwia odnawianie (ang. *refresch*) obrazu wyświetlanego na ekranie bez utraty szybkości przetwarzania. Często stosowane są również mikroprocesory Z80 (ZILOG), oraz INTEL 8080. Ciekawe jest, że mikroprocesor MOTOROLA 6800, który znalazł szerokie zastosowanie w sprzęcie profesjonalnym, w komputerach osobistych występuje rzadko.

Każdy komputer osobisty jest wyposażony w klawiaturę alfanumeryczną. Część cyfrowa tej klawiatury jest zazwyczaj wyodrębniona.

Do zobrazowania informacji służy przeważnie monitor ekranowy, często stanowiący jedną całość z jednostką centralną i klawiaturą. Wiele rozwiązań pozwala na wykorzystanie — jako monitora — telewizora, czarno-białego lub kolorowego. Obniża to znacznie koszt zestawu.

Pamięć zewnętrzną w wersjach oszczędnościowych stanowi zwykły magnetofon kasetowy. Coraz częściej jednak używane są pamięci na małych dyskach elastycznych (*mini-floppy*). Sprzężenie z urządzeniami zewnętrznymi (komunikacja szeregowa), jak: drukarka, modem itp. jest najczęściej rozwiązywane wg standardu RS-232C (V-24). Natomiast do tworzenia rozbudowanych systemów, gdzie wymagane są większe prędkości przesyłu informacji, stosuje się różne magistrale, np. S-100, IEEE-488, a nawet firmowe, nieznormalizowane.

Konstrukcja dobrego komputera osobistego powinna pozwalać na modułowe rozszerzanie półprzewodnikowej pamięci operacyjnej — zarówno ROM — zawierającej firmowe oprogramowanie podstawowe, jak i RAM — przeznaczonej na programy i dane użytkownika. Jest to w praktyce bardzo ważna sprawa, tak jak i elastyczność całego systemu. Występujące tu ograniczenia są przeważnie zręcznie maskowane przez wytwórcę, co bywa przyczyną rozczarowań.

Z roku na rok szybko wzrasta złożoność układów scalonych. Komputer osobisty tak zmienia swoje wymiary, że bez przesady staje się „kieszonkowy", a z drugiej strony kalkulatory programowane uzyskują coraz to nowe możliwości obliczeniowe. Powstaje więc pytanie, gdzie leży granica między tymi urządzeniami? Jeszcze niedawno taką wątpliwość rozstrzygało się następująco: kalkulator operuje tylko liczbami, a komputer — znakami alfanumerycznymi. Dziś widać już wyraźnie, że obie klasy urządzeń dążą do wspólnego punktu. Komputery osobiste mieszczą się już bowiem w kieszeni, a kalkulatory akceptują litery. W obu przypadkach można dołączać urządzenia zewnętrzne, np. pamięci magnetyczne.

Typowe komputery osobiste średniej klasy porównuje poniższa tabela.

TYPOWE URZĄDZENIA ZEWNĘTRZNE

Klasyczną pamięcią zewnętrzną komputerów osobistych był — i jest (jak już wspomniano) — zwykły magnetofon kasetowy. Jest on łatwo dostępny i tani, ma jednak liczne wady: ręczne odszukiwanie programów na kasecie, małą niezawodność, niewielką gęstość zapisu, dość wysoką stopę błędów. Stąd coraz częściej stosowaną pamięcią zewnętrzną jest jednostka małych dysków elastycznych [3]).

[2]) Dziś należy ona do COMMODORE, znanego producenta komputerów osobistych

[3]) o średnicy 5 cali

Wybrane dane techniczne niektórych komputerów osobistych

Typ komputera osobistego (producent, cena)	Typ mikroprocesora	Pamięć operacyjna w K bajtach podstawowa/dodatkowa		Minidyski elastyczne	Dołączenie magnetofonu kasetowego	Wsuwane moduły pamięci ROM	Zobrazowanie na ekranie monitora		Rodzaj monitora ekranowego
		RAM	ROM				grafika: liczba punktów, liczba kolorów	tekst: liczba znaków w linii × liczba linii	
ATARI 800 (ATARI, 999 $)	6502	16/48	18/26	88 KB 4 jedn.	+	+	320×192 16 kol.	40×24	odbiornik TV czarno-biały lub kolorowy
APPLE II (APPLE COMP., 1195 $)	6502	16/48	12/—	108 KB 4 jedn.	+	—	40×40 16 kol. 280×100 6 kol.	40×24	odbiornik TV czarno-biały lub kolorowy (konieczny modulator)
PET (COMMODORE BUS. MACH., 995 $)	6502	16/32	14/14	88 KB 4 jedn.	+	—	64 znaki graficzne	40×25	czarno-biały, wbudowany
EXIDY SORCERER (EXIDY, 995 $)	Z 80	8/48	12/—	88 KB 4 jedn.	+	+	512×240 czarno-białe	64×30	odbiornik TV czarno-biały
TRS-80 (RADIO-SHACK, 849 $)	Z 80	16/48	12/12	90 KB 4 jedn.	+	—	128×48	64×16	czarno-biały, wbudowany
COMPUCOLOR II (COMPUCOLOR CORP., 1695 $)	8080	8/32	14/—	51 KB 2 jedn.	—	—	128×128 8 kol. lub 64 znaki	64×32	kolorowy, wbudowany
TI 99/4 (TEXAS INSTRUMENTS, 1150 $)	9900	16/16	26/56	80 KB 1 jedn.	—	+	192×256 16 kol.	32×24	kolorowy, nie wbudowany

Uwaga: wszystkie urządzenia mają sprzęg RS-232C oraz translator języka BASIC.

Jest to urządzenie dość tanie (400—600$) z przeciętnym czasem dostępu rzędu 0,5 s. Niewielkie wymiary i mały ciężar są szczególnie atrakcyjne, zwłaszcza, że pojemność takiego dysku wynosi przeciętnie ok. 100 K bajtów.

Jako znakowe urządzenie wyjściowe stosowane są często drukarki termiczne. Są tanie, niezawodne i coraz bardziej trwałe. Ich powolność nie stanowi tu problemu, zwłaszcza że jest rekompensowana cichą pracą. Poważną wadą tych drukarek jest natomiast wysoki koszt termoczułego papieru. Dlatego coraz szerzej upowszechniają się małe drukarki znakowo-mozaikowe. Nowsze ich modele zawierają często własny mikroprocesor, co pozwala na druk nie tylko różnych rodzajów pisma (np. pochyłego, pogrubionego itp.), ale również na punktowe drukowanie obrazów. Takie wyjście graficzne może z powodzeniem zastępować drogą autokreślarkę (*plotter*). Przykładem może być drukarka EPSON MX-80, w cenie ok. 650 $. Należy podkreślić, że w komputerach osobistych nie są stosowane dziurkarki i czytniki taśmy papierowej.

OPROGRAMOWANIE

W tej dziedzinie wiele problemów jest odbiciem sytuacji, w jakiej znajdują się również „konwencjonalne" komputery; kryzys w dziedzinie oprogramowania jest bowiem zjawiskiem ogólnoświatowym i duże ośrodki komuterowe często z konieczności dorabiają oprogramowanie do kupionego sprzętu. Natomiast użytkownik komputera osobistego, oczywiście, nie jest w stanie opracować np. translatora potrzebnego mu języka. W wyniku dużego popytu na oprogramowanie, na świecie powstało wielu tzw. niezależnych producentów oprogramowania. Dowodem tego, że doceniają oni również rynek komputerów osobistych jest fakt istnienia ponad 100 firm oferujących oprogramowanie dla APPLE. Jednak poważną część oprogramowania — zwłaszcza programy użytkowe, przeznaczone dla konkretnych zastosowań — piszą „amatorzy", często w czasie wolnym od pracy. Dzięki temu jest ono tanie.

Głównym problemem oprogramowania przygotowywanego dla komputerów osobistych przez zawodowych informatyków jest ochrona praw własności i sposób sprzedaży. Typowe sposoby opracowane dla rynku „klasycznych" komputerów nie są tutaj skuteczne. Nie ma np. odpowiednio tanich i prostych środków technicznych utrudniających bezprawne kopiowanie oprogramowania.

W praktyce najczęściej dąży się do tego, aby koszt zakupu był niższy od kosztu skopiowania programu. Sprzyja temu rozpowszechnianie oprogramowania zapisanego na dyskach elastycznych. Koszt dyskietki z prostym programem aplikacyjnym wynosi ok. 30 $, a dobry kompilator rozszerzonego BASICA (na dwóch dyskach elastycznych) kosztuje ok. 300 $ [1]). Dla porównania: koszt niezapisanego dysku elastycznego nie przekracza 5 $.

Oprogramowanie firmowe dostarczane przez producenta komputera osobistego jest zawarte w pamięci ROM. Dzięki temu rozruch sprzętu po załączeniu do sieci elektrycznej jest natychmiastowy i zupełnie nie absorbuje użytkownika. W tej postaci dostarczany jest przede wszystkim interpreter BASICA — od bardzo prostego, aż do zaawansowanego. Często dochodzi do tego program monitora, pozwalający na korzystanie z pamięci na dysku elastycznym. Dostępne są również asemblery, pomimo że ich użyteczność w tym przypadku jest dyskusyjna. Zdarzają się komputery osobiste nawet bez BASICA — choć APPLE III dysponuje asemblerem, rozszerzonym BASICEM, FORTRANEM 77 i PASCALEM!!

[1]) Przykład dotyczy opracowania renomowanej firmy produkującej oprogramowanie

MOŻLIWOŚCI I BARIERY DALSZEGO ROZWOJU

Dziś producenci ze względów handlowych nie ujawniają swoich zamierzeń technicznych. Ogólna prawidłowość, wynikająca z postępu technologii jest taka, że następuje stały wzrost parametrów użytkowych sprzętu w przeliczeniu na jednostkę kosztu.

Przede wszystkim należy spodziewać się szerszego wprowadzenia mikroprocesorów 16-bitowych. Oczekiwany jest dalszy postęp w budowie pamięci masowych oraz tanich drukarek mozaikowych. Kładzie się duży nacisk na elastyczność konfiguracji sprzętu, choć trudno mieć nadzieję na unifikację sprzęgów (ang. *interface*). W najbliższej perspektywie czasu będą się upowszechniały urządzenia komunikacji głosowej człowieka z maszyną cyfrową. Problem syntezy dźwięku przez komputer osobisty wydaje się już technicznie opanowany (przy umiarkowanych cenach). Natomiast urządzenia do analizy głosu człowieka, aczkolwiek dla komputerów osobistych są już wytwarzane, są nadal relatywnie drogie.

Poprzednio już wspomniano o niedostatku, wręcz głodzie oprogramowania. Pewne nadzieje wiąże się z rozwojem systemów produkcji oprogramowania i środków wspomagania programisty. Z drugiej strony — przenośność oprogramowania dla komputerów osobistych to problem, który zaledwie jest sygnalizowany. Coraz częściej mówi się, że w opisywanej dziedzinie następcą BASICA będzie PASCAL, który pozwoli na rozwiązywanie bardziej złożonych zagadnień. Przewiduje się dalszy rozwój urządzeń graficznych oraz zastosowanie oprogramowania lepiej wykorzystującego możliwości ekranu. Powstaje także oprogramowanie umożliwiające pracę komputerów osobistych w sieci teleinformatycznej — a więc zamieniające je w terminale. Bardzo charakterystycznym zjawiskiem jest powstanie tzw. *„teachware"* czy *„courseware"* — programów uczących użytkownika m.in. korzystania z dostarczonego oprogramowania.

Bardzo ważną sprawą jest dostosowanie oprogramowania do możliwości użytkownika, który nie jest informatykiem. Opis programów musi być wówczas obszerny i przystępny; żargon, skróty myślowe — są wykluczone. Oprogramowanie komputerów osobistych powinno zostać szybko rozwinięte, ale trudności są tu znacznie większe niż przy opracowywaniu nowego sprzętu. Ostatnio napotkano nowe nieoczekiwane przeszkody: komputer osobisty tak szeroko się upowszechnił, że w wielu państwach zażądano, aby urządzenie to spełniało również wszystkie wymogi bezpieczeństwa użytkowania pod względem elektrycznym. Ponadto, zakłócenia radioelektryczne powstające podczas jego pracy powinny spełniać odpowiednie normy.

KOMPUTERY OSOBISTE W POLSCE?

Czy pisanie o tych zagranicznych nowościach techniki ma sens, gdy brakuje papieru dla INFORMATYKI, a kraj znajduje się w bardzo trudnej sytuacji? Sądzę, że tak, ponieważ takie urządzenia jak komputer osobisty mogłyby mieć w Polsce spore szanse. Reforma gospodarcza może skłonić ludzi do nowego spojrzenia na informatykę i komputery. Konieczność wymiernego ekonomicznie podejścia do produkcji zmusi przedsiębiorstwa (zwłaszcza niewielkie) do szukania tanich, prostych i praktycznych urządzeń informatyki. Będą one potrzebne nie tylko do prac inżynierskich, badań, kontroli, pomiarów itp., ale również jako środek do usprawnienia rozliczeń finansowo-materiałowych oraz szeroko rozumianej administracji i zarządzania.

Wielki rynek dla opisywanego sprzętu stanowi szkolnictwo. Tani, prosty i masowo produkowany komputer stanowi jedyną drogę do informatycznego wychowania i nowoczesnego przygotowania zawodowego dla najbliższych pokoleń. Oceniam, że już dziś szkolnictwo wyższe zakupiłoby natychmiast co najmniej kilkaset komputerów osobistych o większej mocy obliczeniowej, a ponadto kilka tysięcy tanich i prostych. Potencjalny rynek, bardzo ważny ze społecznego punktu widzenia, stanowi też szkolnictwo średnie.

Niektóre tanie odmiany komputerów osobistych mogą interesować szerokie grono osób prywatnych: specjalistów, naukowców, hobbystów. Nawiasem mówiąc, ci ostatni nigdy nie byli doceniani na krajowym rynku. A liczne doświadczenia uczą, że zaspokajanie potrzeb hobbystów to świetny interes.

Wszystkie powyższe uwagi poczyniłem przy założeniu, że ceny komputerów osobistych byłyby w racjonalnych proporcjach do cen innych dóbr. Oceniam, że cena przeciętnego zestawu takiego sprzętu powinna być porównywalna z ceną telewizora kolorowego. Nie jestem natomiast pewien, czy wyprodukowanie wersji oszczędnościowej w cenie odpowiadającej np. telewizorowi czarno-białemu, będzie możliwe ze względu na słabość krajowego przemysłu elektronicznego.

Pod koniec 1981 r. cześć niezbędnych dla komputera osobistego elementów było już w Polsce wytwarzanych (do innych celów), niektóre elementy zostały opracowane (przy różnym stopniu zaawansowania prototypów), a reszta leży w zasięgu możliwości posiadanej technologii. Poważny niepokój budzą zaniedbania i opóźnienia w pracach nad krajowym systemem mikroprocesorowym. Oparcie się na rodzimym „8080" to jedyne sensowne wyjście. Wiadomo, że niektóre ośrodki dysponują odpowiednim oprogramowaniem dla tego mikroprocesora — m.in. interpreterem BASICA.

Podstawowe urządzenie peryferyjne — monitor ekranowy z klawiaturą — jest wprawdzie dostępne, ale celowe (i realne) wydaje się zrobienie taniego wyjścia na telewizor. Na Targach Poznańskich demonstrowano małą, uproszczoną drukarkę mozaikową (DZM-80). Należy się jednak spodziewać kłopotów z masową pamięcią zewnętrzną. Niewątpliwie najodpowiedniejszy byłby mały dysk elastyczny, ale powszechnie wiadomo, że wymaga on znacznego importu części. Magnetofon kasetowy miałby rację bytu tylko z wersją oszczędnościową, najprostszą. Można zastanawiać się nad pamięcią kasetową podobną do PK-1.

Czy krajowy przemysł komputerowy byłby w stanie podjąć produkcję komputerów osobistych? Moim zdaniem: z technicznego punktu widzenia — tak; z punktu widzenia ekonomii i organizacji — raczej nie. Nie wiem, czy przemysł ten uzna komputer osobisty za rzecz wartą uwagi. Ale niezależnie od tego, produkcja dla informatyki musi ulec wielkim przeobrażeniom. Na to trzeba czasu. Wytwarza się pustka. Kto może wkroczyć na ten obszar gospodarki?

Widzę wielką szansę przedsiębiorstw polonijno-zagranicznych. Wiadomo, że działają już w zbliżonych dziedzinach, nawet na styku informatyki i automatyki. Mimo trudności, jakich wszyscy doświadczamy, produkują dobre, nowoczesne wyroby, cieszące się uznaniem odbiorców. Czy dziedzina komputerów osobistych wyda im się dobrym interesem? Trudno być tu prorokiem. W każdym razie wierzę, że mały, tani komputer ożywiłby naszą informatykę.

LITERATURA

[1] Bernhard R.: Computers (III). IEEE Spectrum, No. 1, 1981

[2] Holmen M. G.: Who Is Using Personal Computers? IEEE Trans. Syst. Man and Cybern., No. 8, 1980

[3] Raskin J., Whitney T.: Perspectives on Personal Computing. COMPUTER, January, 1981

[4] Warren C.: Inside and Out, Personal Computers Aim to Please Their Users. EDN, 5 February 1980

(Ponadto wykorzystano literaturę firmową i ogłoszenia reklamowe)

JANUSZ ZALEWSKI
Instytut Badań Jądrowych
Warszawa

Nie można dopuścić, aby ten język w obecnej postaci był używany w zastosowaniach, których cechą krytyczną jest niezawodność.

C.A.R. Hoare
1981, ACM Turing Award Lecture

ADA — nowy język programowania (5)

Krytyka języka

W dotychczasowych czterech artykułach na temat języka ADA omówiono go dość pobieżnie. Rzeczywistych problemów dotyczących zarówno definicji języka, jak i programowania w nim jest znacznie więcej, jednak omówienie bardziej szczegółowych zagadnień należy odłożyć do czasu, aż zainteresuje się nim większa liczba specjalistów w Polsce. W niniejszym artykule, którym kończymy prezentację ADY, przedstawimy kilka negatywnych opinii dotyczących zarówno właściwości języka, jak i konsekwencji — także społecznych, związanych z jego stosowaniem.

Najwięcej uwagi poświęcono dotychczas wyeliminowaniu niedoskonałości samego języka, tj. jego różnych konstrukcji. Wiele niejasności dotyczy również treści podręcznika. Szereg kontrowersji wzbudziła także specyfikacja wymagań, jakie miał spełniać język. Najciekawsze są jednak rozważania dotyczące przewidywanych konsekwencji społecznych, jakie może wywołać rozpowszechnienie języka ADA.

KONSTRUKCJE JĘZYKOWE

W krótkim artykule nie sposób nawet wspomnieć o wszystkich zastrzeżeniach do projektu języka, których było bardzo wiele. Obecna wersja języka (na razie niedostępna w Polsce) jest już trzecią z kolei opublikowaną w ciągu czterech lat. Wiele uwag do początkowych propozycji już w niej uwzględniono. Nie przedstawimy zatem szczegółowych zastrzeżeń dotyczących poszczególnych konstrukcji językowych, lecz jedynie sformułowania bardziej ogólne.

Przykładowo — autorzy definicji języka i ich współpracownicy za jego mocną stronę uważają silne mechanizmy typizacji i abstrakcji danych. Zdarzają się jednak głosy kwestionujące samą podstawę tego stwierdzenia i celowość opartego na nim podejścia. Mówiąc dokładniej — np. zadeklarowanie typu zmiennej i każdorazowe sprawdzenie zgodności typu przypisywanej jej wartości powinno zwiększyć niezawodność programów. Deklaracje jednakże naruszają psychologiczne zasady ludzkiego myślenia, ponieważ oddzielają informację o zmiennej od samego użycia tej zmiennej. Z tego względu nie można podać wystarczających dowodów ani co do istnienia, ani co do braku zależności między niezawodnością programów a typizacją danych [8] [1]).

Wydaje się, że bardziej sensowne zarzuty dotyczyły konkretnych konstrukcji językowych. Za poważniejszy uznałbym często wymieniany brak ortogonalności, a więc nadmiar różnych konstrukcji spełniających podobną, nierzadko identyczną rolę.

Przykładowo — wzmocnieniu ortogonalności służyłoby rozszerzenie właściwości typów i podprogramów (co zrobiono, np. dla funkcji w ALGOLU-68 i w LISPIE), a więc postulowanie [2] istnienia ich typów, operacji przypisania i równości oraz możliwości używania ich jako zmiennych i parametrów. Szczególnie ostatnia właściwość jest ważna ze względu na ujednolicenie wszystkich innych mechanizmów parametryzacji, a w szczególności zastąpienie jednostek generycznych wprowadzonych w ADZIE lub klauzuli **new**. Zresztą, to samo stwierdzenie dotyczy pakietów i zadań.

Inny poważny zarzut dotyczy przeciążania, które z natury jest sprzeczne z wymaganiem silnej typizacji języka. Zbyt dużo jest także słów zastrzeżonych, które w całości stanowią ok. 10% podstawowego słownika języka angielskiego (Basic English). Sytuację pogarsza fakt, że niektóre słowa są używane w różnych, lecz bliskich znaczeniach, np. klauzula **new** służy do określenia typu pochodnego, pełni rolę alokatora i określa rozwinięcie jednostki generycznej.

Na ogół panuje przekonanie, że mechanizmy współbieżności zawarte w ADZIE stanowią zadowalające rozwiązanie dla wielu przypadków praktycznych. Wskazywano jednak szereg sposobów dalszego polepszenia tych właściwości języka. Język mógłby zyskać na prostocie, skuteczności i ortogonalności, gdyby wyraźniej wydzielić trzy konstrukcje dotyczące współbieżności, określające porządek (ang. sequencing), wybór i powtarzalność operacji związanych ze współbieżnością. Można by usunąć wtedy ze składni klauzulę **else** i instrukcję **delay** [9]. Inne konstrukcje, takie jak instrukcja **abort** i ogłaszanie wypadków w innym zadaniu, krytykuje się, gdyż nie są spójne z całością i mogą powodować trudności w wykonaniu.

Inny zarzut wiąże się z brakiem symetrii spotkania, wynikającym z faktu, że zadanie wywoływane nie zna nazwy wywołującego, a więc nie zna źródła informacji. Ma to wiele konsekwencji, np. wywoływane zadanie nie może wybrać partnera spośród kilku zgłoszeń, musi polegać na dostarczanej podczas spotkania informacji identyfikującej, angażuje partnera aż do przygotowania odpowiedzi itp. Rozwiązanie takie — przede wszystkim ze względu na zawodność — jest niezadowalające. Poważne kłopoty może powodować także brak dostatecznych narzędzi do programowania upływu czasu w celu stwierdzenia gotowości (ang. time out). Choć zadanie wywoływane ma takie możliwości, to zadania wywołujące będą czekać nieskończenie długo przy braku odpowiedzi.

Wymienione, a także inne zastrzeżenia, np. niedostosowanie języka do wyrażania jawnej informacji o istnieniu zadań — mogą powodować znaczne trudności w szeregowaniu zadań przez użytkownika. Ponieważ optymalny sposób szeregowania zależy od zastosowania, programista powinien mieć tego rodzaju możliwości w ADZIE.

