P.1877/88

4 1988

informatyka

Prof. Griswold o przetwarzaniu napisów

Systemy ekspertowe na PC

Sieć UMMLAN-2

WYDAWNICTWO NOT SIGMA

Cena 200 zł

informatyka

4 1988

Nr 4
Miesięcznik Rok XXIII
Kwiecień 1988

Organ Komitetu
Naukowo-Technicznego NOT
ds. Informatyki

KOLEGIUM REDAKCYJNE:

Mgr Jarosław DEMINET,
dr inż. Wacław ISZKOWSKI,
mgr Teresa JABŁOŃSKA
(sekretarz redakcji),
Władysław KLEPACZ
(redaktor naczelny),
dr inż. Marek MACHURA,
dr inż. Wiktor RZECZKOWSKI,
mgr inż. Jan RYŻKO,
mgr Hanna WŁODARSKA,
dr inż. Janusz ZALEWSKI
(zastępca redaktora naczelnego).

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab. Juliusz Lech
KULIKOWSKI

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 01-517 Warszawa, ul. Mickiewicza 18 m. 17, tel. 39-14-34

Zakł. Graf. „Tamka". Zam. 0120-1300/88. Obj. 4,0 ark. druk. Nakład 8650 egz. U-14.

ISSN 0542-9951. INDEKS 36124

Cena egzemplarza 200 zł
Prenumerata roczna 2400 zł

WYDAWNICTWO
NACZELNA ORGANIZACJA TECHNICZNA
CZASOPISM I KSIĄŻEK TECHNICZNYCH
SIGMA
00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE: Strona

Języki wysokiego poziomu do przetwarzania napisów — COMIT, SNOBOL4 i Icon 1
Ralph E. Griswold, Madge T. Griswold 1

Narzędzia tworzenia systemów ekspertowych na mikrokomputerach
Jerzy Stawicki 3

Mikrokomputerowe systemy zarządzania
Zbyszko Królikowski 6

Architektura oprogramowania lokalnej sieci komputerowej UMMLAN-2
Krzysztof Zieliński, Jadwiga Indulska, Paweł Magura-Witkowski, Tomasz Walasek 9

Algorytmy kombinatoryczne i ich efektywność (2). Problem plecakowy
Maciej M. Sysło 13

Język Smalltalk-80 (2)
Artur Krępski 16

Język C. Coraz bliżej normy (2)
Jan Bielecki 19

Pamięci dyskowe Mazovii (2)
Andrzej Warda 21

ZE ŚWIATA 23

Co to jest MAP?

Podejście do tworzenia kompilatorów Ady

Akademicka sieć superkomputerowa

Kto jest kim

Z KRAJU 27

Mikrokomputery z Domu Handlowego Nauki

Baltcom'87

TERMINOLOGIA 31

Basic po polsku (1)

W NAJBLIŻSZYCH NUMERACH:

- Halina Ciechomska, Teresa Wójciekian i Ewa Zaborowska prezentują system zarządzania bazą danych dla mikrokomputerów klasy SM-4
- Michał Bartyś omawia podstawowe zasady łączenia i przekazywania parametrów między programami napisanymi w języku asemblera MASM i języku wysokiego poziomu MS-Pascal
- Krzysztof Dzik pisze o konfiguracji sprzętowej Mazovii 1016
- Jan Bielecki opisuje sposoby stosowania plików nagłówkowych i prototypów oraz deklarowanie funkcji w języku Turbo C
- Henryk Biesiada i Jerzy Grzywocz przedstawiają koncepcję uniwersalnego, modularnego systemu przetwarzania danych medycznych WSAD, z wykorzystaniem istniejącego oprogramowania komputera ODRA 1305

RALPH E. GRISWOLD
MADGE T. GRISWOLD
Department of Computer Science
Uniwersytet stanu Arizona, Tucson
USA

POLITECHNIKA ŚLĄSKA Biblioteka Główna
P. 1877/88

Języki wysokiego poziomu do przetwarzania napisów — COMIT, SNOBOL4 i Icon (I)