POZOSTAŁE ZARZUTY

Wielu zastrzeżeń, jakie zgłoszono do treści podręcznika (dotyczących niekonsekwencji, niekompletności, błędów, sprzeczności, niedokładności itp. [4, 7]) nie będziemy omawiać, ponieważ są łatwe do poprawienia i mają znaczenie drugorzędne. Choć nie świadczą one ujemnie o języku, a raczej o jego autorach, to mają niewątpliwy wpływ na zrozumienie podręcznika, a więc na realizację i użycie języka.

Omawiając zarzuty stawiane projektowi języka, a szczególnie — jego niedostosowanie do niektórych zagadnień, warto wspomnieć o próbie posługiwania się nim jako ję-

[1]) W praktyce istnieje wiele przykładów, że zależność taka występuje. Weźmy choćby słynną pomyłkę komputera podczas lotu amerykańskiej sondy kosmicznej na Wenus (opisaną np. w pracy [6], która kosztowała Amerykanów miliard dolarów. Błąd ten nigdy by nie wystąpił, gdyby program napisano w języku ADA, a nie w FORTRANIE.

zykiem do wydawania poleceń systemowych. W serii dokumentów formułujących wymagania dla środowiska programowego języka ADA, opracowanych przez amerykański Departament Obrony, zalecono wykorzystanie ADY jako języka poleceń (ang. job control language).

Praktyczna realizacja tego zalecenia napotyka jednak na szereg trudności lub jest — co najmniej — niewygodna [1]. Przykładowo — ADA wymaga statycznego deklarowania identyfikatorów przed użyciem, natomiast w zwykłych językach poleceń czas wiązania przesuwa się zazwyczaj do interpretacji poleceń. Jasne jest także, że pełna realizacja typów stało- i zmiennoprzecinkowych w takim języku jest zbędna. Składnia poleceń byłaby ponadto o tyle dziwna, że zawierałaby wywołania funkcji lub procedur z nawiasami i parametrami. Nie wiadomo również, czy wielozadaniowość w obecnej postaci jest najbardziej odpowiednia do wykorzystania w języku poleceń.

Istnieje wiele podobnych wątpliwości i jednoznacznie nasuwający się wniosek brzmi [1]: nie tylko ADA, lecz żaden inny język nie jest odpowiedni do tej roli. Dostosowanie go wymagałoby wprowadzenia tylu zmian i poprawek, że prawdopodobnie niewiele by zostało z pierwotnego kształtu języka. Wydaje się, że opisane trudności są skutkiem karkołomnego pomysłu dostosowania istniejącego języka do nowych wymagań, nie znanych jeszcze autorom podczas jego projektowania.

Ostateczne wymagania postawione projektowi języka, mające charakter specyfikacji (dokumenty IRONMAN, STEELMAN) poddał miażdżącej krytyce, nie pozbawionej złośliwości, Edsger Dijkstra [3]. Stwierdził on, że wymagania są bezsensowne, całkowicie dezawuują pracę projektantów języka i podał szereg przykładów dla poparcia tej tezy.

Wymagania zawarte w dokumencie IRONMAN, będącym podstawą do projektu, stanowią przede wszystkim jedynie opis cech, jakie powinien mieć język, a nie cele, jakim ma służyć. Twórcy języka musieli się zatem domyślać, na podstawie opisu właściwości, do jakich celów będzie on przeznaczony.

Specyfikacja języka, jak ją sformułowano, zawiera ponadto rażące sprzeczności, niejasności i braki, budzące podejrzenie, że została napisana przez amatorów. Ironizując na temat treści wymagań dotyczących instrukcji, Dijkstra pyta wprost, czy średniki i instrukcje **goto** są istotne do wzmocnienia wojskowej potęgi USA? Reasumując, według Dijkstry Departament Obrony nie powinien podpisywać kontraktów na zaprojektowanie języka mającego silne właściwości „typizacji", nie wiedząc czego i dlaczego żąda.

KONSEKWENCJE SPOŁECZNE

Lista ogólnych uwag dotyczących ADY na tym się nie kończy. Bardzo ważnym, lecz w ogóle nie uwzględnianym dotąd problemem są konsekwencje społeczne związane z wprowadzaniem języka [5]. Nie jest wcale pewne czy podstawowe założenie, na jakim oparto całe przedsięwzięcie Departamentu Obrony jest uzasadnione, a dokładniej — czy właściwie zidentyfikowano przyczyny wysokich kosztów utrzymania oprogramowania.

Znacznie więcej niejasności, niż mogłoby się wydawać, dotyczy skuteczności wprowadzenia języka ADA, a więc zrealizowania postawionych celów. Już obecnie nietrudno naszkicować szereg kwestii, które mogą mniej lub bardziej wpłynąć na powodzenie i opłacalność całego przedsięwzięcia, np.:

- czy dostarczone narzędzia programowe będą właściwie używane (tzn. czy niewłaściwe użycie nie spowoduje dodatkowych kosztów)
- czy użytkownicy korzystający z tych narzędzi nie staną się źródłem dodatkowych kosztów, być może przewyższających zyski spowodowane ich użyciem
- czy poziom użytkowników i wyposażenie laboratoriów będą odpowiadać zamierzeniom twórców języka
- czy praktyka przesuwania wielu prac programowych z fazy opracowania do fazy utrzymania programu jest na tyle powszechna wśród programistów, że może zaciążyć na rzeczywistym obrazie kosztów oprogramowania
- czy występująca dotąd wśród producentów tendencja do ścisłego przywiązania klientów różnymi sposobami do nabywania towaru będzie sprzyjać rozwojowi wymiennych systemów programowania w ADZIE
- czy koszt oprogramowania zestawów w innych językach nie wzrośnie przesadnie, gdy najzdolniejsi programiści będą masowo przesuwani do projektów związanych z ADĄ
- czy znaczenie innych czynników (tzw. lokalnych) nie wystarczy do wywarcia wpływu na szefów organizacji i przekonania ich, aby nie stosowali języka ADA.

Wszystkie wymienione wątpliwości stanowią mały ułamek trudnych do przewidzenia sytuacji społecznych, nie uwzględnionych w modelu cyklu życia oprogramowania, który posłużył do uzasadnienia celowości prac nad językiem ADA. Nie ma więc w zasadzie powodów, aby wierzyć, że dzięki omawianemu przedsięwzięciu te koszty ulegną zmniejszeniu.

* * *

Oczywiście, przedstawionego rozumowania nie można ze szczegółami przenosić do warunków polskich. Płynie stąd jednak z pewnością ważna dla nas nauka: rzeczywiste wyniki wykorzystania technologii zależą istotnie od społecznych warunków jej użycia. Dowodów na to dostarcza in extenso rozwój rodzimej informatyki, by nie sięgać do innych dziedzin. Natomiast z technicznego punktu widzenia zbyt wolna reakcja na prace nad językiem, mające od dawna zasięg międzynarodowy (ADA jest rozważana jako norma ISO), może zubożyć nie tylko nasz potencjał intelektualny, ale także myśl techniczną w dziedzinie informatyki.

LITERATURA

[1] Bender R. F.: The Case Against ADA as an APSE Command Language. SIGPLAN Notices, Vol. 15, No. 10, p. 27 (1980)
[2] Boute R. T.: Simplifying ADA by Removing Limitations. SIGPLAN Notices, Vol. 15, No. 2, p. 17 (1980)
[3] Dijkstra E. W.: DoD-1 — The Summing Up. SIGPLAN Notices, Vol. 13, No. 7, p. 21 (1978)
[4] Dijkstra E. W.: On the GREEN Programming Language Submitted to DoD. SIGPLAN Notices, Vol. 13, No. 10, p. 16 (1978)
[5] Kling R., Scacchi W.: The DoD Common High Order Programming Language Effort — What Will the Impact be? SIGPLAN Notices, Vol. 14, No. 2, p. 29 (1979)
[6] Myers G. L.: Projektowanie niezawodnego oprogramowania. WNT, Warszawa, 1980, s. 253
[7] Nicolescu R.: Some Short Comments on the Documentation of the ADA Programming Language. SIGPLAN Notices, Vol. 15, No. 7—8, p. 64 (1980)
[8] Raskin J.: An Apple for ADA. Electronic Design, Vol. 29, No. 1, p. 13 (8 January 1981)
[9] Van den Bos J.: Comments on ADA Process Communication. SIGPLAN Notices, Vol. 15, No. 6, p. 77 (1980)

Styl programowania w języku FORTRAN

Komitety normalizacyjne odegrały ważną rolę w rozwoju języka FORTRAN. Odpowiednie standardy zostały opracowane przez ANSI (1966), ISO (1972), SAA [1]) (1973) i powtórnie przez ANSI (1978). Pierwsze trzy standardy ograniczyły się w istocie do potwierdzenia istniejącej już praktyki, tj. ustalenia standardu na najwęższym wspólnym zakresie FORTRAN implementowanego na różnych komputerach. Ostatni standard ANSI stanowił już jednak duży krok naprzód — przyniósł bowiem specyfikacje umożliwiające wprowadzenie nowych cech do istniejących kompilatorów FORTRAN. Specyfikacje te zostały już uwzględnione w niektórych podręcznikach (np. Balfour, Marwick, 1979 [1]).

Drugą sprawą było opracowanie w ciągu ostatnich dziesięciu lat norm dotyczących przenośności. Wiele organizacji jest bardzo zainteresowanych przenośnością programów, które muszą być eksploatowane na różnych typach komputerów. Część z nich ma już wewnętrzne instrukcje, w których sprecyzowano szczegółowe wskazówki postępowania dla programistów w tego rodzaju sytuacjach.

W standaryzacji języka FORTRAN zbyt wiele uwagi poświęcono poszczególnym rozkazom. Programiście zostawia się natomiast nadmierną swobodę w dziedzinie ogólnej struktry programu. Dlatego też prace Kernighana i Plaugera [4, 5], które zajmują się tym zagadnieniem, winny stać się obowiązkową lekturą dla programistów w FORTRANIE; dają one bowiem proste, praktyczne rady jak tego języka używać. W porównaniu z projektowaniem obiektów inżynierskich, gdzie większość czasu zużywa się na studia opłacalności oraz projekty wstępne i techniczne, które pozwalają skrócić okres realizacji (pochłaniającej 90% kosztów), prace nad skomplikowanymi systemami oprogramowania prowadzi się metodami jakby wziętymi ze średniowiecza. Do programowania powinno się natomiast podchodzić z nastawieniem podobnym do tego, które towarzyszy działalności produkcyjnej.

PIĘĆ POZIOMÓW STYLU PROGRAMOWANIA

Programujący w FORTRANIE osiągają pięć wyraźnych stopni opanowania tej umiejętności. Postępy, jakie w niej czynią, zależą w dużej mierze od kolegów-programistów oraz od organizacji, w której są zatrudnieni. Średni czas opanowania jednego stopnia umiejętności wynosi jeden rok.

Stopień 1

Na poziomie pierwszym programista pisze programy długie i zawodne w działaniu z małą liczbą komentarzy, ze słabą dokumentacją, dające użytkownikowi niewielki wybór możliwości. Programista zwykle nie używa podprogramów ani napisanych przez siebie, ani bibliotecznych.

Stopień 2

Na poziomie drugim programista pisze już programy bardziej zwięzłe i pewniejsze w działaniu, które ponadto dają użytkownikowi większą liczbę wariantów postępowania. Programista stracił już ambicję samodzielnego pisania każdej instrukcji i w ograniczonym stopniu korzysta z procedur bibliotecznych. Zasadniczym postępem w stosunku do stopnia pierwszego jest stosowanie podprogramów. Programista dzieli zadanie na pewną liczbę oddzielnych, logicznych kroków. Typowy program wygląda następująco:

```
COMMON
CALL    CZYTAJ
CALL    ECHO
CALL    SPRAWDŹ
CALL    UKŁADAJ
CALL    ROZWIĄŻ
CALL    DRUKUJ
STOP
END
```

Taka metoda daje programy bardzo „usztywnione", ponieważ dane między podprogramami są przesyłane poprzez bloki COMMON. Oznacza to, że bloki programu — chociaż rozdzielone funkcjonalnie — w działaniu są nadal od siebie zależne. Nawet drobne zmiany w podprogramie wymagają zatem istotnych zmian w programie źródłowym. Z punktu widzenia wydajności programisty, programy tego typu są nieekonomiczne. Poszczególne podprogramy są bowiem tak ściśle związane z danym programem, że nie mogą być użyte powtórnie w innych programach bez wprowadzenia zmian. Zmiany te są zaś równoważne napisaniu nowego podprogramu.

Cechą charakterystyczną tego stopnia jest też uwzględnienie czasu (a więc kosztu) eksploatacji programu, z tym że dopiero po opracowaniu programu. Stosuje się tu dwa zabiegi: nakładkowanie i wielokrotne wykorzystanie pamięci w blokach COMMON. Żaden z tych zabiegów nie skraca jednak czasu działania jednostki centralnej, który zwykle kosztuje więcej niż czas pamięci. W eksploatacji znajduje się duża liczba programów napisanych w tym stylu. Wielu programistów nie czyni już dalszych postępów ponad ten stopień, ponieważ ich ogólny poziom wyszkolenia oraz zalecenia wewnętrzne instytucji, w których są zatrudnieni, utrwalają taki stan rzeczy.

Stopień 3

Na trzecim poziomie rozwoju programista przekonał się, że wiele programów zawiera znaczną część identycznych działań. Dlatego też pisze on podprogramy dla wykonywania tych działań, a także w coraz większym stopniu używa podprogramów bibliotecznych, np. z międzynarodowej biblioteki matematycznej i statystycznej (International Mathematical and Statistical Library, IMSL ISCO) [3] lub Grupy Algorytmów Numerycznych (Numerical Algorithms Group, NAG 1978) [7]. Do użycia tych podprogramów, programiście wystarczy określenie kilku parametrów. Parametry te przekazują odpowiednie dane do programu, natomiast bloki COMMON nie są używane lub używane rzadko. Jest to wydajna metoda programowania, ponieważ podprogram jest niezależny od bezpośredniego zastosowania i może być osobno dokładnie sprawdzony i przetestowany. Najważniejsze jest, aby lista parametrów była wystarczająco elastyczna, pozwalająca na zastosowanie jej w dostatecznie szerokiej grupie zagadnień.

Na rynku oprogramowania istnieje cały szereg pakietów podprogramów uwzględniających rozwiązania charakterystyczne dla stopnia trzeciego. Istnieją ponadto pakiety podprogramów do współpracy z urządzeniami do wyjść graficznych (CALCOMP, 1969; TEKTRONIX, 1974). W przypadkach, gdy FORTRAN jest używany do zadań, do których nie był przewidziany przez swych twórców, pakiet podprogramów jest jedyną racjonalną metodą rozszerzania tego języka na nowe dziedziny zastosowań. Przykładowo, FORTRAN ma rozkazy sterujące działaniem drukarki czy czytnika kart, ale nie ma takich rozkazów dla autokreślarek czy też dla pracy w czasie rzeczywistym.

Stopień 4

Stopień czwarty stanowi mały, ale decydujący postęp w stosunku do stopnia poprzedniego. W tym stadium programista w dalszym ciągu stosuje podprogramy wywoływane poprzez parametry, ale redukuje liczbę tych parametrów przez używanie małych bloków COMMON, za-

---

Niniejszy tekst opracowany został na podstawie referatu A. B. Millsa i R. W. Millsa „How good are your programs?", wygłoszonego w ramach Konferencji SOFTENG II w Londynie (patrz INFORMATYKA nr 9-10/81)

[1]) ANSI — American National Standards Institution; ISO — International Standards Organisation, SAA — Standards Association of Australia

wartych w pakiecie i dostępnych dla użytkownika. Przykładami takich zastosowań są pakiety graficzne GINO (1976) [2], DIGRAF (1979) [8], DISSPLA (1970).

Stosowanie pakietu podprogramów napisanych w sposób charakterystyczny dla stopnia czwartego jest łatwe i przyjemne, ponieważ programowanie jest zbliżone do programowania w języku naturalnym. Znaczenie każdego wywołania nie jest sztywne, lecz zmienia się zależnie od kontekstu.

W pakiecie przyjęto — co ważne — racjonalne, niejako wstępne założenia o sprawach, które nie są bezpośrednio przedstawiane. Metodę wprowadzania takich danych jest podprogram BLOCK DATA. Najlepsze z pakietów pozwalają użytkownikom na wywoływanie wymaganego zestawu podprogramów.

Stopień piąty zostaje osiągnięty wówczas, gdy programista napisał już szereg programów w stylu odpowiadającym stopniowi czwartemu i wyposażył je w pełną dokumentację. Stopień ten osiąga również wtedy, gdy mając w dyspozycji zakupione pakiety podprogramów (także w stylu stopnia czwartego) — jest w stanie skomponować programy szczególnie złożone, a jednocześnie dobrze funkcjonujące. Programy takie są bowiem zbudowane z wytestowanych, niezawodnych elementów. Sztuka inżynierska nie polega przecież na własnoręcznym wykonywaniu śrub i nakrętek, ale na umiejętności złożenia odpowiednich części — tak, aby służyły danemu zastosowaniu.

Podsumowując, podstawowe cechy poszczególnych stopni opanowania umiejętności programowania można ująć w sposób następujący:

Stopień	Cechy programowania
1 (początkujący)	programy rozwlekłe mało komentarzy mało podprogramów
2 (doświadczony)	wiele dużych bloków COMMON wiele podprogramów wywoływanie podprogramów bez parametrów dobrze zaprojektowane podprogramy uporządkowana kolejność numeracji rozkazów dość dobra organizacja programu
3 (dojrzały)	brak bloków COMMON podprogramy z długą listą parametrów wykorzystanie podprogramów bibliotecznych komentarze zorganizowane wg z góry ustalonej konwencji numeracja wersji programu procedury do wprowadzania poprawek
4 (ekspert)	niewielka liczba małych bloków COMMON podprogram BLOCK DATA pisane przez użytkowników biblioteki podprogramów użytkowych dobra dokumentacja dobra diagnostyka błędów współpraca pomiędzy podprogramami użytkowników
5 (uznany autorytet)	szerokie używanie obcego oprogramowania (bibliotecznego lub zakupionego) szeroka sprzedaż własnych programów

WPŁYW STYLU PROGRAMOWANIA NA ROZWÓJ PAKIETU PROGRAMÓW

Cecha ta jest zwykle nieznana użytkownikowi, ponieważ rzadko ma on wgląd w wersję źródłową pakietu. Styl programowania jest zaś — zdaniem autorów — kluczowym zagadnieniem przy tworzeniu oprogramowania dla zastosowań inżynierskich. Dobry styl to praktycznie gwarancja sukcesu w stosowaniu oprogramowania, dlatego też aspekt ten należy omówić nieco dokładniej. Autorzy programów nie zawsze mają wystarczające środki, aby spełnić wszystkie wymienione poniżej warunki dobrego pakietu. Wydaje się, że wiele z tych problemów można rozwiązać przez poprawę stylu programowania oraz zakup gotowych bibliotek podprogramów.

Gdy użytkownik decyduje się na stosowanie określonego programu, to — zanim zacznie oceniać same wyniki — zazwyczaj zapoznaje się z jego dokumentacją oraz wejściem i wyjściem. Jeśli którakolwiek z wymienionych trzech pozycji nie wzbudzi jego zaufania, to nie będzie miał on również zaufania do wyników programu.

Dokumentacja

Dobry pakiet programów ma zawsze **drukowaną** dokumentację, często w specjalnie zaprojektowanym skoroszycie. Istnieją następujące cztery rodzaje dokumentacji: reklamowa, podręcznik użytkownika, podręcznik operatora oraz dokumentacja systemu, umożliwiająca jego konserwację i rozwój. Każdy z podręczników winien mieć numer kolejnej wersji, spis treści oraz odsyłacze (wyjaśnienia). Testy gwarantujące dobrą jakość instalacji winny być włączone do opisu instalacji programu. Całość zaś powinna prezentować się w sposób budzący zaufanie.

Dokumentacja upraszcza się, gdy autor programu nie musi osobiście opisywać każdej jego części. Stosując firmowe podprogramy biblioteczne lub podprogramy indywidualnie zakupione, może on ograniczyć opracowanie dokumentacji tylko do części napisanych przez siebie — a więc zwykle do podprogramów inżynierskich.

Wejście

Podprogram wejścia winien mieć dobrą strukturę i logikę oraz odpowiednią elastyczność stosowania. W zastosowaniach tradycyjnych dane do programów inżynierskich wczytuje się z klawiatury lub z czytnika kart. Dobry pakiet winien mieć format swobodny oraz pozwalać na zmienną kolejność czytania i zmianę znaczenia (zależnie od kontekstu). Powinien też zawierać możliwość stosowania kilku różnych sposobów wejścia (np. ANSYS, 1978).

Tradycyjne wejście w języku FORTRAN ma stały format i niezmienną kolejność, bez sygnalizacji błędów. Znając możliwości wejść graficznych, użytkownicy są coraz mniej skłonni przyjmować tego rodzaju ograniczenia, charakterystyczne dla metod tradycyjnych.

Autor programu powinien nabyć dobry pakiet uniwersalnych procedur wejścia, a także podobny paiket procedur sterowania ekranem. Pozwoli to użytkownikom na dowolne formatowanie ekranu oraz stosowanie łatwych do nauczenia się i użycia procedur wejścia.

Wyjście

Wyjście z programu ma decydujące znaczenie dla użytkownika, dlatego też dobre pakiety powinny zapewniać możliwość działania w trybie konwersacyjnym oraz wybór wyników. Tak jak w przypadku wejścia, użytkownik spodziewa się tu ze strony systemu znacznej pomocy i współpracy. Prostym testem wejścia jest zbadanie skutków decyzji, gdy w celu sfotografowania lub wykonania kserokopii strony wydruków zostaną rozdzielone. Jeśli bowiem nie można zidentyfikować każdej strony w sposób jednoznaczny, to wyniki budzą zastrzeżenia.

Dobrze zaprojektowane wyjście zawiera numer strony, a na każdej stronie nazwę użytkownika, nazwę programu, datę i czas realizacji. Ważną, choć często niedocenianą, pozycją na wyjściu są komunikaty zawierające ostrzeżenia i informacje o błędach. Powinny być one dobrze objaśnione w dokumentacji. Duże systemy mają specjalne podręczniki, w których każdy komunikat jest objaśniony w języku zrozumiałym dla użytkownika.

Na wyjściu należy również użyć zakupionego pakietu uniwersalnych procedur. Istnieje szereg takich pakietów (np. Mills [6]), chociaż należy podkreślić, że FORTRAN ma dostatecznie rozwinięte możliwości wyjścia. Dla celów wyjścia graficznego takie pakiety, jak GINO [2] czy DIGRAF [8] zapewniają znaczne możliwości zaspokojenia wymagań wielu użytkowników.

Poradnictwo i konserwacja

Dobry pakiet programów ma ustalony tryb poradnictwa. Funkcje doradcze pełni zwykle osoba przeszkolona u użytkownika, niemniej kontakt z autorami pakietu powinien być utrzymany.

Dobrą formą współpracy jest rozsyłany użytkownikom biuletyn zawierający uzupełnienia i poprawki. W przypadku dużych systemów powstają kluby użytkowników, a jednostki autorskie prowadzą odpowiednie kursy szkoleniowe. W regularnych odstępach czasu ukazują się nowe wersje pakietu. Jeśli autor programu zastosował np. gotowy pakiet procedur wejścia lub wyjścia graficznego, to nie musi się on troszczyć o konserwację i konsultacje tej części programu. Co więcej — jeśli zostanie wprowadzona nowa wersja takiego pakietu, to automatycznie jego program został unowocześniony bez napisania nawet jednego wiersza. Program taki stanie się bardziej niezawodny, pozwalając jego autorowi na zajęcie się szkoleniem użytkowników i przygotowaniem niezbędnych biuletynów informacyjnych, zwalniając go od przykrej funkcji występowania w charakterze straży pożarnej.

Technologia

Trudności w programowaniu powstają głównie w tych obszarach, w których programista ma najmniejsze doświadczenie. W przypadku inżynierów, trudności takie wynikają raczej w indywidualnie wykonanym dekoderze swobodnego formatu, niż w podprogramach rozwiązujących konkretne zagadnienie. Wynika to z faktu, że wiadomości programisty-inżyniera na temat swobodnego formatu są zazwyczaj znikome. Programista winien więc unikać kodowania tam, gdzie istnieją niezawodne pakiety procedur i skoncentrować cały wysiłek na części problemowej zadania.

W przypadku, gdy program lub pakiet programów jest na rynku unikalny, tzn. że żaden inny program nie rozwiązuje podobnego zadania, to można wówczas wybaczyć wiele jego słabych punktów w zakresie technologii (np. słaba dokumentacja, niewygodne wejście). Taki pakiet jest jednak na rynku produktem bardzo wrażliwym. Jeśli bowiem firma konkurencyjna zaoferuje pakiet rozwiązujący to samo zadanie, lecz bez wspomnianych słabych punktów w zakresie technologii, wówczas użytkownik zmieni pakiet dotychczas używany na produkt firmy konkurencyjnej.

Koszty

Cena pakietu programów musi być opłacalna zarówno dla ośrodka autorskiego, jak i dla klienta. Oprogramowanie w dziedzinie zastosowań inżynierskich było dotychczas sprzedawane po wysokich cenach stosunkowo niewielkiej liczbie klientów. Ten stan rzeczy z pewnością się zmieni. Niektóre firmy — na przykład — oferują obecnie pakiet programów graficznych na „osobistym" mikrokomputerze za ok. 1/20 ceny analogicznego pakietu na dużym komputerze.

Koszt opracowania i konserwacji oprogramowania jest wprost proporcjonalny do czasu jaki zużywa na te czynności programista. Chociaż zakup gotowego oprogramowania, celem włączenia go do opracowywanych programów, wymaga często znacznych wydatków, to jednak nakłady takie prawie zawsze są bardzo opłacalne. Dlatego ich unikanie jest błędnie pojętą oszczędnością. Przy przenoszeniu na inny komputer lub inne urządzenie wyjścia, wiele pakietów handlowych ma również odpowiednie wersje alternatywne, co powoduje, że programu nie trzeba przerabiać. Przykładowo wspomniane już pakiety graficzne GINO i DIGRAF są całkowicie niezależne od urządzenia końcowego.

LITERATURA

[1] Balfour A., Marwick D. H.: Programming in FORTRAN 77. Heinemann Educational Books, London 1977
[2] GINO-F User's Manual. Computer Aided Design Centre, Cambridge, 1976
[3] International Mathematical and Statistical Library. IMSL Houston, Texas, 1980
[4] Kernighan B. P., Plauger P. J.: The elements of programming style. Mc Graw Hill, 1974
[5] Kernighan B. P., Plauger P. J.: Software Tools. Prentice Hall 1976
[6] Mills A. B.: Calcomp Emulator Package for the Line Printer. United Computing, London, 1980
[7] Numerical Algorithmus Group. NAG Library Manual, Oxford, 1978
[8] Warner J.: Device Independent Graphics in FORTRAN. DIGRAF User's Manual. University of Colorado, Boulder, 1979.

Oprac. MACIEJ S. WINIARSKI

HANNA STARZYK

Instytut Organizacji Zarządzania i Doskonalenia Kadr
Zakład Informatyki
Warszawa

OSIRIS III w badaniach statystycznych

OSIRIS III [1] jest pakietem programów przeznaczonych do obliczeń statystycznych. Opracowany został w USA w latach 1967—1976 przez Institute for Social Research, The University of Michigan. W naszym kraju stosowany jest dosyć szeroko w badaniach społecznych (a także medycznych). Artykuł poświęcony jest omówieniu niektórych jego zastosowań.

CHARAKTERYSTYKA PAKIETU

Pakiet składa się z 60 programów, które zapewniają:
— organizację danych, edycję i korektę błędów
— drukowanie danych
— kopiowanie zbioru lub wybranej jego części
— generowanie kart dla programu IBM SORT/MERGE

— transformacje zmiennych za pomocą operacji arytmetycznych i logicznych

— generowanie zestawień statystycznych, statystyk, testów, analizę regresji, wariancji itd.

Bogate oprogramowanie w zakresie obliczeń statystycznych oraz wszechstronne możliwości w zakresie edycji, manipulacji i korygowania danych pozwalają na zredukowanie do minimum, a nawet całkowite wyeliminowanie potrzeby opracowywania dodatkowego oprogramowania. Z drugiej strony — użytkownik może pracować na zbiorze danych OSIRIS-a za pomocą własnego oprogramowania, w przypadku gdy OSIRIS nie może zaspokoić specyficznych wymagań takiego użytkownika. Pakiet może być eksploatowany na komputerach serii IBM 360 i 370, których konfiguracje obejmują:

- 110 KB pamięci operacyjnej dla programów użytkownika
- 1 czytnik + perforator kart
- 1 drukarkę wierszową
- 1 jednostkę sterującą pamięci dyskowej
- 4 jednostki pamięci dyskowej
- 2 jednostki pamięci taśmowej.

Dotyczy to także komputerów JS, począwszy od R-20, wyposażonych w jednostki pamięci dyskowej EC 5052 oraz jednostki pamięci taśmowej EC 5517.

SPOSÓB POSŁUGIWANIA SIĘ PAKIETEM

Każda ankieta statystyczna zapamiętywana jest jako kolejny rekord zbioru. Do tego tworzy się słownik, który zawiera opis każdej zmiennej w ankiecie. Słownik zawiera więc unikalny numer zmiennej, adres, typ, nazwę itp. Generowanie słownika odbywa się przy użyciu programu OSIRIS-a — FBUILD. Te dwa zbiory — zbiór danych oraz opisujący go słownik — są podstawowymi zbiorami OSIRIS-a. Często w trakcie przetwarzania zbioru ankiet zachodzi potrzeba wyodrębnienia pewnych podzbiorów danych do dalszego przetwarzania. Każdy z nich musi posiadać odpowiadający mu słownik, który może być utworzony przez użytkownika w osobnym trybie (programem FBUILD), bądź jest generowany automatycznie przez program wybierający określony podzbiór danych.

Prześledźmy na przykładzie uruchomienie wybranego programu OSIRIS-a:

```
a) || EXEC OSIRIS
b) || DICTIN DD DSN = SLO, UNIT = 3330, VOL = SER
      = WOJTEK, DISP = OLD
c) || DATAIN DD DSN = DATA, UNIT = 3330, VOL =
      = SER = WOJTEK, DISP = OLD
d) || SETUP DD *
e) $ RUN MDC
f) $ RECODE
            R1 = 80—V10
            IF V11 EQ O THEN R2 = 1 ELSE R2 =
            = V11
            R3 = V15/R2
   $ SETUP
g)          EXCLUDE V5 = 1,3 *
h) MACIERZ KORELACJI DLA ZMIENNYCH GRUPY 1
   W POPULACJI A
i) BADDATA = SKIP *
j) R1, R3, V16—V20 *
```

Omówimy pokrótce kolejne karty tego programu:
a) powoduje wywołanie procedury głównej, inicjuje pracę
b) i c) opisują zbiory wejściowe — słownik SLO oraz zbiór danych DATA

d) karta kontrolna, po której może wystąpić szereg sekwencji e-j; w jednym przebiegu istnieje więc możliwość uruchomienia wielu programów OSIRIS-a

e) wywołanie konkretnego programu OSIRIS-a, w tym przypadku — programu MDC, liczącego macierz współczynników korelacji Pearson'a

f) zdania RECODE (fakultatywne) pozwalają na modyfikację istniejących zmiennych lub tworzenie nowych przy użyciu operacji i funkcji arytmetycznych i logicznych; nowe zmienne mogą być tworzone albo tylko na czas przebiegu programu (jak w powyższym przykładzie), bądź też zapisywane na stałe w zbiorze danych na nośniku fizycznym — co w przypadku wielokrotnego przetwarzania jest bardziej praktyczne; proces ten obsługują specjalne programy OSIRIS-a

g) zdanie to jest tzw. filtrem, służącym do wyodrębnienia z całej populacji podzbioru ankiet, które spełniają określone warunki; w omawianym przykładzie z obliczeń zostaną wyłączone te ankiety, w których zmienna V5 ma wartość 1 lub 3; filtr jest zdaniem fakultatywnym — jego brak powoduje przetwarzanie całego zbioru danych
h) na karcie tej użytkownik umieszcza własny komentarz dotyczący danego przebiegu

i) w zależności od potrzeb umieszczona jest lista parametrów wymaganych przez ten program; w tym przypadku parametr BADDATA = SKIP powoduje „przeskoczenie" błędnej ankiety w przypadku wystąpienia błędu w danych
j) karta zawiera listę zmiennych, które będą przetwarzane; w tym przypadku są to zmienne o numerach od 16 do 20 oraz zmienne utworzone ze zmiennych źródłowych — R1 i R3.

PRZETWARZANIE ODPOWIEDZI NA PYTANIA OTWARTE

Pewne problemy przy przetwarzaniu mogą sprawiać zmienne, które zawierają odpowiedzi na tzw. pytania otwarte, przykładowo — gdy ankieta zawiera szereg pytań dotyczących znajomości języków obcych:

Pola (kolumny) ankiety:

```
30 |_|
31 |_|
32 |_|
33 |_|
34 |_|
35 |_|
36 |_|
```

kody języków:
1 — angielski, 2 — niemiecki, 3 — włoski, 4 — francuski, 5 — rosyjski, 6 — węgierski, 7 — esperanto.

Respondent może wypełnić różną liczbę pól różnymi kodami języka, z których każdy może wystąpić na dowolnej pozycji. Zmienną taką (ang. *multiple response*) można opisać w słowniku podając liczbę możliwych odpowiedzi. Występuje tu jednak szereg poważnych ograniczeń podczas przetwarzania. Jeśli taka zmienna wystąpi w zdaniach RECODE lub w filtrze, wówczas wzięta będzie do obliczeń tylko pierwsza wartość — w tym przypadku z kolumny 30. Gdybyśmy chcieli np. wyodrębnić z całej populacji osoby znające esperanto i użyli w tym celu zdania INCLUDE V5 = 7*, wówczas wyodrębniony podzbiór składałby się wyłącznie z tych ankiet, w których kod esperanto (7) wystąpił jako pierwszy. Jest to oczywicie dalece niewystarczające. Trudność tę można usunąć, tworząc nowy słownik, w którym zamiast jednej zmiennej (znajomość języków obcych) — użyjemy siedmiu odrębnych zmiennych (V5—V11). Wybrania grupy osób znających esperanto można wówczas dokonać pisząc zdanie:

```
INCLUDE V5—7 OR V6 = 7 OR V7 = 7 OR V8 = 7 OR
V9 = 7 OR V10 = 7 OR V11 = 7*
```

W sytuacji, gdy w ankiecie występuje zmienna (lub zmienne) tego typu, to — aby zapewnić najbardziej efektywne przetwarzanie — wygodnie jest tworzyć dwa słowniki.

PRZYKŁAD ZASTOSOWANIA

Pakiet OSIRIS jest eksploatowany w Instytucie Organizacji Zarządzania i Doskonalenia Kadr od 1977 r. Od tego czasu zostało wykonanych wiele badań ankietowych, dotyczących oceny kadry kierowniczej, sposobu zarządzania oraz różnych problemów z zakresu nauk społeczno-politycznych.

W 1978 r. podjęto badanie spożycia leków w Polsce. W pierwszym okresie badaniem objęto 1500 przychodni zdrowia w całym kraju. W następnym etapie analizie poddano około 11 000 dokumentów z przychodni zdrowia oraz 6000

ze szpitali z różnych regionów kraju. Każda ankieta zawierała dane personalne pacjenta (wiek, płeć, identyfikator, źródło utrzymania) oraz dane diagnostyczne (nr statystyczny choroby lub chorób towarzyszących, kody stosowanych leków, ich dawkowanie), a także koszt leczenia. Ze względu na dużą liczbę dokumentów źródłowych oraz ich sukcesywne dostarczanie, etap przygotowania danych do analizy był stosunkowo długi. Wstępnej kontroli formalnej dokonano programami napisanymi w języku PL/I. W momencie dostarczenia ostatniej partii dokumentów zbiór danych był już poprawny z dużą dozą prawdopodobieństwa. Sformułowane przez użytkownika warunki logiczne posłużyły do dalszej kontroli danych w systemie OSIRIS. Przykładowo — program wykrywający ankiety, w których pacjentowi-mężczyźnie została wpisana błędna choroba kobieca, przedstawia się następująco:

```
//EXEC OSIRIS
//DATAIN DD DSN = LEKI. DATA, UNIT = 3330, VOL =
= SFR = WOJTEK, DISP = OLD
//DICTIN DD DSN = LEKI. DICT, UNIT = 3330, VOL =
= SFR = WOJTEK, DISP = OLD
//SETUP DD *
$RUN WCC (wywołanie programu sprawdzającego)
      INCLUDE V7 = 1 * (zmienna V7 — płeć, wybranie
      populacji mężczyzn)
CHOROBY KOBIECE U MĘŻCZYZN
ID = (V2,V3,V4) * (numery zmiennych identyfikujących
   błędną ankietę, jakie pojawią się w wydruku)
V9 = 610-678,256,180-184,217-221,133-136 * (zmienna V9—
   rozpoznanie chorobowe nie może przybierać wymienionych kodów)
```

Wykryte w ten sposób błędy, po ich konfrontacji z treścią dokumentu źródłowego, korygowane były programem OSIRIS-a TCOR, który pozwala na korygowanie błędów zarówno w zbiorze danych, jak i w słowniku. Charakterystyczną cechą tej ankiety był fakt, że w szpitalu liczba zapisanych pacjentowi leków mogła dochodzić nawet do 22.

Zmienna, którą intuicyjnie chcielibyśmy nazwać „lek zapisany pacjentowi" może zawierać od 1 do 22 możliwych odpowiedzi, tzn. kodów leków z urzędowego spisu leków, zawierającego ok. 1000 pozycji. Wystąpiła tu więc sytuacja analogiczna do opisanej we wcześniej podanym przykładzie. Został utworzony słownik, w którym zmienną LEK opisano jako typ „*multiple response*". Programy dokonujące analizy ilościowej pozwoliły na efektywne przetwarzanie tej zmiennej. Program generujący tablice jedno- lub dwuwymiarowe, które zawierają rozkłady liczebności jednej cechy lub rozkłady warunkowe dla dwóch zmiennych — może uwzględnić wszystkie występujące warianty odpowiedzi:

```
$RUN TABLES
ROZKŁAD LICZEBNOŚCI ZMIENNEJ LEK
*
TABLES
UNIVAR R = 0 % = TOTAL MR* (parametr MR — żąda
   uwzględnienia wszystkich możliwych odpowiedzi)
V7*
```

Taki sposób organizacji danych okazał się jednak niewystarczający dla przetwarzania tych zdań, w których kod leku musiał pojawić się w filtrze, a więc wówczas, gdy należało wybrać do obliczeń populację leczoną określonym rodzajem leku lub grupą leków. Problem nie mógł być rozwiązany za pomocą zdefiniowania 22 oddzielnych zmiennych jak w cytowanym wcześniej przykładzie (znajomość języków obcych) z uwagi na ograniczenia formalne narzucone na zdanie filtru. Przewidziano bowiem maksymalnie trzy karty na filtr, co przy konieczności wymienienia 22 zmiennych jest niewystarczające. Dlatego też oprócz podstawowego zbioru danych, który w każdym rekordzie zawierał od 1 do 22 kodów leków, został utworzony nowy zbiór, w którym rekord wejściowy zawiera wszystkie informacje dotyczące pacjenta i tylko jeden lek. Dla każdej ankiety powstało więc tyle rekordów, ile leków miał zapisanych pacjent. Na zbiorze tym założono słownik, w którym jest już tylko jedna zmienna dotycząca leku. Słownik ten wraz z odpowiadającym mu zbiorem danych był używany dla tych obliczeń, w których kod leku był filtrem.

Np. sekwencja dotycząca pytania — „W jakich rozpoznaniach stosowano leki recepturowe (kod leku recepturowego — 2000)" wygląda następująco:

```
$RUN TABLES
          INCLUDE V21 = 2000*
LEKI RECEPTUROWE W ROZPOZNANIACH GŁOWNYCH
PRINT = NODICT*
TABLES
UNIV R 0 % = TOTAL*.
V9*
```

Rozpoznanie główne oraz rozpoznania dodatkowe (towarzyszące) powinny być w pewnych przypadkach traktowane jako jedna zmienna. Stąd wyłoniła się konieczność założenia słownika, w którym również zmienna ROZPOZNANIE jest typu „*multiple response*". W tym przypadku powyższe pytanie (rozszerzone o rozpoznania dodatkowe) jest realizowane przez przebieg:

```
$RUN TABLES
          INCLUDE V21 = 2000*
LEKI RECEPTUROWE W ROZPOZNANIACH
PRINT = NODICT*
TABLES
UNIV R 0 % = TOTAL MR*
V9*
```

Często jednak uzyskanie oczekiwanej odpowiedzi wymagało tworzenia całych sekwencji programów realizujących kolejno takie funkcje, jak: wybieranie określonego podzbioru, sortowanie, łączenie zbiorów, dopisywanie zmiennej itp. Pokaźna grupa pytań dotyczyła interakcji lekowych (leki, których pacjent nie powinien zażywać jednocześnie), bądź przypadków, w których stosowanie leków z pewnej grupy powinno być połączone z podawaniem leków z innej grupy. Rozwiązanie tego wymagało wyodrębnienia wybranych populacji chorych (program SUBSET), posortowania (program SORMER) oraz zbadania części wspólnej (program MMP). W sumie przetworzono kilkaset pytań dotyczących prawidłowości terapii, analizy kosztów leczenia, zużycia leków itp., zarówno w skali kraju, jak i w obrębie poszczególnych regionów i województw.

OSIRIS III okazał się pakietem bardzo elastycznym, pozwalającym na efektywne jego stosowanie, nawet w tak mało sformalizowanych dziedzinach, jak badania społeczne, medycyna czy farmakoterapia.

LITERATURA

[1] OSIRIS III. Institute for Social Research. The University of Michigan Ann Arbor, Michigan USA, Volume 1—6.

KAZIMIERZ WIECZORKOWSKI
Uniwersytet im. Mikołaja Kopernika
Toruń

REKRUTACJA
– system kwalifikowania kandydatów na studia

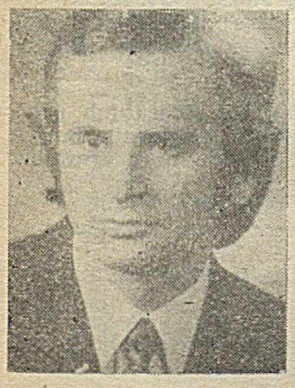
Przeprowadzenie rekrutacji kandydatów na studia jest dużym przedsięwzięciem organizacyjnym. Można powiedzieć, że jest to największa „akcja" szkoły wyższej. Angażuje ona dużą liczbę pracowników naukowych, technicznych i administracyjnych (nierzadko 10—30% stanu osobowego); dotyczy zwykle wszystkich jednostek dydaktycznych szkoły; wreszcie — charakteryzuje się dużą intensywnością i różnorodnością prac w krótkim okresie czasu. Wynikami egzaminów zainteresowane są również organizacje społeczno-polityczne, władze oświatowe regionu, a w zakresie sprawozdawczości także urzędy centralne. Szkoły średnie oczekują analiz wyników egzaminów dotyczących ich kandydatów.

W przebiegu egzaminów wstępnych wyróżnić można dwa zasadnicze aspekty: merytoryczny (samo egzaminowanie) oraz administracyjny (ewidencja kandydatów, rejestracja wyników egzaminu, kwalifikacja zgodna ze stosowanymi kryteriami, przydzielanie miejsc noclegowych, harmonogramowanie egzaminów, analizy wyników i sprawozdawczości, przydział pomocy materialnej itp.). Dla przeprowadzenia naboru kandydatów na studia rektor powołuje komisję uczelnianą oraz komisje wydziałowe (lub kierunkowe), w skład których wchodzą przeważnie pracownicy naukowo-dydaktyczni. Stroną administracyjną zajmują się sekretarze tych komisji; oni również prowadzą z reguły punkt konsultacyjny. Większość ich prac to czynności proste i powtarzające się. W zakresie tych prac wymagana jest jednak wiarygodność informacji, bezstronność i dyscyplina w podejmowaniu decyzji.

Kwalifikacja kandydatów przeprowadzana jest przy zastosowaniu systemu punktowego, uwzględniającego wyniki egzaminów oraz dodatkowe kryteria kwalifikacyjne (np. pochodzenie, staż pracy, służba wojskowa, oceny w szkole średniej). Przyjmowani są również laureaci olimpiad przedmiotowych, Turnieju Młodych Mistrzów Techniki itp. Stosowane są różne skale ocen i punktacji za egzaminy kierunkowe i pozostałe.

ZAKRES FUNKCJONALNY SYSTEMU

Zaprojektowany system informatyczny — REKRUTACJA przeznaczony jest dla szkół uniwersyteckich, technicznych, rolniczych, pedagogicznych i ekonomicznych. Celem systemu jest gromadzenie, przetwarzanie i udostępnianie informacji o kandydatach i przebiegu egzaminów.



Mgr inż. KAZIMIERZ WIECZORKOWSKI ukończył w 1974 r. studia na Wydziale Elektroniki Politechniki Gdańskiej. Do roku 1976 pracował w Pracowni Metod Numerycznych Instytutu Maszyn Matematycznych Oddział w Toruniu. Obecnie jest kierownikiem Działu Oprogramowania w Ogólnouczelnianym Ośrodku Obliczeniowym Uniwersytetu Mikołaja Kopernika. Zajmuje się projektowaniem i oprogramowaniem systemów do zarządzania i informacji naukowej.

System ma zadanie ułatwić komisjom rekrutacyjnym, administracji szkoły, organizacjom społeczno-politycznym, szkołom średnim i kuratoriom oświaty i wychowania żmudne prace rutynowe (wyszukiwanie informacji z dokumentów wg różnych kryteriów, przygotowywanie list kandydatów i zawiadomień, wykonywanie analiz wyników, sprawozdań itp.).

Podstawowe funkcje realizowane przez system to:

- ewidencja kandydatów
- przydział miejsc noclegowych na czas egzaminów
- przydzielenie kandydatom miejsc na czas pierwszego egzaminu
- rejestracja wyników i decyzji komisji
- emisja list i protokołów
- analiza wyników i sprawozdawczość
- przyznawanie pomocy materialnej dla studentów I roku.

STRUKTURA SYSTEMU

Bazę informacyjną systemu tworzy siedem zbiorów podstawowych umieszczonych w pamięci dyskowej oraz jeden zbiór roboczy. Informacje są dostarczane do systemu za pomocą 13 dokumentów, przy czym trzy z nich (protokoły) są przygotowywane jako dokumenty wynikowe (na maszynie cyfrowej) w postaci częściowo wypełnionych formularzy (rys. 1).