Wysokopoziomowe przetwarzanie napisów, w którym ciągi znaków są poddawane złożonej analizie i skomplikowanym przekształceniom, ma liczne zastosowania w technice obliczeniowej. Można tu wymienić: formatowanie dokumentów, analizę lingwistyczną, redagowanie danych, generowanie programów, sztuczną inteligencję, genetykę molekularną, kryptografię, lingwistykę matematyczną, metody translacji i przekształcanie symbolicznych wyrażeń matematycznych. Pierwsze języki przetwarzania ciągów znaków powstały już około 1958 roku, dojrzewały wraz z rozwojem zastosowań. Niniejszy artykuł poświęcono trzem spośród nich — językom: COMIT, SNOBOL4 i Icon.

Chociaż techniki obliczeniowe początkowo skupiały się głównie na rachunkach przeprowadzanych na liczbach — dla potrzeb nauki i handlu — nawet języki zaprojektowane do tych zastosowań (jak Fortran i Cobol) wymagały pewnych środków z zakresu przetwarzania napisów. W istocie, zapotrzebowanie na te środki występuje w większości spośród zastosowań komputerów. Kompilatory przyjmują na wejściu ciągi znaków i analizują je. Systemy operacyjne badają napisy kierowane do nich jako polecenia i wytwarzają komunikaty. Edytory działają niemal w zupełności na napisach. Kompilatory, systemy operacyjne i edytory są w częstym użytku, więc muszą być bardzo wydajne. Procesory tego rodzaju są zwykle pisane w językach niskiego poziomu. Pewne zagadnienia przetwarzania napisów są tak złożone, iż możliwość posłużenia się w ich rozwiązywaniu językami wysokiego poziomu jest nie tylko pożyteczna, ale wręcz niezbędna. Ponadto, podejmowane w językach wysokiego poziomu wstępne próby rozwiązań służą często wytwarzaniu prototypów, które są następnie podstawą do tworzenia programów w językach niskiego poziomu.

Jako przykładowe, z różnorodnych problemów przetwarzania napisów, rozważymy dekodowanie zaszyfrowanego komunikatu, wykonywanie symbolicznych przekształceń na wzorach matematycznych, formatowanie danych tekstowych zawierających instrukcje odnośnie postaci składu i ... fabrykowanie poezji. O ile kryptografia stała się wyrafinowaną wiedzą, w skład której wchodzą zaawansowane metody numeryczne, w technikach deszyfrowania ważny czynnik stanowi analiza lingwistyczna cech zaszyfrowanego tekstu odnoszona do jego napisowej reprezentacji (wydruk 1). Manipulowanie formułami symbolicznymi pociąga

Tekst zaszyfrowany

```
wh x srriseht xilog xeryl womks g
Icijrx milqx vevrviqrikwxe
```

Tekst odszyfrowany

```
do not send the packages until
they confirm the arrangements
```

Wydruk 1. Tekst zaszyfrowany i odszyfrowany

za sobą odkrywanie wzorców (ang. patterns) prowadzących do prostszych form (wydruk 2). W formatowaniu tekstu

Upraszczanie i manipulacja wzorami

```
(xA3)+3*xA2+x→x*(xA2+3*x+1
(xA2-1)/(x+1)→x-1
D(xA3+2*x,x)→3*xA2+2
```

Wydruk 2. Upraszczanie i manipulacja wzorami

trzeba podołać złożoności elementów składu, włączając w to wielkość krojów pisma, symbole specjalne i skomplikowany układ tekstu (wydruk 3). W wypadku poezji ge-

Dane wejściowe dla formatera tekstu

```
.PF
The \fIconcatenation\fR operation
sometimes called \*(oqcatenation\*(oq
appends one string to another to produce a
longer string. For example, the result of
concatenating the strings \fMCAT\fR and
\fMNIP\fR is the string \fMCATNIP\fR
```

Sformatowane wyniki

The *concatenation* operation sometimes called "catenation" appends one string to another to produce a longer string. For example, teh result of concatenating the strings CAT and NIP is the string CATNIP

Wydruk 3. Przykład formatowania tekstu

nerowanej przez komputer, zdolności twórcze poety są wzmagane przez właściwe komputerom możliwości wybierania alternatywnych struktur i form gramatycznych, i kojarzenie słów stosownie do ich atrybutów (wydruk 4).

Poezja wytworzona przez komputer

```
As he outlines a star
she stares, breathing slowly, tearfully
momentarily the boy lingers
then turns to damn the sky again
```

Wydruk 4. Przykład tekstu wygenerowanego przez komputer

Pierwsi projektanci języków do obliczeń numerycznych dysponowali dopracowanymi modelami opartymi na notacji matematycznej. Modele te były szeroko znane i powszechnie unormowane na długo przed tym, zanim jakiekolwiek przetwarzanie komputerowe doczekało się opracowania stosownych języków. Na tym tle przetwarzanie napisów okazało się dziedziną nową, w której nie było ani ogólnej zgody co do operacji, jakie mają być wykonywane, ani jakiegokolwiek standardowego zapisu. W rezultacie — notacje, struktury programowe i podejścia do precyzowania programu w językach przetwarzania napisów są skrajnie odmienne w stosunku do tych, które znamionują języki bardziej konwencjonalne.

DZIAŁANIA NA NAPISACH

Aby rozumieć języki wysokiego poziomu do przetwarzania tekstów, dobrze jest rozporządzać pewnym ogólnym pojęciem o działaniach wykonywanych na napisach. W ję-

Wersja oryginalna artykułu ukazała się w czasopiśmie Abacus, Vol. 3, No. 4, 1986. Tłumaczenia dokonano za zgodą wydawcy.

żykach przeznaczonych do przetwarzania napisów występują powszechnie cztery działania: konkatencja, czyli spinanie napisów, identyfikowanie podnapisów, tj. fragmentów napisów (ang. substrings), dopasowywanie wzorców i przekształcanie napisów (przez zastępowanie wykrytych fragmentów tekstu innymi napisami).

Działanie określone mianem **spinania** polega na dołączeniu jednego napisu do drugiego i otrzymaniu w wyniku napisu dłuższego. Rezultatem spięcia napisów CAT i NIP jest napis CATNIP. Działanie to wypływa w sposób naturalny z pojmowania napisu jako ciągu znaków. Podnapisem jest napis zawarty w innym napisie — AT jest podnapisem CATNIP.

Jednym z najważniejszych i najwszechstronniejszych działań napisowych jest **dopasowywanie wzorców**. Mówiąc ogólnie, proces dopasowywania wzorca polega na badaniu napisu pod kątem występowania w nim podnapisów, odznaczających się specjalnymi cechami. Przykładowo, może to być szukanie pewnego podnapisu na określonym miejscu w innym napisie. **Przekształcania napisów** dokonuje się zwykle w związku z procesem dopasowywania — uzyskane przez dopasowywanie wzorców wyniki służą do zastępowania pewnych podnapisów przez inne.

W następującym dalej przeglądzie języków przedstawiono ogólne koncepcje przetwarzania napisów i główne środki rozwiązywania problemów tego rodzaju. Nie próbowano opisywać któregokolwiek z tych języków w sposób zupełny. Czytelnicy zainteresowani pełnymi opisami tych języków odnajdą stosowne informacje w pracach przytoczonych na końcu artykułu.

COMIT

Język COMIT zaprojektowano w latach 1957—1958 jako odpowiedź na zapotrzebowanie na narzędzie do mechanicznego tłumaczenia języków naturalnych oraz do badań lingwistycznych. Jego twórcami była grupa pracowników MIT kierowana przez Victora Yngve. COMIT jako pierwszy szeroko używany język przetwarzania napisów, wywarł znaczący wpływ na następne języki tego rodzaju. Odzwierciedlając przyczynę swego powstania, COMIT jest ukierunkowany na reprezentowanie języków naturalnych, w których zdania są utworzone ze słów.