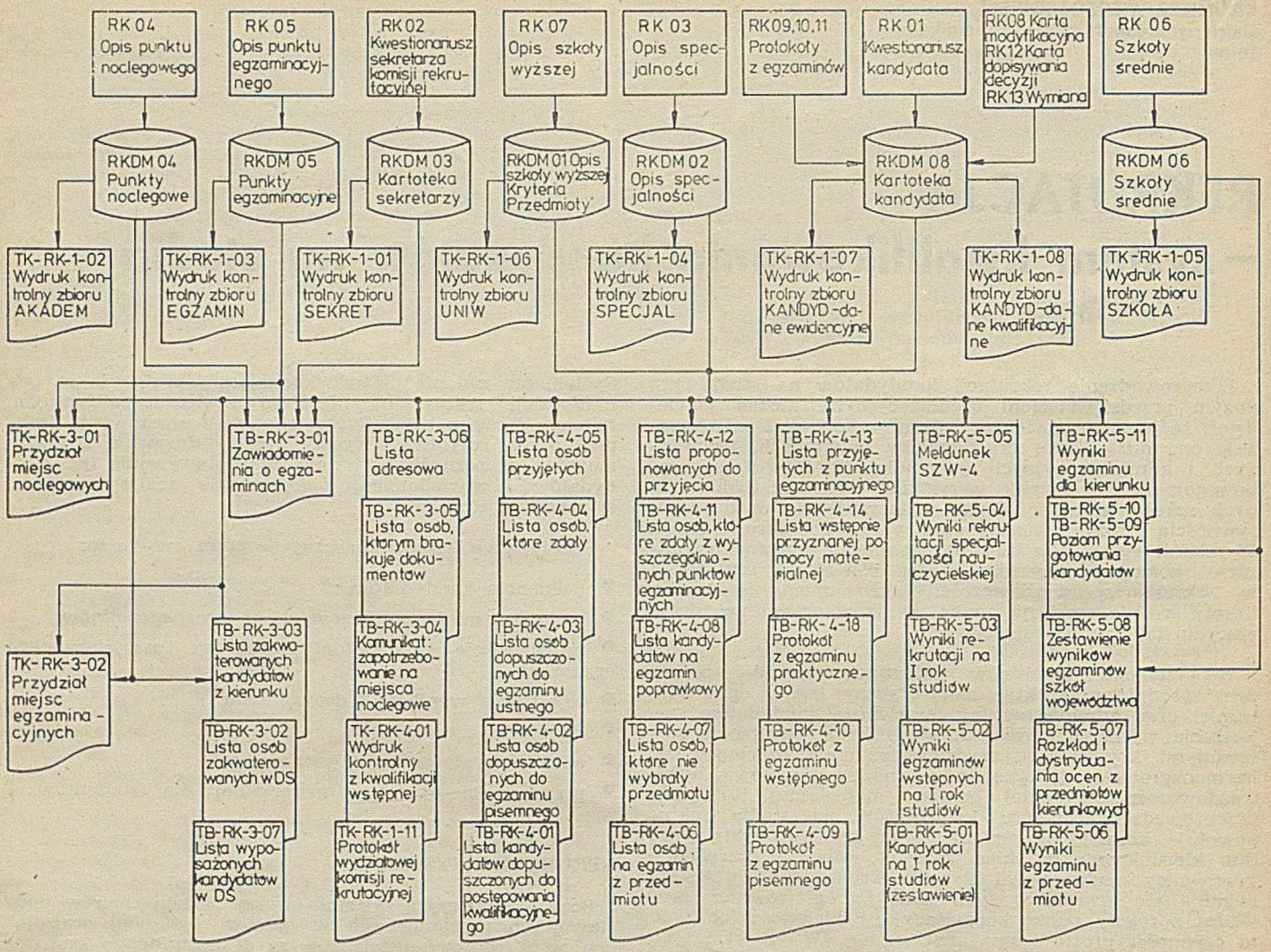
System składa się z 50 procedur, które tworzą 9 zadań. Programy zakładania wszystkich zbiorów oraz aktualizacji kartoteki kandydatów stanowią odrębne zadania; natomiast wszystkie pozostałe funkcje systemu realizowane są w jednym programie o strukturze nakładkowej, dzięki czemu maksymalna wymagana pamięć operacyjna wynosi 140 K bajtów. Dla ułatwienia operowania tym programem ustalono skrócone nazwy funkcji (hasła), które stanowią klucze dostępu do określonych procedur wykonujących te funkcje. Wszystkie dane wejściowe czytane w tych procedurach mają ujednoliconą postać i są kontrolowane przez jedną procedurę. W systemie przewidziano 13 typów tabulogramów kontrolnych oraz 37 zestawień wynikowych.

Dokumenty systemowe

RK01 — Kwestionariusz kandydata na studia. Zawiera dane ewidencyjne kandydata. Służy do wpisywania danych do kartoteki kandydatów.

RK02 — Kwestionariusz sekretarza kierunkowej komisji rekrutacyjnej. Zawiera dane sekretarza, adres prowadzonego przezeń punktu konsultacyjnego, adres dziekanatu oraz limity miejsc w punktach noclegowych dla danego kierunku.

RK03 — Opis specjalności. Obejmuje takie dane, jak: nazwy wydziałów, kierunków, specjalności studiów, limity przyjęć, wartości barier średniej ocen kierunkowych (powyżej której kandydat powinien zostać przyjęty na studia) oraz określenie przedmiotów egzaminacyjnych z podaniem typu egzaminu (w tym informacja czy jest to egzamin kierunkowy) i punktacji.



Rys. 1. Schemat funkcjonalny systemu REKRUTACJA

RK04 — Opis punktów noclegowych. Zawiera dane o adresie i typie, liczbie miejsc, terminie rozpoczęcia kwaterowania i godzinach kwaterowania w poszczególnych punktach.

RK05 — Opis punktów egzaminacyjnych. Opisuje punkty egzaminacyjne dla specjalności. Zawiera takie dane, jak: adres miejsca pierwszego egzaminu, termin rozpoczęcia, kod przedmiotu oraz pojemność sali.

RK06 — Szkoły średnie. Jest dokumentem zakładającym indeks szkół.

RK07 — Opis szkoły wyższej. Jest dokumentem kilkuczęściowym, zawierającym: dane o szkole wyższej oraz o rekrutacji, kryteria ocen i punktacji, opis kwestionariusza kandydata oraz nazwy przedmiotów egzaminacyjnych. Zawiera między innymi takie dane, jak: nazwa szkoły, liczebność wydziałów i kierunków, nazwy kryteriów punktów dodatkowych, dane dla przyznawania stypendium, nazwy dokumentów, dane dla odwołań, datę rozpoczęcia roku akademickiego.

RK08 — Karta modyfikacyjna. Służy do modyfikowania informacji z kwestionariusza kandydata w jego rekordzie. Możliwe jest też indywidualne dopisywanie ocen i decyzji komisji z egzaminów, jednak nie zleca się tego, ponieważ oceny winny być wprowadzane w zasadzie bezpośrednio z protokołów egzaminacyjnych.

RK09 — Protokół z egzaminu praktycznego. Formularz protokołu przygotowany jest jako dokument wynikowy na drukarce — dla kierunków, które zdają egzamin praktyczny. Jest on częściowo wypełniony, tzn. wydrukowane są już identyfikatory i nazwiska kandydatów. Stanowi jednocześnie dokument źródłowy dla wprowadzenia ocen z egzaminu.

RK10 — Protokół z egzaminu pisemnego. Stanowi dokument źródłowy przygotowywany przez system jako dokument wynikowy z wpisanymi nazwiskami i identyfikatorami kandydatów i z pozostawionymi wolnymi kratkami na wpisanie ocen. Przygotowywany jest dla danego kierunku i egzaminu.

RK11 — Protokół z egzaminu wstępnego. Jest również częściowo wypełnionym dokumentem wynikowym, stosowanym do wprowadzenia ocen ostatecznych do rekordu kandydata.

RK12 — Karta dopisywania decyzji o przyjęciu na studia. Stanowi dokument aktualizacyjny, służący do zmian decyzji przyjętych podczas automatycznej kwalifikacji oraz do dopisywania decyzji komisji rekrutacyjnych różnych szczebli.

RK13 — Karta usuwania kandydata ze zbioru lub przenoszenia na inny kierunek. Służy do przepisywania danych kandydata w przypadku zmiany kierunku oraz usuwania jego danych z ewidencji.

Dokumenty wynikowe

System pozwala na uzyskanie m.in. następujących typów dokumentów wynikowych:

- wydruków kontrolnych dotyczących zakładania i modyfikowania zbiorów podstawowych

- wydruków kontrolnych dotyczących rozdziału miejsc noclegowych

- zapotrzebowania na miejsca noclegowe dla kierunku, wydziału, uczelni

- zawiadomienia o rozpoczęciu egzaminów ze wskazaniem brakujących dokumentów; zawiadomienie drukowane jest wraz z adresem w formie nadającej się do wysłania pocztą; zawiera informację o przyznanym miejscu noclegowym i stanowi jednocześnie kartę mieszkańca

- list kandydatów: dopuszczonych do postępowania kwalifikacyjnego; zakwaterowanych w punktach noclegowych dla określonego kierunku; zakwaterowanych w określonym punkcie noclegowym, u których stwierdzono brak dokumentów; dotyczącej wyposażenia kandydatów w punkcie noclegowym; dopuszczonych do egzaminów pisemnych; na egzamin pisemny z danego przedmiotu; zdających w punkcie egzaminacyjnym; dopuszczonych do egzaminów ustnych; na egzamin ustny z danego przedmiotu; na egzamin poprawkowy z danego przedmiotu; którzy zdali egzamin wstępny; którzy zdali egzamin wstępny z wyszczególnieniem punktów za oceny i punktów dodatkowych; proponowanych do przyjęcia; przyjętych na I rok studiów; ze wstępnie przyznaną pomocą materialną; którym przyznano stypendium na I rok studiów;

- formularzy protokołów z egzaminu praktycznego; z egzaminu pisemnego; z egzaminu wstępnego;

- zawiadomień o przyjęciu (nieprzyjęciu na I rok studiów).

Drukowane są również (w formie nadającej się do wysłania pocztą następujące zestawienia: kandydatów na I rok studiów w roku akademickim, wyników egzaminów wstępnych na I rok studiów dziennych, wyników rekrutacji na I rok studiów, meldunek SzW-4 o naborze; rozkład i dystrybuanta ocen z przedmiotów kierunkowych, rozkład wyników uzyskanych przez absolwentów z określonego województwa lub określonej miejscowości, analiza poziomu przygotowania kandydatów, wyników egzaminów dla danego kierunku, protokół końcowy Wydziałowej Komisji Rekrutacyjnej (rys. 2).

OSZACOWANIE KOSZTÓW

Na koszty eksploatacji systemu składają się koszty przygotowania dokumentów i maszynowych nośników informacji, koszty pracy maszyny cyfrowej oraz koszty obsługi systemu.

Dla przeprowadzenia rekrutacji dla każdego kandydata trzeba wydziurkować 6—8 kart maszynowych. Przetwarzanie dla 1000 kandydatów wymaga ok. 1 godz. pracy procesora R-32 (tj. około 4 godzin pracy START-STOP). Łączny koszt eksploatacji dla 1000 kandydatów wynosi ok. 32 tys. zł. Ze wzrostem liczby kandydatów koszt nie rośnie jednak proporcjonalnie. Nie ma bowiem potrzeby zwiększania liczebności obsługi systemu. Należy zaznaczyć, że wykorzystywany czas pracy maszyny w dużym stopniu zależny jest od organizacji przetwarzania. Koszty z tym związane są niższe, gdy przetwarzanie odbywa się dla wielu kierunków grupowo, a nie dla poszczególnych kierunków oddzielnie. Zaleca się więc, jeśli to jest możliwe, tworzenie większych wsadów.

* * *

Wdrożenie systemu REKRUTACJA w zasadniczy sposób usprawnia pracę komisji rekrutacyjnych i administracji uczelni. Powoduje ujednolicenie trybu postępowania członków komisji oraz pełną standaryzację dokumentów. Zapewnia zwiększenie szybkości i rzetelności opracowań danych oraz wzmaga kontrolę informacji. Dzięki automatyzacji procesu kwalifikowania kandydatów oraz wstępnemu przyznawaniu pomocy materialnej, system pozwala na szybsze przygotowanie materiału decyzyjnego dla komisji i władz uczelni. Zwiększona pracochłonność na etapie przygotowawczym przyspiesza pracę sekretarzy w czasie trwania egzaminów. System umożliwia archiwację danych o kandydatach oraz pozwala na zabezpieczenie ochrony danych ewidencyjnych i kwalifikacyjnych kandydatów.

UNIWERSYTET MIKOLAJA KOPERNIKA W TORUNIU 80.07.18

WYNIKI EGZAMINOW WSTEPNYCH NA I ROK STUD W R.AK. 80/81

TB-RK-5-02 STR. 01

SYMBOL	NAZWA KIERUNKU	LIMIT	KANDYDACI	ZDALI EGZAMIN WSTEPNY								NIE ZDALI EGZ.				REZERWA MIEJSC	UWAGI
				PRZYJECI	PRZYJECI NA I ROK STUD.					NIE PRZYJECI							
			OGOLEM	BEZ EGZ.	OGOLEM	ABS. LO	KOBIET	POCH ROB.	POCH CHL.	OGOLEM	POCH ROB.	POCH CHL.	OGOLEM	POCH ROB.	POCH CHL.		
	HISTORIA	42	173	12	30	27	18	3	9	11	1	0	120	10	24	0	
	FILOLOGIA POLSKA	70	131	17	28	28	21	2	3	0	0	0	86	9	21	25	
	FILOLOGIA KLASYCZNA	8	11	0	8	8	7	0	2	0	0	0	3	0	0	0	
	FILOLOGIA GERMANSKA	45	139	2	43	40	36	3	9	4	0	0	90	6	24	0	
	ARCHEOLOGIA	7	26	1	6	6	3	1	0	0	0	0	19	1	6	0	
	PEDAGOGIKA	31	131	2	29	22	26	2	11	20	1	6	80	13	19	0	
	BIBLIOTEKOZNASTWO	30	61	1	22	20	22	1	8	0	0	0	38	5	11	7	
	NAUCZANIE POCZATKOWE	40	50	1	23	22	22	0	5	0	0	0	26	1	11	16	
	BIOLOGIA	60	163	15	46	43	42	2	15	29	2	4	73	5	26	-1	
	GEOGRAFIA	30	88	6	24	24	15	9	5	12	0	1	46	8	11	0	
	ASTRONOMIA	10	18	0	7	7	2	0	3	0	0	0	11	2	4	3	
	CHEMIA	70	71	8	27	24	15	4	9	0	0	0	36	9	11	35	
	FIZYKA	90	28	10	12	9	7	1	2	0	0	0	6	1	0	68	
	MATEMATYKA	110	56	7	30	28	19	3	2	0	0	0	19	4	3	73	
	ADMINISTRACJA	130	313	0	0	0	0	0	0	0	0	0	0	0	0	130	
	PRAWO	120	540	0	0	0	0	0	0	0	0	0	0	0	0	120	
	KONSERWACJA I REST.	20	125	0	20	16	18	3	3	35	1	4	70	0	8	0	
	KONSERWATORSTWO-MUZ.	20	67	1	19	10	13	0	4	2	0	0	45	0	7	0	
	WYCHOWANIE PLASTYCZ.	40	203	6	34	11	21	2	14	37	0	13	126	8	8	0	
	EKON. I ORG. PROD.	210	368	6	100	70	60	10	20	0	0	0	262	37	63	10	
	EKONOMICZNO-SPOLECZ.	25	34	0	10	10	6	0	2	0	0	0	24	2	5	15	
	RAZEM UCZELNIA	1208	2796	95	518	425	373	46	126	150	5	28	1180	121	262	595	

KONIEC ZESTAWIENIA TB-RK-5-02 DATA PODPIS

Rys. 2. Przykład drukowanego zastawienia: wyniki egzaminów wstępnych na I rok studiów dziennych

Efektywność systemu rośnie wraz ze stopniem centralizacji przygotowania danych. Elementem determinującym wysoką efektywność jest zapewnienie przetwarzania na życzenie sekretarzy (natychmiastowe). Cennym efektem działania systemu jest głęboka analiza wyników egzaminów i ocena przygotowania kandydatów. Ze względu na dużą pracochłonność tego typu analiz były one dotychczas rzadko podejmowane. Sprawozdania i meldunki wykonywane w systemie posiadają formę wymaganą przez ministerstwo i nadają się do bezpośredniego wysyłania.

System opracował zespół Ogólnouczelnianego Ośrodka Obliczeniowego Uniwersytetu Mikołaja Kopernika w Toruniu w składzie: M. Blechacz, B. Gliniecki, M. Mądrala, S. Kamiński, M. Sytniewski, K. Wieczorkowski. Oprogramowany jest w języku PL/I i pracuje pod kontrolą systemu operacyjnego OS na maszynie R-32. W pierwszej wersji został wdrożony na kierunku chemii w 1978 r. W 1979 r. przeprowadzono pełną eksploatację dla całej uczelni. W obecnej poprawionej wersji był wykorzystywany z powodzeniem, również dla całej uczelni, w roku 1980.

System udokumentowany jest wg standardów jednolitego systemu i przekazany do Problemu Resortowego R.I-14 „Rozwój komputeryzacji szkół wyższych". Jest rozpowszechniany przez jednostkę autorską oraz przez ZETO-Wrocław.

KRZYSZTOF GERWIN
Centrum Informatyki Gospodarki Morskiej
Gdańsk
JERZY SUKIENNIK
Stocznia Marynarki Wojennej
Gdynia

Zastosowanie systemu operacyjnego R 800 na standardowym zestawie MERA 9150

W kraju pracuje aktualnie kilkaset zestawów MERA 9150, a także kilkadziesiąt zestawów SEECHECK oraz kilka minikomputerów typu R850. W paru ośrodkach krajowych prowadzone są próby zastosowania systemu operacyjnego minikomputerów serii R800 na standardowych lub częściowo rozbudowanych zestawach MERA 9150. Celem tych zastosowań jest wypełnienie na rynku krajowym luki wynikłej z braku dostatecznej podaży uniwersalnych minikomputerów.

Możliwości zastosowań zestawów MERA 9150 ze „standardowym" systemem operacyjnym 7E — ukierunkowanym na proces przygotowania i wstępnego przetwarzania danych — zostały szeroko opisane [2, 3] i należy przypuszczać, że nie wymagają dodatkowej prezentacji. Wspomnieć jedynie należy, że wielu użytkowników stosuje te zestawy również do kompleksowego przetwarzania danych [1], oczywiście na małą skalę.

Informacje o poszczególnych urządzeniach minikomputera R850 oraz o systemie operacyjnym R800 były również publikowane [4]. Niniejszy artykuł ogranicza się zatem do informacji podstawowych. Z punktu widzenia sprzętu systemy minikomputerowe z serii R800 i R1800 w niczym nie przypominają „starych" systemów SEECHECK, z których wywodzi się MERA 9150. Przykładowo — nowe jednostki centralne wyposażone są w pamięć operacyjną, odpowiednio — 128 i 512 KB. Zastosowane są stosunkowo „duże" dyski, np. 4 × 66 MB oraz monitory ekranowe o pojemności 1920 znaków z drukarkami terminalowymi. Ostatnio pojawiły się również telewizory kolorowe adaptowane do wyświetlania informacji tekstowej lub graficznej oraz terminale odczytujące pismo odręczne. Rozwój sprzętu spowodował znaczne zmiany w systemie operacyjnym oraz języku programowania. Zastosowano sprawdzone w praktyce i odpowiednio rozbudowane funkcje operatorskie sterujące procesem, klawiaturowego

---

Mgr inż. KRZYSZTOF GERWIN, po ukończeniu w 1968 r. Wydziału Elektroniki Politechniki Gdańskiej, pracował w Instytucie Informatyki tejże Uczelni, zajmując się różnymi zastosowaniami komputerów i maszyn analogowych. W latach 1972—1981 pracował w Centrum Informatyki Gospodarki Morskiej, najpierw w służbie technicznej, a następnie w pionie projektowania, zajmując się m.in. zagadnieniami komputeryzacji terminali kontenerowych. W 1981 r. podpisał kontrakt z brytyjską firmą Rediffusion Computers Ltd., podejmując w niej obowiązki analityka-programisty.

Inż. JERZY SUKIENNIK ukończył w 1966 r. studia w Pradze (CSRS) na kierunku mechanizacja i automatyzacja zarządzania. Do roku 1977 pracował w Stoczni im. Komuny Paryskiej jako szef biura informatyki. Ostatnio pracuje w Stoczni Marynarki Wojennej jako kierownik zakładowego ośrodka elektronicznej techniki obliczeniowej. Pionier wszechstronnego wykorzystania minikomputerowych wielostanowiskowych systemów przygotowania danych Seecheck (MERA 9150) w Polsce, autor szeregu publikacji i referatów w tym zakresie.

wprowadzania i sprawdzania danych oraz nadzorcze (superoperatorskie), sterujące procesem operowania całym zestawem urządzeń i pełnym zakresem funkcji.

Zasadnicze zmiany nastąpiły w języku programowania oraz organizacji zbiorów. Dotyczą one szczególnie dwu nowych funkcji:

- wprowadzenia pojęcia tzw. programu kontroli pola, który sprawdza poprawność wprowadzanych danych lub umożliwia bezpośrednie, konwersacyjne przeszukiwanie i aktualizowanie zbiorów danych (w programie kontroli pola istnieje możliwość dowolnego formatowania danych, własnych list operacyjnych oraz komunikatów. Program ten umożliwia kontrolę pól w konfrontacji wzajemnej oraz w konfrontacji z uprzednio założonymi zbiorami; Kontrola może być dokonywana w trakcie wprowadzania danych, umożliwiając opracowywanie prostszych systemów przetwarzania w trybie *on-line*

- tworzenie hierarchicznie zorganizowanych zbiorów danych z dostępem bezpośrednim (indeksowo-sekwencyjnym). Problematyka zbiorów w systemie R800 opisana jest szerzej w dalszej części artykułu.

Nowe możliwości systemu operacyjnego są w porównaniu do standardowego oprogramowania zestawów MERA 9150 na tyle atrakcyjne, że skłoniły do podjęcia na dostępnych w kraju minikomputerach prób zastosowania specjalnie wygenerowanego systemu operacyjnego R800.

Podstawowymi ograniczeniami są oczywiście:

- mała pojemność pamięci operacyjnej MERA 9150 (32 K słów 16-bitowych)

- mała pojemność dysków magnetycznych (5 lub 10 MB)

- mała pojemność monitorów ekranowych

- brak drukarek terminalowych.

Podkreślenia wymaga jednak fakt łatwego przenoszenia „starego" oprogramowania i zbiorów z systemu operacyjnego 7E na R800. Przeróbki wymagają jedynie tzw. programy kontroli rekordu oraz te programy, w których używano dwuliterowych skrótów instrukcji. System operacyjny R800 pozwala na jednoczesne wprowadzanie danych przez operatorki z 8—12 stanowisk lub obsługę 4—6 terminali przeznaczonych do pracy bezpośredniej na zbiorach danych; obie te funkcje mogą być oczywiście wykonywane równolegle, ale należy pamiętać, że sam system R800 zajmuje minimum 23 K słów pamięci operacyjnej.