Podstawowe koncepcje

Odmiennie niż w większości języków przetwarzania napisów, w których napis tworzy ciąg znaków, napis w języku COMIT jest zbudowany ze **składowych** (ang. constituents), które mogą być utworzone z więcej niż jednego znaku. Tak więc, wieloznakowe słowo może być pojedynczą składową napisu. Sam napis jest reprezentowany jako ciąg składowych pooddzielanych znakami plus:

IT + WAS + A + DARK + —
AND + STORMY + NIGHT

Ponieważ do wprowadzania danych w języku COMIT są przewidziane karty, długie wiersze mogą być łamane przez zakończenie wiersza (karty) znakiem minus i kontynuowane w nowym wierszu, jak to zilustrowano powyżej.

Znaki, które nie są literami, mają w języku COMIT znaczenie składniowe. Gwiazdka poprzedzająca znak nie będący literą znamionuje, iż w przeciwieństwie do przysługującej mu treści składniowej, znak ma zostać potraktowany literalnie. Na przykład, zdanie "24 LEFT EARLY." zapisuje się w postaci:

***2*4 + LEFT + EARLY + *.**

Gdyby zamiast *2*4 użyto 24, oznaczałoby to wskazanie pozycji składowej (patrz dalej), a nie składową przedstawiającą liczbę 24.

Napis aktualnie podlegający obróbce pozostaje w **obszarze roboczym**. Ponadto, istnieje 128 **półek**, z których każda może być w razie potrzeby wymienioną z obszarem roboczym. W konsekwencji, w tym samym czasie w programie może być co najwyżej 129 różnych napisów.

Program w języku COMIT jest utworzony z ciągu reguł. Każda reguła ma pięć części:

nazwa lewa-połowa = prawa-połowa // postępowanie skok Nazwa wyróżnia regułę. **Lewa-połowa** stanowi wzorzec, który odnosi się do obszaru roboczego. **Prawa-połowa** określa działanie, któremu ma być poddana część obszaru roboczego dopasowana do lewej połowy. **Postępowanie**, oddzielone od reszty reguły dwoma ukośnymi kreskami, dotyczy wykonania operacji innych niż dopasowywanie wzorców. Reguła, w której pole określające postępowanie nie znajduje zastosowania, nie wymaga stawiania ukośnych kresek. **Skok** nadzoruje przebieg sterowania w programie.

Dopasowywanie wzorców

Najistotniejszą cechą języka COMIT jest dopasowywanie wzorców, dokonywane w lewej połowie i transformacja obszaru roboczego, wykonywana przez prawą połowę. Lewa połowa może zadawać — na przykład — pełne składowe (tak jak to ma miejsce przy określaniu napisów), określoną liczbę składowych o nieznanych wartościach, dowolną liczbę składowych lub wcześniejszą składową identyfikowaną przez jej pozycję w lewej połowie. Na przykład, $n dopasowuje **n** składowych, a $ dopasowuje dowolną liczbę składowych. Liczba naturalna **n** oznacza dopasowanie takiego samego napisu, jaki zostanie dopasowany przez **n**-tą składową lewej połowy.

Przykładem lewej połowy jest ciąg:

READ + $1 +$1 + $ + 3

Tutaj lewa połowa jest złożona z pięciu składowych: składowej stanowionej przez słowo READ, po której występuje pojedyncza składowa, za nią inna pojedyncza składowa, dalej zaś dowolna liczba składowych [1]), aż do odnalezienia składowej identycznej z tą, która w procesie dopasowywania zostanie przyporządkowana trzeciej składowej wzorca. Dopasowywanie odbywa się od lewej strony do prawej. Składowe lewej połowy muszą dopasowywać kolejne składowe (napisu) w obszarze roboczym.