Charakterystyczne elementy oprogramowania minikomputerów serii R800/R1800

Pierwszym, bardzo istotnym elementem w charakterystyce oprogramowania minikomputerów R800/R1800 jest zachowanie wszystkich funkcji wcześniejszego oprogramowania z jednoczesnym zachowaniem zewnętrznej struktury systemu operacyjnego, tzn. z zachowaniem podziału funkcji systemu na część operatorską i nadzorczą (superoperatorską) oraz z zachowaniem zasady rozdzielenia wszystkich elementów oprogramowania użytkowego na tzw. biblioteki. Mówiąc z pewnym uproszczeniem — zachowano całe wcześniejsze oprogramowanie zestawów SEECHECK (MERA 9150), a więc zachowano wszystko to, co w przeszłości zadecydowało o ich sukcesie. Systemy serii R800/R1800 są więc nadal doskonałymi systemami do wprowadzania danych i w tym zakresie są kontynuacją linii SEECHECK/R300-400, wzbogaconą o nowe możliwości.

Rozwijając oprogramowanie pierwotne i nadając mu charakter uniwersalny, dobrze przystosowany do pracy w trybie *one-line*, opracowano mechanizm tworzenia i obsługi zbiorów indeksowych. Z punktu widzenia programisty na mechanizm ten składa się aktualnie sześć odmian instrukcji indeksowego odczytu informacji ze zbioru, cztery instrukcje aktualizacji zbioru i tablic indeksowych oraz kilka kodów sterujących. Jak widać, zestaw środków do pracy na zbiorach indeksowych jest dość bogaty i jest stale rozszerzany.

Istotnym rozszerzeniem możliwości systemu R800/R1800 w zakresie wprowadzania danych jest utworzenie czegoś w rodzaju „drugiej warstwy" oprogramowania w stosunku do formatów wejścia. Polega to na wprowadzeniu wspomnianego już programu kontroli pola, który jest związany z pracą standardową. Przejście od dowolnego formatu rekordu wejścia do programu kontroli pola może być zaprogramowane po każdym polu formatu, natomiast powrót z programu do formatu może być wykonany w dowolnym miejscu programu i w dowolne miejsce formatu. Ponieważ wejście do programu kontroli pola może być przewidziane także po ostatnim polu w formacie, funkcja programu kontroli pola zawiera w sobie i zastępuje funkcje tzw. programu kontroli rekordu. Jest oczywiste, że dzięki programowi kontroli pola można bardzo rozszerzyć i usprawnić kontrolę poprawności wprowadzanych danych. Pozwala on również sterować procesem wprowadzania, np. programując zmiany w powiązaniach pomiędzy formatami wejścia, a także wykorzystywać wszystkie instrukcje czytania i aktualizacji zbiorów indeksowych. Można więc, w pewnym sensie, równolegle prowadzić proces rejestracji danych (tworzenia zbiorów roboczych) oraz aktualizacji lokalnej bazy danych (zbiorów stałych). Istnieje też możliwość oderwania się od rejestracji danych, wykorzystując ten tryb wyłącznie do aktualizacji lokalnej bazy danych przy zachowaniu wszystkich zalet pracy z uwzględnieniem formatów rekordów wejścia.

Sam język programowania EDITOR jest włączony w oprogramowanie serii R800/R1800 na tych samych zasadach jak we wcześniejszych systemach język VALIDATOR — jest zintegrowany z systemem operacyjnym. Interesujące jest, że podzbiór instrukcji tego języka jest używany do pisania makroprocedur. W terminologii serii R800/R1800 makroprocedura może być zdefiniowana jako program wiążący pomiędzy sobą makrorozkazy, a z kolei makrorozkaz może być tu zdefiniowany jako zapis dialogu pomiędzy operatorem a systemem operacyjnym (do automatycznego operowania) przy uruchamianiu dowolnej użytkowej funkcji systemu, np. programu kontroli zbioru, sortowania czy programu wyjścia. Zespół środków „makroprocedura/makrorozkaz" pełni tu zatem funkcję języka opisu zadań, ale nie tylko. W makroprocedurze mogą być ujęte elementy konwersacyjne, ułatwiające i przyspieszające dostęp do funkcji *on-line* systemu użytkowego. Jest to wykorzystywane w przypadku dużych systemów i zasadniczą zaletą zastosowania tego narzędzia jest ułatwienie pracy użytkownikowi — nie musi on już pamiętać nazw programów i zbiorów ani znać sposobu dochodzenia do poszczególnych funkcji systemu. W szczególnych przypadkach sterowania pracą systemu użytkowego można wykorzystać możliwość przenoszenia danych pomiędzy programem a makroprocedurą.

Dla systemów eksploatowanych w trybie *on-line* przygotowano instrukcje blokady jednoczesnego dostępu do rekordu w celu jego aktualizacji. Jest to wykorzystywane przy programowaniu obsługi zbioru — w sytuacji, gdy na zbiorze pracować będzie jednocześnie więcej niż jeden użytkownik. Bardzo przydatna w praktyce jest również instrukcja wydruku zawartości ekranu na drukarce terminalowej. Mechanizm wydruku jest tu tak przygotowany, że wydruk można łatwo realizować na dowolnej drukarce, inicjując go z dowolnego monitora. Systemy R800/R1800 są więc przygotowane do realizacji przepływów informacji pomiędzy terminalami, co jest często wymagane w sytuacji, gdy terminale są zainstalowane w znacznej od siebie odległości, a czynności różnych użytkowników systemu są od siebie wzajemnie uzależnione. W systemie MERA 9150/R800 może być do tego celu wykorzystywana drukarka systemowa.

Podstawowym narzędziem partiowego przetwarzania danych są tzw. programy wyjścia. Ich umiejscowienie w systemach R800/R1800 jest takie same, jak w systemach wcześniejszych, jednak mogą one jednocześnie przetwarzać konwencjonalną paczkę (ang. *batch*) oraz pewną liczbę (do 63) zbiorów indeksowych, co oznacza znaczny postęp w porównaniu do wcześniejszego oprogramowania.

Firma REDIFFUSION COMPUTERS LTD. (dawniej REDIFON) podejmuje energiczne działania na rzecz stworzenia oprogramowania dla ciągle rozszerzającej się oferty sprzętowej oraz nowych zastosowań swoich zestawów minikomputerowych. Sprawy te nie mieszczą się w temacie niniejszego artykułu, dlatego też ograniczymy się do wzmianki, że chodzi tu m.in. o oprogramowanie dla terminali VIEWDATA (połączenie minikomputera z kolorowym telewizorem), dla czytników pisma ręcznego, dla

przetwarzania tekstów i innych zastosowań z silnym zaakcentowaniem transmisji danych i zdalnego dostępu do zbiorów.

Powyższe informacje odnoszą się jedynie do zasadniczych, zewnętrznych różnic pomiędzy systemem 7E (SEECHECK, MERA 9150) a systemami R800/R1800. Szczegóły znajdują się w materiałach firmowych [5, 6]. Próby zastosowania wcześniejszych wersji oprogramowania R800/R1800 na standardowych zestawach MERA 9150 dały wyniki pozytywne. Świadczą o tym podane dalej przykłady zastosowań. Pewnym problemem może być natomiast uzyskanie krótkich czasów reakcji w przypadku pracy kilku terminali. Jak już wspomniano, podstawowym elementem limitującym jest wielkość pamięci operacyjnej. Jeżeli występują problemy z uzyskaniem wymaganej sprawności systemu, to pierwszym krokiem powinno być zwiększenie pamięci operacyjnej z 32 do 64 K słów. Przy najnowszych wersjach oprogramowania serii R800/R1800 pamięć 64 K słów 16-bitowych stanowi praktyczne minimum. Pozostałe elementy ograniczające są już zależne od konkretnych zastosowań. W pierwszej kolejności chodzi tu o wielkość pamięci dyskowej. Jak wykazały próby, oprogramowanie serii R800/R1800 w zależności od wielkości zbiorów użytkowych może działać na dyskach o pojemności 2 × 5 MB (a nawet 1 × 5 MB). W materiałach firmowych [5] znajdują się ścisłe reguły obliczania zajętości dysku przez zbiory użytkowe.

Mając na względzie powyższe stwierdzenia, można wyprowadzić generalny wniosek, że zestawy sprzętowe MERA 9150 wyposażone w oprogramowanie serii R800/R1800 stają się minikomputerem uniwersalnym, przystosowanym szczególnie dobrze do pracy w trybie *on-line*. W standardowym zestawie, tj. z pamięcią operacyjną 32 K słów i z jednym dyskiem 5 MB, można pracować na 4—6 terminalach. Rozszerzenie pamięci operacyjnej do 64 K słów zapewnia pracę w pełnym standardowym zestawie sprzętowym MERA 9150 przy założeniu, że wielkość obszaru systemowego na dysku (wraz z bibliotekami) nie przekroczy 2 MB. Dla większych zastosowań należy się liczyć z wymaganą wielkością obszaru systemowego 4 MB, co z kolei implikuje potrzebę dołączenia drugiej pamięci dyskowej 5 MB. Dla porównania — standardowo oferowana przez producenta pamięcią dyskowa zestawów R800/R1800 jest jednostka o pojemności 33 lub 66 MB.

Zbiory w systemie R800

Zagadnienie zbiorów stanowi zwykle zasadniczy element w ocenie możliwości konkretnego zastosowania minikomputera, dlatego sprawie tej poświęcamy nieco więcej miejsca. System operacyjny R800 umożliwia tworzenie i przetwarzanie następujących rodzajów zbiorów:

- zbiory robocze transakcyjne (zwane uprzednio „paczkami danych") przechowywane okresowo

- zbiory główne („master") — kartoteki przechowywanych długookresowo danych stałych aktualizowane w systemie R800

- zbiory posortowane (robocze lub główne) będące uporządkowanymi listami wskaźników do rekordów w zbiorach podlegających sortowaniu (zbiory te nie zawierają danych i mogą być kasowane bez naruszania zawartości zbiorów roboczych lub głównych); umożliwiają one sekwencyjne przetwarzanie danych w żądanym porządku (w odróżnieniu od organizacji zbiorów seryjnych zawierających dane w porządku ich wprowadzania na dysk)

- indeksy (tablice indeksowe) będące mechanizmem bezpośredniego indeksowo-sekwencyjnego dostępu do danych (do poszczególnych rekordów lub tzw. dokumentów, stanowiących hierarchię wielu rekordów); indeksy są zbiorami systemowymi składającymi się z ciągów „kieszeni", z których każda zawiera wartość klucza i wskaźnik do rekordów w zbiorze głównym

- bazy danych — traktowane jako zespoły zbiorów głównych dotyczących tej samej dziedziny oraz utworzonych dla tych zbiorów indeksów systemowych (dla tego samego zbioru głównego można utworzyć szereg indeksów wg różnych kluczy, dzięki czemu uzyskuje się swego rodzaju „listy inwersyjne" umożliwiające tworzenie wielu mechanizmów dostępu do danych).

Dodatkowego omówienia wymagają następujące trzy podstawowe metody organizowania zbiorów głównych:

- Zbiór prosty — złożony z rekordów „jednopoziomowych", np. zbiór rekordów zawierających dane o poszczególnych pracownikach.

- Zbiór hierarchiczny — zawierający tzw. dokumenty „dwupoziomowe", ze zmienną liczbą rekordów poszczególnych typów; np. dla jednego rekordu „czołowego" identyfikującego materiał może wystąpić:

— 1-n rekordów danych o potrzebach materiałowych
— 1-n rekordów danych o planie zaopatrzenia
— 1-n rekordów zamówień złożonych na dany materiał
— 1-n rekordów danych o realizacji dostaw oraz rozchodach materiału.

- Zbiór gniazd hierarchicznych — złożony z tzw. dokumentów „wielopoziomowych" — zawierających w swoim układzie również tzw. poddokumenty (wtórnie zależne, zgodnie z zasadą *owner-member*); np. rozbudowany zbiór zamówień materiałowych w układzie:

— 1 rekord czołowy (dane „adresowe" o dostawcy materiałów)
— 1-n rekordów identyfikujących materiał
— 1-n rekordów specyfikujących zamawiane ilości w poszczególnych terminach
— 1-n rekord dotyczący warunków odbioru i płatności.

Z powyższego wynika, że możliwe jest tworzenie zbiorów o złożonej strukturze logicznej. Warunki po temu stwarza bardzo elastyczny mechanizm tworzenia tablic indeksowych, zachowujący wszystkie własności procesu sortowania (podstawowy poziom tablicy indeksowej tworzy normalny program sortujący) z wyjątkiem długości klucza, który przy indeksacji nie może przekraczać 36 znaków (klucz numeryczny) oraz 24 znaków (klucz alfanumeryczny). Dodatkowe korzyści stwarza wspomniana wyżej możliwość tworzenia dowolnej liczby tablic indeksowych, przy czym każda z nich może obejmować swoim zakresem więcej niż jeden zbiór. Oprócz dostępu indeksowego możliwy jest także dostęp sekwencyjny, przy czym może on być związany z dostępem indeksowym w tym sensie, że np. odczyt nagłówka dokumentu może być indeksowy, a przetwarzanie dalszej części dokumentu może być sekwencyjne. Bardzo istotną zaletą jest możliwość określenia w programie miejsca, w którym rekord ma być dopisany do zbioru. Dzięki temu w zbiorze po indeksowej aktualizacji może być zachowana logiczna spójność dokumentów, pozwalająca na ich sekwencyjne przetwarzanie.

Z obserwacji wynika, że ograniczenie dotyczące długości klucza nie jest istotnym utrudnieniem przy projektowaniu systemów, ponieważ może być łatwo zrekompensowane możliwością budowy kilku tablic indeksowych dla jednego zbioru. Interesujące jest to, że nawet dla kluczy numerycznych buduje się tablice typu E, tj. z ograniczeniem klucza do 24 znaków. Decyduje o tym możliwość pracy na tablicach typu E z częściowo określonym kluczem, w których znakiem „gwiazdki" można zastępować nieznane lub obojętne elementy klucza (co umożliwia dostęp do grupy podobnych danych).

Ważną własnością jest wprowadzanie nowych kluczy do tablicy indeksów w sposób uporządkowany, tzn. nowy indeks jest w tablicy wprowadzany w miejsce odpowiadające wartości klucza. Jest oczywiste, że przy dużych tablicach operacja ta może być czasochłonna, co jest jednak w pełni rekompensowane bieżącą gotowością tablicy do sekwencyjnego przetwarzania zbioru według wartości kluczy. Pozwala to w wielu sytuacjach wyeliminować potrzebę sortowania zbioru, a także znacznie ułatwia programowanie różnorodnych procedur przeszukiwania zbiorów.

* * *

Opisane możliwości systemu operacyjnego R800 eksploatowanego na zestawach minikomputerowych MERA 9150 nadają im cechy uniwersalnego systemu przetwarzania

danych, pozwalającego realizować nie tylko proces przygotowania, kontroli i wstępnego przetwarzania danych, ale również tworzyć systemy informatyczne eksploatowane w trybie *on-line*. System MERA 9150/R800 pozwala na szybkie przeszukiwanie oraz aktualizację zbiorów danych zorganizowanych jako kartoteki z dostępem bezpośrednim (indeksowo-sekwencyjnym). Na żądanie uzyskiwać można również raporty wg doraźnie przyjętych parametrów, np. wykaz pozycji materiałowych z zapasami poniżej ustalonego minimum. Raporty te mogą być drukowane na drukarce systemowej, a wprowadzanie parametrów i operowanie wydrukiem może być realizowane za pomocą zdalnego monitora ekranowego zainstalowanego bezpośrednio u użytkownika systemu.

Intencją autorów artykułu było wykazanie, że system MERA 9150 może znaleźć nowy obszar zastosowań, a jego produkcja powinna być nadal realizowana i znacznie rozszerzona, bowiem dla celów przetwarzania danych gospodarczych w małych i średnich przedsiębiorstwach odpowiednio rozbudowana konfiguracja systemu MERA 9150 wraz z oprogramowaniem analogicznym do systemu R800 może stworzyć autonomiczny, uniwersalny system minikomputerowy.

LITERATURA

[1] Janik A., Myszka A.: Minikomputerowy system dyspozytorski dla Kombinatów budowy Domów. Konferencja naukowa „Obiektowe systemy informatyczne — AMPIG-76", Poznań, kwiecień 1976

[2] Sukiennik J.: Zastosowanie systemu przygotowania i wstępnego przetwarzania danych SEECHECK w Stoczni im. Komuny Paryskiej. INFORMATYKA 5/74

[3] Sukiennik J.: Uwagi doświadczonego użytkownika. INFORMATYKA 5/77

[4] Sukiennik J., Szumielewicz J.: R850 — nowy system firmy Redifon. INFORMATYKA 1/80

[5] R800 — Podręcznik programisty

[6] R800 — Podręcznik operowania.

Dwa sposoby obsługi tablic rozproszonych

Załóżmy, że mamy dany zbiór $\{K_1,...,K_n\}$ kluczy, z których m ($m \leqslant n$) ma zostać zapamiętanych w pewnej strukturze danych, przy czym struktura ta ma dopuszczać trzy operacje: dołączenie nowego klucza, usunięcie starego oraz zlokalizowanie każdego z kluczy.

MOTYWACJA

Problem powyższy jest bardzo często spotykany w praktyce, np. przy konstruowaniu baz danych czy tablic symboli podczas kompilacji programów. Typowym zbiorem kluczy jest zbiór identyfikatorów alfanumerycznych. Jeśli identyfikatory (nazwiska, nazwy zmiennych itp.) składają się z dziesięciu znaków z alfabetu 36-literowego, to jest ich 36^{10} (tj. rzędu 10^{15}), natomiast w bazie danych czy w tablicy symboli wystąpi znacznie mniejsza ich liczba.

Zaprezentowane poniżej rozwiązania, polegające na użyciu tablic rozproszonych (ang. *hash tables*), są nadzwyczaj użyteczne z praktycznego punktu widzenia, ponieważ wyszukanie klucza wymaga średnio mniej niż trzech dostępów do tablicy, niezależnie od liczby kluczy w niej umieszczonych.

OPIS METODY LUHNA

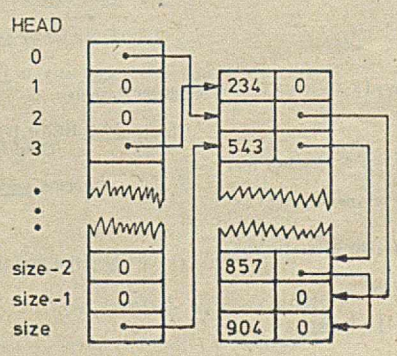
(Knuth D. E.: The Art of Computer Programming. Vol. 3, Addison-Wesley, 1973)

Załóżmy, że mamy daną funkcję h: $\{K_1,...,K_n\} \rightarrow \{1,..., size\}$, gdzie $size$ jest zakresem tablicy rozproszonej HT. Jeśli K jest liczbą całkowitą, to możemy zdefiniować $h(K) = 1 + K \bmod size$. Różne sposoby definiowania h czytelnik znajdzie w wyżej wspomnianej książce Knutha (rozdz. 6.4). Oczywiście funkcja h nie musi być różnowartościowa, a więc metoda wstawiania (wyszukiwania, usuwania) kluczy powinna uwzględniać przypadek, w którym różne klucze znajdujące się w HT mają te same wartości h. Metoda Luhna polega na umieszczeniu wszystkich kluczy K takich, że $h(K) = i$ ($1 \leqslant i \leqslant size$) w tablicy HT w liście o początku $HEAD[i]$. W celu wyszukania klucza K musimy po prostu przejrzeć listę o adresie $HEAD[h(K)]$. Usunięcie klucza K z HT, to usunięcie go z właściwej listy.

Metoda ta znana jest również pod nazwą metody listowej (ang. *chaining*).

STRUKTURA DANYCH

Dla uproszczenia zapisu przyjmujemy, że klucze są liczbami całkowitymi. Zapamiętujemy je w tablicy $HT[1:size, 1:2]$ przy czym pierwszym atrybutem jest klucz, drugim zaś wskaźnik do następnego elementu listy. Początki list pamiętane są w tablicy $HEAD[0:size]$. Element $HEAD[0]$ jest adresem listy miejsc wolnych w HT. Uzyskujemy zatem następującą przykładową strukturę:



Początkowe wartości (przed wpisaniem kluczy) są następujące:

HEAD[0] = 1

HEAD[i] = 0 dla 1 ≤ i ≤ size

HT[i,2] = i+1 dla 1 ≤ i < size

HT[size,2] = 0.

Ponadto zakładamy, że funkcja *h* jest zdefiniowana przez użytkownika.

OPIS PROCEDUR

Procedury *LInsert*, *LDelete* oraz funkcja *LFind* mają następujące parametry:

- klucz *K*
- tablicę całkowitą *HEAD[0 : size]*
- tablicę całkowitą *HT[1 : size, 1 : 2]*
- zmienną całkowitą *size*.

Procedura *LInit* ma jedynie trzy powyższe parametry (bez klucza *K*).

```
procedure LInit(HEAD,HT,size); integer array HEAD,HT;
                               integer size;
comment inicjalizuje wartości struktury danych;
begin integer i;
  HEAD[0] := 1;
  for i := 1 step 1 until size do
    begin HEAD[i] := 0; HT[i,2] := i+1 end;
  HT[size,2] := 0
end LInit;
procedure LInsert(K,HEAD,HT,size); integer K,size;
                                   integer array HEAD,HT;
comment wstawia klucz K do tablicy HT;
begin integer i,x;
if LFind(K,HEAD,HT,size) = 0 then
  if HEAD[0] = 0 then write("przepełniona tablica HT")
  else begin i := h(K); x := HEAD[0]; HEAD[0] := HT[x,2];
       HT[x,1] := K; HT[x,2] := HEAD[i]; HEAD[i] := x
  end
end LInsert;
procedure LDelete(K,HEAD,HT,size); integer K,size;
                                   integer array HEAD,HT;
comment usuwa klucz K z tablicy HT;
begin integer i,prev,next;
  i := h(K); prev := 0; next := HEAD[i];
  while next ≠ 0 do
    if HT[next,1] = K then
      begin if prev = 0 then HEAD[i] := HT[next,2]
            else HT[prev,2] := HT[next,2];
            HT[next,2] := HEAD[0]; HEAD[0] := next; next := 0
      end
    else begin prev := next; next := HT[next,2] end
end LDelete;
integer procedure LFind(K,HEAD,HT,size); integer K,size;
                                         integer array HEAD,HT;
comment wynikiem jest adres klucza K w tablicy HT, jeśli
        K nie występuje w HT, to LFind = 0;
begin integer i,next;
  i := h(K); next := HEAD[i]; LFind := 0;
  while next ≠ 0 do
    if HT[next,1] = K then begin LFind := next; next := 0 end
    else next := HT[next,2]
end LFind;
```

Metoda Luhna jest bardzo efektywna, gdyż średnia długość listy w tablicy *HT* jest mniejsza niż 1,5 niezależnie od liczby elementów zapamiętanych w *HT*, wymaga jednak dodatkowych miejsc pamięci dla wskaźników. Jeżeli użytkownik chce tego uniknąć, może użyć algorytmu Maddisona, przy czym koszt czasowy tej metody jest większy (zwłaszcza podczas wykonywania procedury *MInsert* — dopisywania).

OPIS METODY MADDISONA

(The Computer Journal, vol. 23, No 2, May 1980, 188—189)
Metoda ta jest poprawioną wersją modyfikacji Brenta (CACM, 16(1973), 105—109).