Przykładowo, dla poniższej zawartości obszaru roboczego dopasowanie poszczególnych składowych do wzorca będzie następujące:

READ + THE + FIRST + ONE + FIRST
1 2 3 4 5

Piąta składowa lewej połowy dopasowuje z obszaru roboczego składową o tej samej wartości, co składowa dopasowana do trzeciej składowej lewej połowy. W procesie dopasowywania następuje kojarzenie odpowiednich składowych obszaru roboczego ze składowymi lewej połowy. Po dopasowaniu każda ze składowych obszaru roboczego otrzymuje numer stosownej składowej lewej połowy.

Prawa połowa może zawierać składowe wyartykułowane w pełni i liczby naturalne, którym odpowiadają składowe z lewej połowy. Dopasowana porcja obszaru roboczego zostaje zastąpiona przez składowe określone w prawej połowie. W odniesieniu do poprzedniego przykładu, reguła:

READ + $1 + $1 + $ + 3 = —
1 + 2 + LAST + 4 + 5

przekształca obszar roboczy na postać:

READ + THE + LAST + ONE + FIRST

Składowe mogą posiadać atrybuty, nazywane **wskaźnikami** (ang. subscripts). Wskaźniki mogą być logiczne lub numeryczne. Wskaźniki oddziela się od składowej ukośną kreską, a same między sobą — przecinkami:

WOLFE/.38, SEX F, HOBBY GOLF

Kropka identyfikuje wskaźnik numeryczny, w danym przypadku o wartości 38. SEX i HOBBY stanowią wskaźniki logiczne, odpowiednio, o wartościach F i GOLF. Dopuszcza się wiele wskaźników logicznych, z których każdy może mieć dowolne wartości. Wskaźniki przyporządkowują składowym informacje, które z kolei można wykorzystywać przy dopasowywaniu wzorców do rozróżniania składowych o specyficznych atrybutach. W języku COMIT istnieje pewna liczba operacji do manipulowania wskaźnikami.

Inne właściwości

Część reguły nazywana **postępowaniem** dopuszcza operacje, które nie mogą być wykonywane w prawej połowie. Są to między innymi: wymiana obszaru roboczego z półką, przemieszczanie składowych między obszarem roboczym a półkami, drukowanie obszaru roboczego, wczytywa-

[1]) Dowolna liczba składowych — w sensie dopasowywania napisu do danego wzorca (przyp. tłum.).

dokończenie na s. 5

JERZY STAWICKI
Instytut Organizacji Przemysłu Maszynowego
Warszawa

Narzędzia tworzenia systemów ekspertowych na mikrokomputerach

Sztuczna inteligencja i systemy ekspertowe stają się dostępne także na mikrokomputerach. Fakt ten potwierdza pojawienie się na rynku oprogramowania szeregu pakietów wspomagających budowanie systemów ekspertowych (ang. expert system shells). Celem niniejszego artykułu jest przedstawienie trzech takich pakietów, działających na mikrokomputerach klasy IBM PC XT/AT, tj. **pakietów M.1, EXSYS** oraz **ESIE.** Artykuł został opracowany na podstawie doświadczeń autora zdobytych przy uruchamianiu tego oprogramowania, konsultacji z pokazowymi bazami wiedzy oraz wykorzystaniu pakietów do problemów zarządzania przedsiębiorstwem.

OGÓLNA CHARAKTERYSTYKA NARZĘDZI

Omawiane pakiety należą do klasy systemów regułowych [1, 2], w których wiedza jest reprezentowana w postaci reguł:

IF warunek THEN akcja1

lub

IF warunek THEN akcja1 ELSE akcja2

Wnioskowanie w zbiorze reguł może być realizowane jako:

- wnioskowanie do przodu (ang. forward chaining), tj. dochodzenie do konkluzji na podstawie posiadanych danych,
- wnioskowanie do tyłu (ang. backward chaining), tj. próba udowodnienia prawdziwości postawionej hipotezy przez odwołanie się do zbioru reguł zawartych w systemie.

Narzędzia tworzenia systemów ekspertowych zawierają podstawowe mechanizmy niezbędne w procesie wnioskowania. Zadanie projektanta nowego systemu ekspertowego polega na określeniu zbioru reguł tworzących bazę wiedzy danej dziedziny. Niektóre narzędzia są wyposażone w edytory reguł ułatwiające budowę bazy wiedzy (np. EXSYS), inne wczytują zbiór reguł w postaci zewnętrznego pliku znakowego (np. M.1 i ESIE). System ekspertowy rozwiązuje postawione zadanie na podstawie swojej bazy wiedzy, prowadząc dialog z użytkownikiem i zadając mu niezbędne pytania. Ponadto system może objaśniać tok swego rozumowania udzielając odpowiedzi na pytania: „dlaczego ?" (dlaczego zadał użytkownikowi określone pytanie) oraz „jak ?" (jak doszedł do danego wniosku).

Typowymi zadaniami rozwiązywanymi za pomocą narzędzi omówionych w tym artykule są: diagnozowanie i formułowanie zaleceń, planowanie oraz projektowanie [1].

Dr inż. JERZY STAWICKI ukończył studia w 1978 r. w Instytucie Organizacji Zarządzania Politechniki Warszawskiej. W 1987 r. uzyskał doktorat w Politechnice Wrocławskiej. Od 1981 r. pracuje w Instytucie Organizacji Przemysłu Maszynowego w Warszawie, gdzie zajmuje się informatycznymi systemami zarządzania przedsiębiorstwem oraz zastosowaniami metod sztucznej inteligencji w zarządzaniu.

SYSTEM M.1

Twórcą pakietu M.1 jest firma Teknowledge Inc. Pakiet został wprowadzony na rynek w czerwcu 1984 roku. Stanowi on narzędzie wspomagające inżynierów wiedzy przy budowie prototypowych systemów ekspertowych. Jego główne zadania to [7]: umożliwienie użytkownikom zapoznania się z podstawowymi zasadami i koncepcjami inżynierii wiedzy, pomoc w badaniu możliwości zastosowania systemów ekspertowych do rozwiązania konkretnych problemów oraz pomoc w opracowaniu małych systemów ekspertowych. Pozwala on na budowę systemu obejmującego 100—200 reguł [7]. Cena pakietu w USA wynosi 5000 dolarów, a łącznie z kilkudniowym szkoleniem użytkownika 12 500 dolarów [3].

Reprezentacja wiedzy i mechanizm wnioskowania

Fakty są reprezentowane za pomocą par atrybut-wartość wyposażonych ewentualnie we współczynniki pewności. Reguły mają typową postać IF-THEN lub IF-THEN-ELSE. Cechą charakterystyczną systemu M.1 są tzw. reguły zmienne, pozwalające — dzięki parametryzacji — na określenie wielu podobnych reguł za pomocą jednej reguły podstawowej. W systemie zastosowano mechanizm wnioskowania do tyłu, podobnie jak w systemie MYCIN [2].

Sterowanie procesem wnioskowania można osiągnąć dwojako:

- przez ustalenie kolejności występowania reguł,
- przez zastosowanie reguł o postaci „jeśli znaleziono", umożliwiających wnioskowanie do przodu.

Współpraca z użytkownikiem i implementacja

Oprócz typowych dla systemów regułowych mechanizmów współpracy z użytkownikiem, w pakiecie M.1 dostępne są również funkcje śledzące realizację wnioskowania (przydatne na etapie usuwania błędów z bazy wiedzy). Pakiet ułatwia użytkownikowi reakcję na zadane przez siebie pytania, wyświetlając propozycje możliwych odpowiedzi. M.1 może również ocenić i rozwinąć skrócone odpowiedzi użytkownika. Baza wiedzy jest tworzona poza systemem za pomocą uniwersalnego edytora tekstów. W systemie istnieją także programistyczne narzędzia do usuwania błędów z bazy wiedzy, tzn. system okien umożliwiający śledzenie realizacji programu. Obejmuje on następujące okna: zdarzenia, konkluzje, przebieg wnioskowania, dostępne opcje, menu odpowiedzi na pytania systemu w trakcie konsultacji.