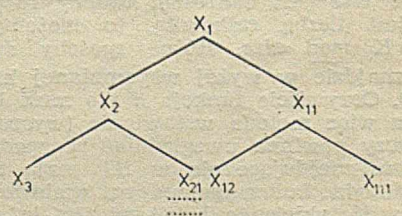
Niech $X_{i_1 \cdots i_q}$ oznacza adres:

$$X_{i_1 \cdots i_q} = \begin{cases} h_1(K) \text{ dla } q=1 \text{ oraz } i_q=1, \\ X_{i_q-1} + h_2(K) + size * sg(size - h_2(K) - X_{i_q-1}) \text{ dla } q=1,\ i_q>1, \\ h_1(HT[X_{i_1 \cdots i_{q-1}}]) \text{ dla } q>1,\ i_q=1, \\ X_{i_1 \cdots i_q-1} + h_2(HT[X_{i_1 \cdots i_{q-1}}]) + size * sg(size - \\ \quad - h_2(HT[X_{i_1 \cdots i_{q-1}}]) - X_{i_1 \cdots i_q-1} \text{ dla } q>1,\ i_q>1. \end{cases}$$

gdzie h_1 oraz h_2 są funkcjami ze zbioru kluczy w zbiór możliwych adresów w tablicy *HT*, *size* — zakresem *HT*, a *sg* definiujemy:

$$sg(x) = \begin{cases} 0 \text{ dla } x \geq 0 \\ -1 \text{ dla } x < 0 \end{cases}$$

Zauważmy, że z $X_{i_1 \cdots i_q}$ możemy utworzyć drzewo binarne:



Załóżmy, że chcemy wstawić klucz K do tablicy HT (procedura MInsert). Jeśli X_1 jest adresem miejsca wolnego ($HT[X_1] \in \{\alpha, \beta\}$), to $HT[X_1] := K$. W przeciwnym przypadku sprawdzamy X_2. Jeśli jest wolne, to K wstawiamy w miejsce X_2, jeśli nie, to sprawdzamy X_{11}. Jeśli X_{11} jest wolne, to $HT[X_{11}] := HT[X_1]$ i $HT[X_1] := K$. Przedstawienie to zmniejsza średni czas dostępu do K o co najmniej dwie próby, natomiast zwiększa czas dostępu do $HT[X_{11}]$ o jedną próbę. Postępowanie to kontynuujemy (sprawdzając pozycje X_3, X_{21} X_{12}, X_{111}, X_4 itd.) do momentu, w którym zostaje znalezione wolne miejsce.

Wyszukiwanie klucza w tablicy (funkcja *MFind*) polega na kolejnym testowaniu pozycji X_1, X_2, X_3,...X_j,..., a usunięciu klucza — na wpisaniu w jego miejsce wyróżnionego symbolu β.

Średnia liczba prób podczas wstawiania klucza zależy od współczynnika λ zapełnienia tablicy (λ = (*liczba elementów wstawionych*)/*size*) i jest równa $\frac{1}{\lambda} \ln \frac{1}{1-\lambda}$ natomiast średnia liczba prób przy wyszukiwaniu kluczy jest mniejsza niż 2,5 niezależnie od liczby elementów w tablicy. Stąd wynika, że metoda Maddisona jest bardzo efektywna, o ile wyszukiwanie kluczy ma miejsce znacznie częściej niż ich wstawianie.

STRUKTURA DANYCH

Klucze z pewnego zbioru $\{K_1,...., K_n\}$ umieszczane są w tablicy HT [*1 : size*] ($size \ll n$) w adresach wyznaczonych przez funkcje $h_1 : \{K_1,...., K_n\} \to \{1,2, ..., size\}, h_2 : \{K_1,..., K_n\} \to \{2,3, ..., size-1\}$

przy czym wyniki funkcji h_2 powinny być względnie pierwsze z *size*. Np. jeśli klucze są liczbami całkowitymi, to $h_1(K)$ można określić następująco: $h_1(K) = 1 + K \bmod size$ i dalej, zakładając np. że *size* jest liczbą pierwszą, możemy zdefiniować $h_2(K) = 2 + K \bmod (size-2)$. Jeśli *size* jest potęgą dwójki, to wystarczy, aby wyniki h_2 były nieparzyste. Dokładniejsze omówienie funkcji h_1 i h_2 czytelnik znajdzie we wspomnianej wcześniej monografii Knutha (rozdz. 6.4).

Zakładamy, że symbole α oraz β nie występują w zbiorze kluczy i że początkowe wartości tablicy HT są równe α.

OPIS PROCEDUR

Parametrami procedur *MInsert*, *MDelete* oraz funkcji *MFind* są:

- klucz K typu T
- tablica HT[*1 : size*] typu T
- liczba całkowita size określająca zakres HT

Procedura *MInit* (nadanie wartości początkowych tablicy HT) ma jedynie dwa powyższe parametry (bez klucza K typu T).

Zakładamy istnienie zdefiniowanych przez użytkownika funkcji h_1 oraz h_2.

Procedura *Insert* proponowana przez Maddisona wymaga w niekorzystnym przypadku wykładniczej ilości pamięci dodatkowej (2^{size}), natomiast zaprezentowana poniżej procedura *MInsert*, zachowując koncepcję Maddisona, nie używa dodatkowej pamięci. Koszt czasowy *MInsert* jest tym razem nieco większy: średnia liczba prób jest równa $\frac{1}{\lambda} \ln \frac{1}{1-\lambda}$ każda próba kosztuje co najwyżej $\log_2 1 \left(\frac{1}{\lambda} * \ln \frac{1}{1-\lambda} \right)$ obliczeń funkcji h_1 lub h_2. W praktyce *MInsert* działa efektywnie, jeśli $\lambda \leqslant 0.9$.

```
procedure MInit(HT,size) ; T array HT; integer size;
comment inicjalizuje tablicę HT;
begin integer i;
  for i:=1 step 1 until size do HT[i] := α
end MInit;

procedure MInsert (K,HT,size) ; T K; T array HT; integer size;
comment wstawia klucz typu T do tablicy HT;
begin integer position,length,n,l,nl,rh; T ix;
            boolean procedure left;
            comment left = true, jeśli następny wierzchołek w drzewie
              utworzonym z X_{i1...ik} jest lewym synem;
  begin integer i,v;
    i:=position;
    for v:=1 step 1 until l-1 do i:= i div 2;
    left := i mod 2 = 0
  end left;
 integer procedure next location;
 comment wynikiem jest adres kolejnej próby;
 begin integer i;
    i:=position:=position+1; length:=0;
   while i≠1 do begin i:= i div 2; length:=length+1 end;
   nl:=h1(K); rh:=h2(K); l:=length;
   while l≠0 do
     begin if left then nl:=nl+rh
           else begin rh:=h2(HT[nl]) ; nl:=nl+rh end;
           if nl>size then nl:=nl-size;
           l:=l-1
     end;
   next location:=nl
 end next location;
 procedure shift;
 comment dokonuje niezbędnych przesunięć w HT;
 begin T x,y; integer iv; boolean loop;
   procedure MLEFT;
   begin
     while left and l>0 do
       begin nl:=nl+rh; l:=l-1;
         if nl>size then nl:=nl-size
       end
   end MLEFT;
   procedure MRIGHT;
   begin
     while (not left) and l>0 do
     begin y:= HT[iv] ; HT[iv] :=x; x:=y; iv:=nl;
         l:=l-1; rh:=h2(HT[nl]) ; nl:=nl+rh;
         if nl>size then nl:=nl-size
     end
 end MRIGHT;
```

```
x:=K;l:=length;nl:=h1(K); rh:=h2(K); MLEFT; loop:=true;
while loop do
  if l=0 then begin HT[nl]:=x; loop:=false end
  else begin iv:=nl; rh:=h2(HT[nl]); nl:=nl+rh; l:=l-1;
            if nl>size then nl:=nl-size;
            MRIGHT; MLEFT; y:=HT[iv]; HT[iv]:=x; x:=y end
end shift;
if MFind(K)≠0 then go to exit;
position:=0; n:=next location; ix:=HT[n];
while ix≠α and ix≠β do
  if length>size then begin write("przepełniona tablica HT");
                      go to exit end
  else begin n:=next location; ix:=HT[n] end;
  shift;
exit:
end MInsert;
procedure MDelete(K,HT,size); T K; T array HT; integer size;
comment usuwa klucz K typu T z tablicy HT;
begin integer i;
  i:=MFind(K,HT,size);
  if i≠0 then HT[i]:=β
end MDelete;
integer procedure MFind(K,HT,size); T K; T array HT;
                                    integer size;
comment wynikiem jest adres klucza K w tablicy HT, jeśli
        K nie występuje w HT, to MFind = 0;
begin integer j,nl,rh; T x;
  nl:=h1(K); rh:=h2(K); j:=MFind:=0; x:=HT[nl];
  while x≠K do
    begin j:=j+1;
      if j>size or x=α then go to exit;
      nl:=nl+rh;
      if nl>size then nl:=nl-size;
      x:=HT[nl]
    end;
  MFind:=nl;
  exit:
end MFind;
```

ANDRZEJ SZAŁAS
ZBIGNIEW ŚWIRSKI

Prawda, która może urazić

Mimo że zawarte poniżej stwierdzenia jednego z największych współczesnych autorytetów naukowych w dziedzinie oprogramowania mogą być dla wielu osób szokujące, postanowiliśmy przenieść je w możliwie najwierniejszym tłumaczeniu na nasze łamy — z numeru 5/82 amerykańskiego czasopisma naukowego SIGNPLAN NOTICES. Jeżeli stwierdzenia te zbulwersują środowisko informatyczne (również w innych krajach) i spowodują odpowiednią replikę — postaramy się udostępnić ją naszym Czytelnikom.

Redakcja

Czasem odkrywamy prawdy nieprzyjemne. Kiedykolwiek to czynimy popadamy w kłopoty. Tłumienie prawdy jest z naukowego punktu widzenia nieuczciwe, musimy więc powiedzieć ją, wiedząc jednak, że może to obrócić się przeciw nam. Jeżeli prawda jest dostatecznie przykra, nasze audytorium nie jest w stanie jej zaakceptować i uzna nas za całkowicie pozbawionych poczucia realizmu, beznadziejnych idealistów, niebezpiecznych rewolucjonistów, głupio naiwnych itp. (poza tym mówienie tego rodzaju prawdy prowadzi z pewnością do utraty popularności w wielu środowiskach, a w związku z tym na ogół nie jest pozbawione osobistego ryzyka — vide Galileo Galilei...).

Wydaje się, że informatyka poważnie cierpi wskutek konfliktu. Na ogół o nim się nie mówi i próbuje omijać go przez zmianę zainteresowań (np. w odniesieniu do COBOL-u można uczynić jedną z dwu rzeczy: walczyć z chorobą lub przyjąć, że ona nie istnieje; większość z ośrodków akademickich wybrała to drugie, łatwiejsze wyjście). Lecz pytam Was, czy jest to uczciwe? Czy nasze przedłużające się milczenie nie podrywa intelektualnej jedności informatyki? Czy przyzwoite jest jego kontynuowanie? Jeśli nie, jak należy mówić?

W celu uzmysłowienia skali problemu, wyliczyłem poniżej niektóre z tych prawd (prawie wszyscy informatycy, których znam, zgodzą się bez wahania ze zdecydowaną ich większością, a mimo to pozwalamy zachowywać się światu tak, jak byśmy o nich nie wiedzieli...):

- programowanie jest jedną z najtrudniejszych dziedzin matematyki stosowanej — gorsi matematycy niech więc lepiej pozostaną czystymi matematykami
- najłatwiejszymi zastosowaniami komputerów są obliczenia naukowo-techniczne
- narzędzia, których używamy, mają zasadniczy (i często zwodniczy) wpływ na nasz sposób myślenia, a w konsekwencji na nasze zdolności umysłowe
- FORTRAN, to „nieporządne dziecko", ma już prawie 20 lat i jest całkowicie nieadekwatny do wszelkich zastosowań komputerów, jakie sobie można dziś wyobrazić — obecnie jest on zbyt nieporęczny, zbyt ryzykowny i za drogi w użyciu
- PL/I — „nieuleczalna choroba" — należy raczej do sfery problemów niż sfery rozwiązań
- praktycznie nie można nauczyć dobrego programowania studentów, którzy przedtem zetknęli się z językiem

Wspomagane projektowanie systemów

Instytut Informatyki Akademii Ekonomicznej we Wrocławiu organizuje w czerwcu 1983 roku dwudniową konferencję „WSPOMAGANE KOMPUTEREM TWORZENIE SYSTEMÓW INFORMATYCZNYCH".

Celem konferencji jest zebranie i udostępnienie szerszemu ogółowi wniosków i praktycznych doświadczeń w dziedzinie wspomaganego komputerem projektowania i wdrażania systemów informatycznych. Celem umożliwienia prezentacji gotowych systemów i programów organizatorzy udostępniają komputer R-32 z systemem operacyjnym OS/JS wersja MVT oraz podsystemami TSO lub CRJE (pamięć operacyjna 512K bajtów, pamięć zewnętrzną dyskową EC 5061, system monitorów ekranowych EC 7900).

Obrady konferencji odbywać się będą w sekcjach tematycznych:

- systemy wspomaganego komputerem zarządzania
- systemy wspomaganego komputerem nauczania.

W ramach pierwszej sekcji wygłoszone zostaną referaty dotyczące następujących zagadnień:

- **cybernetyczne modele systemów informacyjnych tworzonych dla nowych struktur organizacyjnych**
- **metody i techniki wspomaganego komputerem projektowania różnych typów procesów decyzyjnych**
- **teoria i praktyka funkcjonowania i rozwoju informatycznych systemów zarządzania, w warunkach ograniczonych zasobów komputerowych**
- **problemy wspomaganego komputerem projektowania baz danych.**

W ramach sekcji drugiej przewiduje się pokaz wspomaganego komputerem systemu nauczania OSKAR opracowanego przez AE Wrocław oraz wymianę doświadczeń związanych z praktycznymi osiągnięciami w dziedzinie projektowania, wdrażania i użytkowania tego typu systemów nauczania. W sekcji tej przewidziano komunikaty prezentujące dorobek innych uczelni w tej dziedzinie, a zwłaszcza problemy konstrukcji, języków autorskich, grafiki komputerowej oraz wymienności i przenośności materiałów dydaktycznych.

Bliższych informacji udziela sekretarz naukowy Konferencji:

dr inż. Barbara Łukasik-Makowska,
Instytut Informatyki, Akademia Ekonomiczna,
ul. Komandorska 118/120, 53-345 Wrocław,
tel. 68-11-55 w. 376.

BASIC; jako potencjalni programiści są oni wypaczeni psychicznie bez nadziei na poprawę

- używanie COBOL-u paraliżuje umysł, dlatego jego nauczanie należy uznać za przestępstwo

- APL jest omyłką, doprowadzoną do perfekcji; jest to język przyszłości dla metod programowania obowiązujących w przyszłości; tworzy on nową generację „kodujących próżniaków"

- problemy zarządzania przedsiębiorstwem w ogólności, a zarządzania bazą danych w szczególności, są o wiele za trudne dla osób myślących programistycznie niedbałą angielszczyzną

- o użyciu języka: nie można zaostrzyć ołówka tępą siekierą, równie daremne jest używanie do tego dziesięciu tępych siekier

- prawdziwym skarbem kompetentnego programisty jest oprócz skłonności matematycznych, wyjątkowo dobre opanowanie języka ojczystego

- przedsiębiorstwa, które uzależniły się od sprzętu pojedynczego producenta (i postępując tak zaprzedały swe dusze diabłu!) po prostu upadną pod ciężarem szczególnej złożoności swych systemów informatycznych

- nie można oprzeć ani dyscypliny naukowej, ani też prężnego zawodu na błędach Departamentu Obrony, a zwłaszcza pewnego producenta komputerów

- używanie terminologii antropomorficznej w działalności związanej z systemami komputerowymi jest objawem niedojrzałości zawodowej

- głosząc, że mogą coś wnieść do inżynierii oprogramowania, przedstawiciele innych kierunków specjalizacji naukowej w dziedzinie programowania, stają się jeszcze bardziej śmieszni (niestety, nie mniej niebezpieczni!); mimo swej nazwy, inżynieria oprogramowania jest oparta na niesłychanie ścisłej wiedzy

- w dawnych dobrych czasach fizycy powtarzali wzajemnie swoje eksperymenty; obecnie przywiązali się oni do FORTRANU tak, że wciąż używają wspólnie tych samych programów z błędnymi włącznie

- projekty popierające programowanie w „języku naturalnym" są z natury skazane na niepowodzenie.

*

Czy powyższa lista nie jest dostatecznie niepokojąca? Co zamierzamy czynić? Przypuszczalnie, przejść do porządku dziennego...

Prof. dr EDSGER W. DIJKSTRA
doradca naukowy firmy
BURROUGHS

P.S. Jeżeli słuszne jest moje domniemanie, że przesłanie powyższych stwierdzeń zakłóci Wasz spokój, to proszę to dopisać również do listy niewygodnych prawd.

GIEŁDA INFORMACJI

W Giełdzie Informacji

publikujemy bezpłatnie:

- **krótkie prezentacje nowych, nie wykorzystanych dostatecznie systemów**
- **doniesienia o innowacjach programowych**
- **ogłoszenia dotyczące nowych rozwiązań technicznych sprzętu informatycznego**
- **indywidualne oferty informatyków poszukujących pracy oraz informacje o wolnych etatach**
- **ogłoszenia w sprawie szkoleń informatycznych**
- **inne informacje ułatwiające doraźną organizację informatyki w Polsce.**

Adapter interfejsu MERA 100-JS EMC

W Rzeszowskim Przedsiębiorstwie Informatyki Przemysłu Budowlanego „ETOB" przy współpracy Zakładu Metrologii Elektrycznej i Elektronicznej Politechniki Rzeszowskiej opracowany został adapter interfejsu MERA 100-JS EMC. Adapter umożliwia pracę on-line minikomputera biurowego MERA 100 w kanale multipleksorowym komputerów Jednolitego Systemu. Zapewnia wymianę informacji między pamięciami operacyjnymi obu tych maszyn w dwie strony, co stwarza możliwość tworzenia układu dwumaszynowego. Z punktu widzenia maszyn Jednolitego Systemu — MERA 100 zaopatrzona w adapter odpowiada wymogom skandardowego urządzenia pracującego w trybie multipleksorowym.

W budowie adaptera wykorzystano pakiety NO, SC i SD monitoru EC 7076 do automatycznej obsługi sekwencji wybierania początkowego i sekwencji kończącej interfejsu. Sekwencja transmisji (podczas czytania i pisania) sterowana jest programem minikomputera MERA 100. Adapter dokonuje konwersji kodu MERY 100 (ASCII) na kod maszyn Jednolitego Systemu (EBCIDC) w obie strony.

Naturalną możliwością programowania transmisji na konwerter (MERA 100 plus adapter interfejsu JS) jest fizyczny poziom wejścia-wyjścia (makroinstrukcja EXCP). Dla ułatwienia opracowano szereg programów i podprogramów dla konkretnych zastosowań i języków programowania.

Zastosowaniem, które zrodziło potrzebę wykonania adaptera była konieczność konwersji zbiorów z nośników używanych na minikomputerze MERA 100 (taśma kasetowa, dyskietka) do komputera R-32 i na odwrót.

GIEŁDA INFORMACJI

Interesujące jest zastosowanie konwertera jako sterownika linii w systemie teleprzetwarzania poprzez pakiet NOL. Dalsze zastosowania zależą jedynie od inwencji projektanta mającego do dyspozycji w ten sposób utworzony system dwumaszynowy.

Prace wdrożeniowe i uruchomieniowe ukończono 30 kwietnia 1982 r. Dotychczas wykonano dziewięć sztuk urządzeń — na potrzeby własne oraz na zamówienie ETOB KATOWICE i NBP.

Dodatkowych informacji udziela mgr inż. Józef Kamycki, ETOB, ul. Mjr Sucharskiego 2, 35-225 Rzeszów, tel. 348-65.

DMES — wieloprogramowy

ZETO Wrocław oferuje nowy system testowy, wchodzący w skład oprogramowania technicznego komputera R-32. Zalety systemu DMES:

— skrócenie efektywnego czasu testowania zestawu

— skrócenie czasu konserwacji

— możliwość równoczesnego testowania urządzeń pracujących w różnych kanałach R-32

— poprawa organizacji konserwacji

— możliwość przetestowania większości urządzeń w skróconym czasie konserwacji

— prosta obsługa systemu.

Po złożeniu zamówienia na system ZETO Wrocław zapewnia jego wdrożenie i uruchomienie u klienta oraz dostarcza pełny zestaw dokumentacji eksploatacyjnej.

Bliższych informacji o systemie udzielają: mgr inż. Jacek Teisseyre i mgr inż. Maciej Baranowski (ZETO Wrocław, ul. Ofiar Oświęcimskich 7-13, 50-069 Wrocław, tel. 44-54-31 w. 256). Zgłoszenia przyjmuje kierownik Działu Marketingu — mgr Tadeusz Czerniewski.

Materiały nadsyłane do GIEŁDY INFORMACJI będą publikowane w możliwie najkrótszym cyklu.

Kongres IFIP'83

Dziewiąty z kolei, cykliczny światowy Kongres Międzynarodowej Federacji Przetwarzania Informacji IFIP odbędzie się w roku przyszłym w dniach od 19 do 23 września w Paryżu. Po dziewięciu latach odbędzie się on znów w Europie (poprzednie dwa odbyły się w krajach pozaeuropejskich), a po prawie ćwierćwieczu powróci na miejsce swego powstania. Warto bowiem przypomnieć, że właśnie w Paryżu w 1959 r. zorganizowana z inicjatywy UNESCO pierwsza wielka międzynarodowa konferencja informatyczna podjęła uchwałę o utworzeniu IFIP, federacji zrzeszającej narodowe organizacje informatyków z różnych krajów. Formalne powstanie IFIP nastąpiło w 1960 r. na wniosek organizacji ośmiu krajów (w tym również Polski), natomiast kongresy odbywały się regularnie w cyklu trzyletnim, licząc od wspomnianej konferencji paryskiej w 1959 r. (1962 Monachium, 1965 Nowy Jork, 1968 Edynburg, 1971 Lublana, 1974 Sztokholm, 1977 Toronto, 1980 Tokio i Melbourn). Kongres przyszłoroczny połączony będzie jak zwykle z towarzyszącą mu ekspozycją sprzętu, który to warunek spełni — odbywająca się tradycyjnie we wrześniu, dobrze znana również polskim informatykom — doroczna wystawa SICOB. Na dzień dzisiejszy IFIP skupia organizacje informatyczne 41 krajów.

Zlokalizowanie kongresu IFIP w Paryżu pozwoli, miejmy nadzieję, na liczniejszą reprezentację uczestników z Polski (podobnie jak było to w przypadku kongresów w Lublanie i Sztokholmie), jakkolwiek poważną barierę w obecnej sytuacji mogą stanowić coraz wyższe kwoty opłat za uczestnictwo (przy wpłacie do 31.12.82 — 650 fr. szw.; do 30.06.83 — 750 fr. szw., na miejscu 800 fr. szw., a więc wg aktualnego kursu ok. 420 $).

Komitet Programowy Kongresu IFIP'83 ustalił już nazwę i problematykę konferencji, która ujęto w 10 dziedzin problemowych. Wystosowano również apel do wszystkich krajów członkowskich o nadsyłanie referatów (Call For Papers). Jeden z egzemplarzy apelu dotarł do naszej redakcji, co zgodnie z tradycją zobowiązuje nas do szczegółowego zaprezentowania jego treści na łamach czasopisma.

PROCEDURA PRZYGOTOWANIA, PRZESYŁANIA I AKCEPTACJI REFERATÓW

Referaty należy przesyłać tak, aby nie później niż 1 listopada 1982 dotarły pod adres przewodniczącego obszaru problemowego, którego dotyczy treść referatu. Przewodniczącymi poszczególnych 10 obszarów problemowych są następujący członkowie Komitetu Programowego Kongresu:

- Sprzęt i architektura komputerów:

  Dr. H. Schorr, Vice-President of Systems IBM T.S. Watson Research Center, P.O. Box. 218 Yorktown Heights, New York, 10598, USA

- Oprogramowanie komputerów:

  Professor D. Bjorner, Scientific Director Dansk Datamatik Center, Lundtofevej 1 C DK — 2800 Lyngby, DENMARK

- Podstawy teoretyczne przetwarzania informacji:

  Professor V. E. Kotov, Computing Center Novosibirsk, 630 090, USSR

- Sieci komputerowe i telekomunikacja:

  Professor W. M. Newman, Department of Computer Science and Statistics Queen Mary College, Mile End Road London E1 4NS, UNITED KINGDOM

- Baza danych i systemy informacyjne:

  Professor J. W. Schmidt, Fachbereich Informatik Universitaet Hamburg Schlueterstrasse 70, D-2000, Hamburg 13, FEDERAL REPUBLIC GERMANY

- Systemy zastosowań:

  Professor G. Capriz, Director, CNUCE Instituto del Consiglio Nazionale delle Ricerche, Via S. Maria 36 — 56 100 Pisa, ITALY

- Systemy informacyjne w biurach:

  Dr. N. Naffah, KAYAK, Projet Pilote INRIA — ADI, Domaine de Voluceau, Rocqencourt, B.P. 105, 78 153 Le Chesnay Cedex, FRANCE

- Zastosowanie mikroprocesorów:

  Professor R. Mori, Dean, College of Information Sciences University of Tsukuba, Ibarakiken 305, JAPAN

- Implikacje społeczne i ekonomiczne:

  Professor C. J. van Rijsbergen, Department of Computer Science, University College Dublin Belfield, Dublin 4, IRELAND

● Komputery w życiu codziennym:

Dr. T. Ohlin, Committee on New Information Technology
Dept. of Education and Culture
S-103 33 Stockholm, SWEDEN

Przesyłając maszynopis referatu w czterech egzemplarzach pod adresem właściwego przewodniczącego obszaru problemowego należy na kopercie wyraźnie zaznaczyć charakter przesyłki: Program Committee, IFIP Congress 83. Informacja o zaakceptowaniu lub odrzuceniu referatu zostanie przekazana autorom ok. 15 marca 1983 r., a wkrótce po tym nastąpi opublikowanie wstępnego programu konferencji. W przypadku referatów podpisanych przez wielu autorów, korespondencja będzie kierowana do osoby wymienionej na pierwszym miejscu. Ze względu na znaczny napływ referatów oraz stosunkowo krótki okres ich oceny, organizatorzy nie gwarantują możliwości zwrotu materiałów. Natychmiast po otrzymaniu informacji o zaakceptowaniu referatu, autorzy proszeni są o jego przepisanie (w sposób umożliwiający powielenie dla potrzeb publikacji). Szczegółowa instrukcja na ten temat oraz matryce dla wykonania maszynopisu (dającego możliwość bezpośredniego powielania) zostaną przesłane autorom przyjętych referatów. Przepisanie tekstu wraz z wykonaniem materiału ilustracyjnego powinno nastąpić w ciągu dwóch tygodni od chwili otrzymania instrukcji i matryc.

Komitet Programowy zastrzega sobie prawo dokonywania niewielkich zmian redakcyjnych względnie zobowiązania autorów do przeredagowania referatu zgodnie z wymaganiami wydawniczymi. Ocena referatów będzie dokonywana z punktu widzenia ich znaczenia, oryginalności oraz klarowności prezentacji. W celu udzielenia pomocy potencjalnym autorom opracowano broszurkę „Instructions for Authors", o przesłanie której zainteresowani mogą zwracać się listownie pod następującym adresem:

Program Committee, IFIP Congress 83, AFCET 156, Bld Péreire, F-75017 Paris, FRANCE.

Maszynopis referatu napisanego w języku angielskim powinien mieć następującą strukturę:

1. Stronę tytułową, obejmującą:
a) tytuł referatu

b) nazwisko, kraj, miejsce pracy oraz pełny pocztowy adres i telefon autora(ów)

c) jeden z dziesięciu obszarów problemowych, któremu odpowiada treść referatu

d) własnoręcznie podpisane następujące oświadczenie:

„Neither this paper nor any version close to it has been or is being offered elsewhere for publication and, if accepted, the paper will be personally presented at the 9th World Computer Congress by the author or one of the co-authors". („Referat ten ani jakiekolwiek zbliżone do niego wersje nie zostały nigdzie opublikowane lub przygotowane do publikacji. W przypadku akceptacji, referat zostanie osobiście wygłoszony przez autora lub jednego ze współautorów na 9 Światowym Kongresie Informatycznym").

2. Stronę zawierającą streszczenie o objętości ok. 100 słów

3. Tekst ostatecznej wersji referatu

4. Ilustracje z podpisami, ponumerowane kolejno i właściwie odniesione do tekstu. Uwaga: nie należy przesyłać oryginałów ilustracji lub fotografii przy referatach przesyłanych do akceptacji.

Każda strona powinna zawierać w lewym górnym rogu nazwisko autora, natomiast w prawym górnym rogu — kolejny numer strony, z tym, że streszczenie ma numer strony 2. Całkowita objętość referatu łącznie z ilustracjami i wykazem literatury nie może przekroczyć 4000 słów. Jedna ilustracja jest odpowiednikiem 250 słów tekstu. Strony każdego z egzemplarzy należy połączyć zszywkami.

WYKAZ OBSZARÓW PROBLEMOWYCH I TEMATYKI KONGRESU

1. Sprzęt i architektura komputerów (Computer Hardware and Architecture) — postęp technologiczny oraz jego wpływ na projekty systemów komputerowych

● architektura systemów (System architecture)

● procesory specjalizowane (Special purpose processors)

● technologia wejścia/wyjścia (Input/Output technology)

● niezawodność i tolerancja uszkodzeń (Reliability and fault tolerance)

● technologia układów i pamięci (Circuit and memory technology)

● pamięci o wielkiej pojemności (Large scale memories)

● automatyzacja projektowania układów LSI (LSI design automation)

● ocena architektury (Architecture assessment)

● pomiary wydajności (Performance measurement)

● architektura magistrali (Bus architecture)

● zagęszczanie elementów (Packaging)

2. Oprogramowanie komputerów (Computer Software) — metody i techniki ułatwiające opracowanie, eksploatację i ewolucję systemów oprogramowania

● techniki analizy potrzeb i ich definiowania (Requirements analysis and definition techniques)

● techniki specyfikowania architektury (Architecture specification techniques)

● metodologie programowania (Programming methodologies)

● programowanie współbieżne, funkcjonalne, aplikacyjne i logiczne (Parallel, functional, applicative and logic programming)

● lingwistyka i języki programowania (Programming linguistics and languages)

● niezawodność, jakość, adaptacyjność i przenośność programów (Program reliability, quality, adaptability and portability)

● systemy wspomagające programowanie (Programming support systems)

● oprogramowanie dla sztucznej inteligencji (Software for AI)

● wydajność systemów (Systems performance)

3. Podstawy teoretyczne przetwarzania informacji (Theoretical Foundations of Information Processing) — formalizacja koncepcji, metody matematyczne i teorie przetwarzania informacji

● teoria obliczeń, języki sformalizowane i automaty (Theory of computation, formal languages and automata)

● analiza formalna, synteza i transformacja programów (Formal analysis, synthesis and transformation of programs)

● semantyka koncepcji i języków programowania (Semantics of programming concepts and languages)

● teoria systemów i procesów (Theory of systems and processes)

● analiza i optymalizacja algorytmów (Analysis and optimization of algorithms)

● teoria danych i baz danych (Data and data base theory)

● aspekty formalne sztucznej inteligencji (Formal aspects of artificial intelligence)

4. Sieci komputerowe i telekomunikacyjne (Computer Networks and Communications) — użycie połączonych komputerów i ich wpływ na możliwości telekomunikacji

● przetwarzanie rozłożone, sprzęt i oprogramowanie (Distributed processing, hardware and software)

● techniki telekomunikacyjne: technologia i doświadczenia (Communications techniques: technology and experience)

● powiązanie systemu otwartego: normy i protokoły (Open-system interconnection: standards and protocols)

● systemy teletransmisyjne dla biur: lokalne i zdalne (Transmission systems for the office: local and longhaul)

● użycie minikomputerów i mikroprocesorów w sieciach (Use of minicomputers and microprocessors in networks)

● zastosowanie sieci i podział zasobów (Applications of networks and resource sharing)

- sieci lokalne (Local networks)
- sieci danych cyfrowych (Digital data networks)

5. Baza danych i systemy informacyjne (Data base and Information Systems) — modele, techniki i metodologie projektowania, wdrażania i użytkowania systemów danych

- specyfikacja potrzeb informacyjnych (Information requirements specification)
- narzędzia do projektowania danych (Data design aids)
- modele i języki danych (Data models and data languages)
- sprzęgi użytkownika obejmujące język naturalny (User interfaces including natural language)
- konwersja baz danych i programów (Data base and program conversion)
- inżynieria systemów baz danych (Data base systems engineering)
- rozłożona baza danych i systemy przesyłania danych (Distributed data base and data communication systems)
- maszyny baz danych (Data base machines)
- zastosowania baz danych (Data base applications)
- wyszukiwanie informacji (Information retrieval)

6. Systemy zastosowań (Application Systems) — problemy praktyczne i teoretyczne zastosowań w nauce, przemyśle, handlu i administracji publicznej

- komputeryzacja, modelowanie i projektowanie (Computerization, modelling and design)
- normalizacja (Standarization)
- zarządzanie i kontrola działalności (Management and auditing)
- szacowanie wydajności (Performance evaluation)
- niezawodność i bezpieczeństwo (Reliability and security)
- robotyka i automatyzacja przemysłowa (Roboties and industrial automation)
- współdziałanie z użytkownikami, obejmujące sprzęgi specjalne (Interaction with users, including special interfaces)
- problemy numeryczne i manipulowania symbolami (Numerical and symbol manipulation problems)

7. Systemy informacyjne w biurach (Office Information Systems) — zastosowanie komputerów, przetwarzanie informacji i systemów telekomunikacyjnych w biurach

- projektowanie stanowiska pracy w biurze: sprzęt, oprogramowanie i sprzęg użytkownika (Office workstation design: hardware, software and user interface)
- manipulowanie tekstem, głosem i obrazami (Manipulating text, voice and images)
- przechowywanie i wyszukiwanie informacji w biurze (Office information storage and retrieval)
- grafika interakcyjna i obrazy generowane komputerem (Interactive graphics and computer — generated images)
- metodologia instalowania i oceny systemów biurowych (Methodology for installation and evaluation of office systems)
- baza wiedzy w biurze (Knowledge base in the office)
- systemy przekazywania wiadomości oparte na komputerze oraz telekonferencje w biurze (Computer-based message systems and teleconferencing in the office)
- zintegrowane systemy zdalnego przesyłania głosu, obrazu i tekstu (Integrated, voice, image and text communication systems)
- modelowanie pracy biura (Office modelling)
- efekty organizacji i stylu pracy biura (Effects on office organization and workstyles)

8. Zastosowania mikroprocesorów (Microprocessor Applications) — zastosowanie małych systemów i mikroprocesorów w organizacjach i systemach specjalizowanych

- systemy mikroprocesorowe VLSI (VLSI microprocessor systems)
- oprogramowanie konwencjonalne i układowe mikroprocesorów (Microprocessor software and firmware)
- systemy operacyjne w kostce i procesory języka (On-chip operating systems and language processors)
- sieci mikroprocesorowe (Networks of microprocessors)
- zastosowanie mikroprocesorów w handlu, przemyśle i administracji publicznej (Microprocessor applications in business, industry and gouvernment)
- zastosowanie mikrokomputerów w dydaktyce i medycynie (Microcomputer applications in education and medicine)
- mikroprocesory w transporcie (Microprocessors in transportation)
- mikroprocesory w komunikacji głosowej człowiek—maszyna (Microprocessors in man—machine voice communication)
- komputery osobiste (Personal computers)
- standaryzacja systemów mikroprocesorowych (Standarization for microprocessor systems)

9. Implikacje społeczne i ekonomiczne (Social and Economic Implications) — wpływ komputerów i automatyzacji obliczeń na społeczeństwo i sposób rozumowania

- komputery i zatrudnienie (Computers and employment)
- informatyka jako zawód i przemysł informatyczny (Computing as a profession and the computing industry)
- komputery w dydaktyce i nauczaniu informatyki (Computers in education and computing education)
- komputery i prawo do tajemnicy osobistej (Computers and privacy)
- komputery i człowiek (Computers and man)
- implikacje filozoficzne (Philosophical implications)
- komputery i pieniądz (Computers and money (EFTS))
- komputery i rozwój społeczny (Computers and national development)
- prawa autorskie i patenty dla oprogramowania (Copyright and patents for software)

10. Komputery w życiu codziennym (Computers in Everyday Life) — zastosowania komputerów w domu, w sztuce, w formach mających wpływ na każdego człowieka

- komputery i czas wolny (Computers and leisure)
- komputery w domu (Computers in the home)
- komputery w stosunkach sąsiedzkich (Computers in the neighbourhood)
- interakcyjne i dostosowane do potrzeb osobistych środki komputerowe (Interactive and personalized computer media)
- videotekst domowy i teletekst radiowy (Home videotext and broadcast teletext)
- sztuki wizualne i przekształcające (The visual and performing arts)
- współuczestnictwo obywateli (Citizen participation)
- efekty psychologiczne i społeczne zastosowań domowych (Psychological and social effects of home applications)
- komputery jako osobiści pomocnicy (Computers as personal assistants).

Oprac. WŁADYSŁAW KLEPACZ

Macierzowy model gospodarki

Wadą różnych raportów o stanie gospodarki jest ich doraźność. Znany ekonometryk amerykański pochodzenia rosyjskiego, Wassily Leontief już w latach trzydziestych twierdził, że jedynym dosyć odpornym na upływ czasu modelem całości gospodarki narodowej mogą być tylko wymyślone przezeń macierze przepływów międzygałęziowych. Zwykłe statystyki odzwierciedlają działalność gospodarczą bardzo wycinkowo, natomiast macierze Leontiefa pozwalają określić ile *i*-ta gałąź dostarcza dóbr materialnych gałęzi *j*-otej. W szczególności więc można próbować określać proporcje popierania pozostałych działów, aby ustalony dział *k*-ty mógł przyspieszyć swój rozwój powiedzmy o 2% w stosunku rocznym.

Bez komputera można sporządzić macierz przepływów maksymalnie dla kilkunastu gałęzi. System gospodarczy w krajach kapitalistycznych wymaga — wydaje się — wyróżnienia przynajmniej kilkuset gałęzi, żeby z macierzy tego rodzaju można było wyciągać jakieś racjonalne wnioski. Ale chociaż komputery umożliwiły układanie macierzy kilkudziesięciogałęziowych, to jeszcze w latach pięćdziesiątych metoda Leontiefa była bardziej teorią niż praktyką. Anegdotka powiada, że 36-gałęziową macierz dla gospodarki brytyjskiej z początku lat pięćdziesiątych wyliczono „już" w połowie lat sześćdziesiątych. Ściślej mówiąc, obliczenia trwały znacznie krócej, ale głównie ekonometrycy nie mogli się zdecydować co uznać za gałąź.

Gałęzie w modelu macierzowym Leontiefa są określane dosyć umownie — i to właśnie jest główny problem przy mniejszych macierzach. Za miesięcznikiem *Scientific American* można podać, że Amerykanie korzystają z modelu 496-gałęziowego. Wprawdzie taką „maciorę" — jak mówi nasz krajowy entuzjasta modeli makro, dr Krzysztof Rey — można sobie wyobrazić rozwieszoną na jakiejś wielkiej ścianie, ale bez odpowiednich pomocniczych systemów komputerowych staje się ona wręcz nieporęczna. Współcześnie więc problem definiowania gałęzi uległ niejako odwróceniu: najpierw zestawia się olbrzymią macierz „surową", odpowiadającą szczegółowej sprawozdawczości (w krajach socjalistycznych) czy też spisom ludności i przemysłu (w krajach kapitalistycznych). Dopiero potem, drogą licznych próbnych analiz, dokonuje się stopniowej agregacji. Ideałem obliczeniowym jest stan, gdy każda z „wyagregowanych" w ten sposób gałęzi reprezentuje mniej więcej zbliżoną część dochodu narodowego — np. przy 97 gałęziach wyodrębnionych w wydanej niedawno przez wymieniony miesięcznik macierzy [1]) każda z nich reprezentuje w zerowym przybliżeniu ok. 20 miliardów dolarów dochodu narodowego rocznie.

Całość bogatego oprogramowania agregacyjnego zrealizowała uczelnia *New York University* w swym Biurze Analiz Ekonomicznych. Należałoby tu podkreślić, że źródłowy materiał statystyczny pochodzi jeszcze z 1972 roku, a agregacja prowadzona była w ten sposób, aby zestawiane macierze przepływów były przydatne jeszcze w latach osiemdziesiątych. W ten sposób w połowie lat siedemdziesiątych dysponowano wspomnianą macierzą 496-gałęziową, a w 1979 r. opublikowano dla szerszego użytku macierz 85-gałęziową — w dwumiesięczniku *Survey of Current Business*. Ta agregacja okazała ię jednak, jak można przypuszczać, zbyt „gruba", skoro ostatecznie wydano macierz 97-gałęziową, w dodatku wzbogaconą o 12 czynników jakościowych, głównie związanych z dewastacją środowiska naturalnego.

Omawiana macierz zasługuje na uwagę z wielu powodów:

Po pierwsze — można w niej wreszcie znaleźć wyodrębnioo i n f o r m a t y k ę, rozumianą zresztą szeroko jako „maszyny biurowe i sprzęt porkewny". W skali kraju (USA) gałąź ta zajmuje już 7 miejsce w tworzeniu sprzętu trwałego użytku 9 — w eksporcie oraz 11 — w zakupach federalnych na cele cywilne. Sama reprezentuje zaś zaledwie 7 promili dochodu narodowego brutto. Z przyjemnością więc można odebrać fakt, że są tacy ekonomiści, którzy równie cenią produkowanie sprzętu, wytwarzanie oprogramowania jak i sprzedaż czasu komputerowego. Niestety, w naszym kraju takiej jasności myślenia można tylko zazdrościć.

Po drugie — agregację przeprowadzono nowo zaproponowaną metodą triangulacyjną. Polega ona na takim ponumerowaniu poszczególnych gałęzi, aby gałęzie najbardziej służebne dla całości gospodarki jak też społeczeństwa, miały możliwie niskie numery. Informatyka została opatrzona już numerem 14, z dalekim wyprzedzeniem kopalin i różnych metalurgii. Mimochodem warto się żachnąć, że jeszcze w wielu krajach przy tworzeniu macierzy przepływów sprawa kolejności gałęzi jest w ogóle przypadkowa — zazwyczaj bowiem stosuje się numerację mechanicznie przejętą z roczników statystycznych. Niestety, brak na razie bliższych szczegółów o „pakiecie triangulacyjnym", a jest to zapewne oprogramowanie zasługujące na uwagę jako wielokryteriowe, w którym do ostatecznych rozwiązań — tj. różnych wariantów triangulacji — dochodzi się drogą interaktywnych prób i porównań.

Po trzecie — omawiane wydawnictwo jest osobliwością poligraficzną. Zostało przygotowane metodą fotoskładu w trzech krojach cyfr — nie licząc nagłówków — na unikalnej aparaturze firmy DONNELEY z Chicago. Wprawdzie nie obyło się przy tym bez dzielenia wielkiej macierzy na części i montażu technicznego, ale w efekcie jest to chyba rekord światowy co do wielkości planszy składanej tym sposobem, a jednocześnie wręcz ułatwiającej drobiazgową korektę. W naszych warunkach wydrukowanie tablicy absolutnie bezbłędnej należy jeszcze do kategorii mrzonek.

Osobną zupełnie sprawą jest analiza ponad 35 tys. danych liczbowych zawartych w macierzy. Analiza ta, będąca skądinąd wspaniałą kanwą do rozważań makroekonomicznych, może zanudzić na śmierć przeciętnego czytelnika — mimo zastosowanej przez wydawców amerykańskich przejrzystej metody kolorowych wyróżnień. To są jednak rozważania szczegółowe, dobre dla rekinów poszczególnych branż czy działaczy gospodarczych szczebla stanowego lub federalnego, interesujących się celowością inwestowania w określone gałęzie gospodarcze określane „z dokładnością do 1/100 dochodu narodowego". Tak bowiem można w uproszczenie traktować wyodrębnienie 97 gałęzi w omawianym modelu.

Nie ulega natomiast wątpliwości, że informatycy działający na niwie ekonomicznej w naszym kraju byliby nad wyraz szczęśliwi, gdyby mogli — prócz zwerbalizowanych raportów o stanie gospodarki — zapoznawać się także z wartościami konkretnych współczynników naszej macierzy. Nie mówiąc już o możliwości przeprowadzania własnych eksperymentów triangulacyjnych.

ADAM B. EMPACHER

---

[1]) The INPUT/OUTPUT Structure of the United States Economy; prepared by the editors of *Scientific American*, New York 1981, with the cooperation of Wassily W. Leontief and Edward N. Wolf. Plansza zwijana 1650—1330 mm + broszura 11 str.

Terminologia języka ADA (dokończenie)

W serii artykułów dotyczących języka ADA, publikowanych w INFORMATYCE od numeru 11-12/1981, jak również w towarzyszących im notatkach terminologicznych, brakuje wyjaśnienia, jak rozumie się program w tym języku. Poniżej postaramy się omówić nieco dokładniej to ważne zagadnienie.

Trudność polega na tym, że w przeciwieństwie do innych znanych języków wysokiego poziomu, jak np. PASCAL, w języku ADA słowo program nie jest słowem zastrzeżonym. Co więcej, nie występuje także w formalnej definicji języka. Nie zmienia to faktu, że w języku ADA wyrażamy programy komputerowe podobnie jak w innych językach.

Czytając uważnie podręcznik języka ADA (*Reference Manual for the Ada Programming Language, Washington, 1980*) znajdziemy następujące określenie programu w tym języku: *program jest zbiorem jednej lub wielu jednostek kompilacji dostarczonych kompilatorowi w czasie jednej lub wielu kompilacji.* Dalej wyjaśnia się, że jednostką kompilacji jest jednostka programowa poprzedzona tzw. specyfikacją kontekstu (ang. *context specification*), określającą inne jednostki kompilacji, od których zależy dana jednostka. Jednostkami programowymi są deklaracje i ciała pakietów i podprogramów.

Kompilacja w języku ADA oznacza następstwo (ang. *succession*) jednej lub wielu jednostek kompilacji.

Z podanych określeń wynika, że najprostszy program w języku składa się z pojedynczej jednostki kompilacji. Przykład bardziej złożonego programu podano w poprzednim numerze INFORMATYKI.

Choć w formalnej definicji języka ADA jednostki kompilacji określono nieco dokładniej, wydaje się, że na nasze potrzeby podane tu określenia są wystarczające.

Z terminologicznego punktu widzenia należy natomiast rozstrzygnąć pozostałe jeszcze wątpliwości. Nie kwestionując zasadności określenia programu w języku ADA, wypada stwierdzić, że druga część tego określenia — będąca opisem jednostek kompilacji *dostarczonych kompilatorowi w czasie jednej lub wielu kompilacji* — jest zbędna.

Według normy ISO 2382 programem komputerowym jest wykaz lub plan określający działania, które mogą być podjęte lub nie, wyrażony w postaci odpowiedniej do wykonania przez komputer. Jest więc jasne, że program komputerowy nie musi być wykonany, a tym bardziej przetłumaczony, aby mógł być nazywany programem. W odniesieniu do języka ADA wystarczy zatem powiedzieć, że program w tym języku jest zbiorem jednej lub wielu jednostek kompilacji oraz wyjaśnić, czym jest jednostka kompilacji.

Nawiasem mówiąc — definicja zawarta w normie ISO byłaby bardziej komunikatywna, gdyby *wykaz lub plan określający działania, które mogą być podjęte lub nie* nazwać algorytmem.

Analizując jeszcze dokładniej tę definicję można zauważyć, że nie wynika z niej konieczność wyrażania programów w językach programowania. Według innego określenia podanego w tej samej normie język programowania jest językiem sztucznym przeznaczonym do wyrażania programów komputerowych. Można więc domyśleć się, że język programowania jest tylko jednym ze środków wyrażania programów.

W świetle tych definicji określenie podane w normie PN-71/T-01016 — program jest sekwencją rozkazów umożliwiającą wykonanie przez komputer operacji niezbędnych do uzyskania z danych żądanego wyniku — należy uznać za znacznie mniej ogólne, choć nie błędne. Jest tak tym bardziej, że wyjaśnienie podane pod omawianym hasłem brzmi: *program zapisany jest w określonym języku programowania.* Końcowa część powyższej definicji dotycząca operacji *niezbędnych do uzyskania z danych żądanego wyniku* — jest także niezbyt potrzebnym jej uszczegółowieniem.

Na zakończenie podamy dokładne określenie jednostki kompilacji w notacji BCN

```
kompilacja : : = {jednostka kompilacji}
jednostka kompilacji : : =

    specyfikacja_kontekstu  deklaracja_podprogramu |
    specyfikacja_kontekstu  ciało_podprogramu |
    specyfikacja_kontekstu  deklaracja_pakietu |
    specyfikacja_kontekstu  ciało_pakietu |
    specyfikacja_kontekstu  podjednostka

specyfikacja_kontekstu : : = {klauzula_with [klauzula_use]}
podjednostka : : = separate (nazwa_jednostki)) ciało
```

Nawiasy klamrowe oznaczają powtórzenie elementu, nawiasy prostokątne — elementy nieobowiązkowe (ang. *optional*, a kreska pionowa oddziela elementy alternatywne.

Z powyższego zapisu wynika, że w skład kompilacji, a więc również programu, może nie wchodzić żadna jednostka (powtórzenie może być zerokrotne), co pozostaje w sprzeczności z określeniem słownym z tego samego podręcznika. Jest to typowy przykład niedokładności, z jakim można się spotkać w definicji języka.

W praktyce, przy braku jednostek kompilacji należy uważać, że w skład kompilacji wchodzi np. pragma tj. dyrektywa w języku ADA.

JANUSZ ZALEWSKI

O języku służącym do formułowania specyfikacji

Chciałbym podzielić się swoimi uwagami dotyczącymi artykułu Zygmunta Ryznara pt. „S&DL — język specyfikacyjny do projektowania strukturalnego (zarys propozycji)", zamieszczonego w INFORMATYCE nr 2-3/82. Sformułuję je z punktu widzenia języka PSL (Problem Statement Language), mimo że Autor na zakończenie artykułu mówi o „oczywistych różnicach" pomiędzy tymi językami.

Proponowany język S&DL sprawia wrażenie skomplikowanego połączenia (skądinąd dobrych i uznanych) idei zawartych w: systemie SEMS (ISDOS), języku ALGOL 68, języku PSL (ISDOS), językach DML.
Przejawia się to odpowiednio w następujących jego cechach:

- *możliwości określenia samego siebie*

Cecha ta komplikuje opis języka, powodując, że jest on bardziej teoretyczny (matematyczny), niż praktyczny. Sprawę tę, z reguły z lepszym skutkiem, rozwiązuje osobny język definicji języków, np. system SEMS opracowany w ramach projektu ISDOS lub klasyczne już generatory kompilatorów Mc'Keemana, Hornig'a i Wartman'a.

● *możliwości określenia meta-struktur lokalnych*

Jest to rozwiązanie ciekawe i potrzebne (stosowane np. w języku ALGOL 68), wymaga jednak pewnego zaawansowania w stosowaniu standardowych języków specyfikacji. Języki te muszą być ponadto łatwe do definiowania. Praktyka istniejących w tym zakresie systemów nie potwierdza jednak tego kierunku rozwoju języków specyfikacji.

● *możliwość określenia kilkudziesięciu typów obiektów, łączenie ich za pomocą kilkunastu typów relacji (funktorów) i system zdefiniowanych atrybutów.*

Idea ta nie jest przecież niczym innym niż językiem PSL z odwróconymi (nie wiadomo dlaczego) proporcjami ilościowymi. Jakkolwiek obowiązkowy szeroki zestaw cech-atrybutów jest uzasadniony praktycznie, to jednak duże wątpliwości budzi brak „symetrii" w definiowanych relacjach (funktorach). Być może wynika to z przyjętej technologii wytwarzania systemu informatycznego, ale przez to znacznie zmniejsza się zakres stosowalności tego języka.

● *możliwość zagnieżdżania definicji*

Cecha ta, aczkolwiek klarowna teoretycznie, stwarza szereg trudności dokumentacyjnych (konieczność uwzględniania kontekstów każdego obiektu!) i manipulacyjnych. Ponadto pogłębia i tak wysoki już stopień „teoretyczności" języka, oddalając jeden z jego głównych celów — zrozumiałość dla szerokiego kręgu ludzi, a więc użytkowników systemu. W kontekście tym wątpliwe są zalety tak pojętej strukturalizacji opisu.

Z artykułu nie wynika natomiast, czy proponowany język daje możliwość automatycznej analizy i dokumentowania projektu. Dla porównania — w systemie PSL/PSA dopracowano już pewne „standardy", które w połączeniu z generatorem dokumentacji stwarzają taką możliwość.

Ciekawość wzbudza również nie opisany w artykule sposób generowania programów, zwłaszcza że i w tym zakresie mamy już pewne doświadczenia związane z systemami PSL/PSA i DMS2.

Ze strony formalnej przedstawiona propozycja opisu (definicji) języka S&DL daleka jest od precyzyjności, spójności i zupełności. Skomplikowana i niekonwencjonalna symbolika oraz abstrakcyjne przykłady utrudniają pełne zrozumienie prezentowanych idei.

Z tych też względów proponowałbym zwrócenie szerszej uwagi na system PSL/PSA, który — mimo braku możliwości określania metastruktur — ma zasadniczą przewagę nad S&DL polegającą na tym, że analizator opisu systemu jest już dawno wdrożony na polskim sprzęcie.

Na zakończenie drobna uwaga terminologiczna dotycząca pojęcia *język specyfikacji* (np. systemów informatycznych). Termin ten, nie nowy w szerokiej już dzisiaj literaturze przedmiotu, oddaje — moim zdaniem — pełniej sens pojęcia *język służący do formułowania specyfikacji* niż zastosowany przez Autora termin *język specyfikacyjny* brzmiący zupełnie tak samo jak *dodatek brygadzistowski* występujący w aktualnym projekcie reformy systemu płac („Ile? Komu? Za co?", wrzesień 1982, str. 18 punkt 2b).

Zresztą, zdania mogą być tu podzielone, gdyż w praktycznym użyciu jest zarówno termin *języki konwersacji*, dostępne w konkretnym systemie komputerowym, jak i bardziej ogólny *języki konwersacyjne*.

JAN DAWIDOWSKI

Przedpłaty na 1983 r

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

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

Prenumerata ze zleceniem wysyłki za granicę jest dwukrotnie droższa.

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

CENTRUM
INFORMACJI NAUKOWEJ, TECHNICZNEJ I EKONOMICZNEJ

Al. Niepodległości 188, 00-950 Warszawa, skr. pocztowa 355
Telefon: 25-15-46 lub 25-12-41 w. 150
poleca do sprzedaży wysyłkowej następujące wydawnictwa:

1. **Przepisy o informacji naukowo-technicznej i informatyce. Autorzy: J. Góral, K. Hajduk-Popławska. CINTE Warszawa 1981, 527 s. Cena 138 zł.**
2. **Przetwarzanie danych nienumerycznych w komputerach typu SIMD. Autorzy: M. Maruszkiewicz, Z. Nowicki, H. Rybiński, Prace IINTE 1980, nr 30, 36 s. Cena 17 zł.**
3. **Wybrane zagadnienia zastosowania informatyki w procesach informacyjnych. Autorzy: M. Muraszkiewicz, Z. Nowicki, H. Rybiński. Prace IINTE 1979, nr 27, 159 s. Cena 58 zł.**
4. **Model automatyzacji niektórych funkcji ośrodka INTE. Autorzy: R. Faber, M. Muraszkiewicz, Z. Nowicki. Prace IINTE 1978, nr 13, 38 s. Cena 16 zł.**
5. **Wprowadzenie do teorii języków informacyjnych. Autorzy: A. Bielicka, E. Scibor. Materiały szkoleniowe 1981. nr 21, 84 s. Cena 33 zł.**
6. **Wprowadzenie do relacyjnego modelu danych i możliwości jego zastosowania w informacji naukowej. Autorzy: J. Dobosz, B. Szymański, A. Wojewoda. Prace IINTE 1981, nr 38, 96 s. Cena 68 zł.**
7. **Informacja normalizacyjna metrologiczna i dotycząca jakości. Autor: S. Mirowski. Materiały Szkoleniowe 1981. nr 22, 47 s. Cena 23 zł.**
8. **Wykonywanie i wykorzystanie mikroform w dokumentacji technicznej. Autor: A. Kaszuba: Materiały szkoleniowe 1979, nr 10, 130 s. Cena 53 zł.**
9. **Informacja patentowa. Autor: E. Kwapisz. Materiały Szkoleniowe nr 19, 95 s. Cena 40 zł.**
10. **Informator Nauki Polskiej. Praca zbiorowa. CINTE Warszawa 1981, XX edycja, 1106 s. Cena 480 zł.**

UWAGA: realizacja zamówień następować będzie sukcesywnie wg kolejności wpływu, aż do wyczerpania zapasu magazynowego.

.....
(pieczątka i dokładny adres wysyłkowy zamawiającego)

ul. . . . . nr . .

.....
(miejscowość)

.....
(kod, poczta)

CENTRUM INTE
Dział
Rozpowszechniania Wydawnictw
Al. Niepodległości 188b
00-950 WARSZAWA,
skr. pocztowa 355

.....
(pieczęć zamawiającego)

dnia . . . . 1982 r.

ZAMÓWIENIE

Zamawiamy i prosimy o przesłanie pocztą (płatne: za zaliczeniem pocztowym, przelewem po otrzymaniu przesyłki wraz z rachunkiem)* następujących pozycji nieperiodycznych:

Ilość egz.		Cena zł
.....	Przepisy o informacji naukowo-technicznej i informatyce	138.—
.....	Przetwarzanie danych nienumerycznych w komputerach typu SIMD	17.—
.....	Wybrane zagadnienia zastosowania informatyki w procesach informacyjnych	58.—
.....	Model automatyzacji niektórych funkcji ośrodka INTE	16.—
.....	Wprowadzenie do teorii języków informacyjnych	33.—
.....	Wprowadzenie do relacyjnego modelu danych i możliwości jego zastosowania w informacji naukowej	68.—
.....	Informacja normalizacyjna metrologiczna i dotycząca jakości	23.—
.....	Wykonywanie i wykorzystanie mikroform w dokumentacji technicznej	53.—
.....	Informacja patentowa	40.—
.....	Informator Nauki Polskiej	480.—

Przesyłkę zobowiązujemy się odebrać po jej nadejściu oraz uregulować należność za zamówione pozycje.

.....
(Gł. księgowy)

.....
(Dyrektor — kierownik)

* — niepotrzebne skreślić

Bibliografia wydawnictw polskich z dziedziny informatyki

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

Organizacja maszyn cyfrowych. Konwencjonalny poziom maszynowy. Poziom mikroprogramowania. Poziom systemu operacyjnego maszyny. Poziom języka asemblera. Maszyny wielopoziomowe. Sugestie dotyczące dalszej lektury i wykaz literatury. Dodatki: Arytmetyka o skończonej precyzji i liczby dwójkowe. Liczby zmienno-pozycyjne. Algebra Boolea.
Książka ma służyć jako podręcznik do wstępnego wykładu z programowania w języku asemblera i organizacji maszyn cyfrowych. Książka jest zrozumiała dla studentów informatyki pierwszych lat studiów. Może być przydatna programistom i użytkownikom maszyn cyfrowych.

● **Od algebry połączeń do mikroprocesora — PELKAM. Tłum. wyd. niem. z 1977 r. WKiŁ, Warszawa 1980, s. 236, cena 38 zł**

Kody. Układy logiczne. Przerzutniki. Monowibrator. Liczniki. Układy czasowe. Pomiar czasu. Pomiar częstotliwości. Pomiar przebiegów periodycznych. Układy arytmetyczne. Technologie i rodzaje układów scalonych. Cyfrowe układy sterowania. Mikroprocesor — mikrokomputer.
Książka przeznaczona jest dla szerokiego kręgu czytelników zainteresowanych techniką cyfrową.

● **Elektroniczna technika obliczeniowa — SAPIECHA K. WNT, Warszawa 1981 r., s. 288, cena 37 zł**

Pojęcie algorytmu i jego reprezentacja. Zasady działania elektronicznych systemów liczących. Podstawy programowania elektronicznych systemów liczących. Programowanie w języku FORTRAN 1900. Podstawy użytkowania systemów liczących.
Dodatki.
Podręcznik przeznaczony jest dla studentów wydziału elektroniki wyższych uczelni technicznych.

● **Podstawy projektowania wielkich sieci teleinformatycznych. Tłum. wyd. z ros. z 1976 r. — ŻYMIERIN D. G., MAKSYMIENKO W. I. — red. WNT, Warszawa 1981, s. 294, cena 80 zł**

Podstawowe sposoby projektowania sieci teleinformatycznych. Wielodostępne ośrodki obliczeniowe (WOO) na bazie Jednolitego Systemu Elektronicznych Maszyn Cyfrowych (JSEMC). Transmisja danych w sieciach ośrodków obliczeniowych. Określanie podstawowych charakterystyk technicznych i lokalizacji sieci ośrodków obliczeniowych. Wyposażenie informacyjne. Oprogramowanie sieci ośrodków obliczeniowych. Zadania rozwiązywane przez sieć ośrodków obliczeniowych.
Książka przeznaczona jest dla projektantów i użytkowników systemów informatycznych.

● **Projektowanie struktur systemów operacyjnych — OLSZEWSKI J. WNT, Warszawa 1981, s. 120, cena 27 zł**

Programy, procesy i ich struktury. Wybór przykładów. Konstruowanie systemu operacyjnego.
Dodatek: składnia ukierunkowanej maszynowo wersji Pascala.
Książka przeznaczona jest dla programistów.

● **Zbiór zadań z informatyki — BAUER F., GNATZ R., HILL U. Tłum. wyd. niem. z 1975 r. i 1976 r. WNT, Warszawa 1981, s. 349, cena 70 zł**

Informacja i wiadomości. Podstawy programowania. Języki algorytmiczne zorientowane maszynowo. Sieci przełączające i sekwencyjne. Organizacja danych i procesów. Automaty i języki formalne. Semantyka języków algorytmicznych.
Dodatek: Metodyka programowania.
Książka przeznaczona jest dla studentów informatyki oraz programistów.

● **Organizacja przesyłania informacji w systemach cyfrowych — DUDEK Z. T., SOSNOWSKI J. PWN, Warszawa 1981, s. 289, cena 40 zł**

Cz. 1. Zasady przesyłania informacji w systemie cyfrowym; zagadnienie przesyłania informacji. Struktury sieci przesyłania informacji. Technika przesyłania informacji. Zasady przesyłania informacji w łączach. Dopasowanie trybu pracy nadajnika do trybu pracy odbiornika. Sterowanie przepływem informacji w systemie cyfrowym.
Cz. 2. Zasady wprowadzania i wyprowadzania informacji w systemach cyfrowych; klasyfikacja urządzeń wejścia — wyjścia. Urządzenia wprowadzania i wyprowadzania informacji w postaci cyfrowej. Wyprowadzanie informacji w postaci graficznej. Wprowadzanie informacji w postaci graficznej.
Cz. 3. Zasady organizacji współpracy urządzeń zewnętrznych z systemem komputerowym; układy wejścia-wyjścia maszyn cyfrowych. Programowanie przesyłania danych. Autonomiczne przesyłanie danych.
Książka przeznaczona jest przede wszystkim dla studentów specjalizujących się w cyfrowych systemach informatyki i automatyki oraz dla inżynierów projektantów tych systemów.

● **Wprowadzenie do metodologii projektowania banków danych technologicznych — GONTARCZYK T. PWN, Warszawa — Łódź 1981, s. 90, cena 30 zł**

Rozwój i zastosowanie metodologii projektowania systemów informacyjnych. Sformułowanie zadania, problemu w zakresie tworzenia banku danych technologicznych. Studium wstępne budowy danych technologicznych. Projektowanie banku danych technologicznych. Opracowanie szczegółowe projektu. Oprogramowanie i wdrożenie banku danych technologicznych.
Materiały przeznaczone są dla pracowników przedsiębiorstw przemysłowych, zwłaszcza technologów, zainteresowanych problematyką komputerowo wspomaganych metod projektowania.

● **Ekonomiczne problemy zautomatyzowanych systemów zarządzania — KISIELNICKI J. PWE, Warszawa 1981, s. 257, cena 50 zł „Informatyka w praktyce"**

Systemy informacyjne w jednostkach gospodarczych. Kierunki zastosowań zautomatyzowanych systemów zarządzania. Efektywność zastosowań i czynniki na nią wpływające. Ocena efektywności zautomatyzowanych systemów zarządzania za pomocą rachunku ekonomicznego. Elementy rachunku ekonomicznego i wybrane metody ich określania.
Książka przeznaczona jest dla projektantów i użytkowników zautomatyzowanych systemów zarządzania.

● **Metody informatyczne — KISIELNICKI J. PWE, Warszawa 1981, s. 294, cena 40 zł**

Badania i analiza systemów informacyjnych. Projektowanie danych wejściowych i kontrola. Projektowanie danych wyjściowych. Projektowanie elementów systemów informatycznych. Ulepszane metody tworzenia oprogramowania. Projektowanie całościowych systemów informatycznych. Projektowanie systemów informatycznych opartych na bazie danych. Ochrona danych w systemach informatycznych. Ocena systemów informatycznych.
Książka przeznaczona jest dla działaczy gospodarczych, projektantów informatycznych systemów zarządzania i studentów wyższych uczelni ekonomicznych.

● **Funkcje administratora bazy danych i administratora zastosowań — BRANDT A. Wyd. Zjednoczenie Informatyki. Centrum Projektowania i Zastosowań Informatyki, Warszawa 1981, s. 124, cena 82 zł**

„Problemy Informatyki"
Specyfika projektowania bazy danych. Administrator bazy danych. Administrator zastosowań. Obszary bezpośredniej współpracy administratora bazy danych i administratora zastosowań. Kierunki rozwoju funkcji administracyjnych w otoczeniu bazy danych.
Materiały przeznaczone są dla projektantów systemów informatycznych i kadry kierowniczej ośrodków obliczeniowych.

● **Wybrane metody oceny prac projektowych i programowych, Praca zbiorowa. Wyd. Zjednoczenia Informatyki. Centrum Projektowania i Zastosowań Informatyki, Warszawa 1981, s. 124, cena 82 zł**

„Problemy Informatyki"
Metoda oceny jakości prac projektowych. Wykorzystanie techniki PSL/PSA do oceny prac projektowych. Metody oceny prac programowych. Zalety i wady przedstawionych metod oceny oraz ich przydatność w warunkach polskich.
Materiały przeznaczone są dla kadry kierowniczej ośrodków obliczeniowych.

Isotimpex

ELEKTRONICZNA KASA REJESTRUJĄCA ELKA 90

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

Charakterystyka podstawowych możliwości funkcjonalnych:

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

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

- rejestry wyrobów określane przez wprowadzany symbol kodowy 76

Pojemność powyższych rejestrów – 8 cyfr.

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

Podstawowe operacje:

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

Tryb pracy ręcznej:

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

Tryb pracy automatycznej:

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

Czytnik magnetyczny:

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

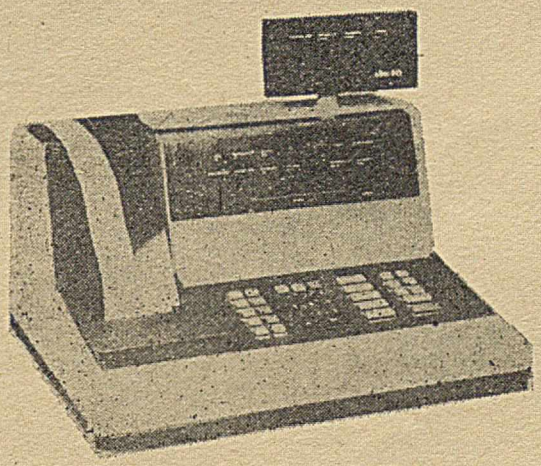
Dane wprowadzone prawidłowo są potwierdzane krótkim sygnałem dźwiękowym.

ELKA 90 ma 2 tryby operacji:

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

Po zarejestrowaniu każdej transakcji ELKA 90 podaje następujące dane:

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



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

EO'403/K/81-B

robotron

A 5130 jest modelem o największej wydajności.
Precyzyjnie reaguje na Wasze życzenia. Drukarka, ekran i klawiatura znajdują się stale w polu widzenia, co w zasadniczy sposób ułatwia pracę operatora. Informacje A 5130 są natychmiastowe, niezawodne i merytorycznie pewne, a ponadto uzyskiwane w trybie dialogu.
Przytoczmy niektóre z licznych dziedzin jego zastosowania: szczególne zalety zostały sprawdzone w transporcie, handlu, bankach, przemyśle, budownictwie, rolnictwie oraz w wielu instytucjach administracyjnych. Dlaczego? Ponieważ zastosowano go wszędzie tam, gdzie występują łącznie zadania rejestracji i przetwarzania danych, dla których niezbędne jest wydajne wyjście na drukarkę, a także tam, gdzie należy łączyć ekranową rejestrację danych z pracą na kontach magnetycznych lub gdzie pojawia się duże zapotrzebowanie na pamięć zewnętrzną.
System księgująco-obrachunkowy A 5130 jest w zastosowaniu szczególnie uniwersalny, do czego przyczyniają się istniejące problemowo ukierunkowane pakiety programów.
Proponujemy Wam zebranie szczegółowych informacji i dokładne obejrzenie systemu księgująco-obrachunkowego A 5130, a następnie przygotowanie miejsca dla jego zainstalowania.

Ma wszystko w swym polu widzenia: centralę i oddziały, plan, statystykę i bilans. Z rodziny maszyn A 5100

A 5130

robotron

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