M.1 został zrealizowany w Prologu. Jest dostępny na mikrokomputerach IBM PC XT/AT i wymaga minimum 384 KB pamięci. Firma zaleca użycie kolorowego monitora. Pakiet obejmuje dwie dyskietki. Jedna zawiera główny program i przykładowe bazy wiedzy, druga — zbiór programów umożliwiających współpracę systemu M.1 z językiem C.

ESIE

System ESIE jest rozpowszechniany przez firmę Lightwave Consultants. ESIE jest akronimem sformułowania Expert System Inference Engine (mechanizm wnioskowania systemu ekspertowego). Sposób reprezentacji wiedzy

i mechanizm wnioskowania są typowe dla systemów regułowych. Podstawowe zalety tego systemu to niski koszt zakupu (kwota według uznania użytkownika, gdyż produkt ten należy do kategorii tzw. shareware [4]), możliwość kopiowania oraz dostępu do nowszych wersji systemu, łatwość nauczenia się i użytkowania, dostępność postaci źródłowej (za dodatkową opłatą).

Reprezentacja wiedzy i mechanizm wnioskowania

System zawiera pięć typów reguł. Dwa typy reguł są wprowadzane do każdej bazy wiedzy tylko raz, jeden typ może być wprowadzony raz lub w ogóle, pozostałe dwa typy — dowolną liczbę razy (nie większą jednak niż wynika to z ograniczeń systemu). Reguły dostępne w ESIE są następujące:

1) [legalanswers is/are ⟨zmienna⟩ [⟨zmienna⟩]...*]
2) goal is ⟨zmienna⟩
3) [if ⟨zmienna⟩ is/are ⟨wartość⟩
[and ⟨zmienna⟩ is/are ⟨wartość⟩]... then
⟨zmienna⟩ is/are ⟨wartość⟩]...
4) [question ⟨zmienna⟩ is/are "⟨tekst⟩"]...
5) answer is/are "⟨tekst⟩" ⟨zmienna⟩

Reguła o postaci LEGALANSWERS służy do ograniczenia zbioru możliwych odpowiedzi na pytania stawiane przez system. Reguła o postaci GOAL określa cel konsultacji użytkownika, tj. poszukiwane przez niego rozwiązanie problemu. Reguły typu IF, stanowią zasadniczą część bazy wiedzy, opisując wiedzę z danej dziedziny. Reguły o postaci QUESTION są wywoływane przez ESIE wtedy, gdy w bazie wiedzy brakuje faktów niezbędnych do kontynuacji wnioskowania i konieczne jest ich wprowadzenie przez samego użytkownika. Reguły o postaci ANSWER służą do przekazywania użytkownikowi odpowiedzi uzyskanej przez system w wyniku wnioskowania.

Ograniczenia ilościowe systemu ESIE są następujące:

- 50 zmiennych dla LEGALANSWERS,
- jedna i tylko jedna reguła GOAL,
- 400 wierszy zawierających reguły IF,
- 100 reguł QUESTION,
- jedna i tylko jedna reguła ANSWER.

Współpraca z użytkownikiem i implementacja

Współpraca z użytkownikiem jest realizowana w sposób standardowy. Dodatkowo na najwyższym poziomie dostępne są następujące polecenia: TRACE ON/OFF, GO oraz EXIT. Polecenia TRACE ON/OFF umożliwiają włączenie lub wyłączenie śledzenia wnioskowania (pokazywane są aktualne cele systemu i wykorzystywane reguły). Umożliwia to stosunkowo łatwe usuwanie błędów z bazy wiedzy.

Pakiet ESIE jest napisany w Pascalu. Za dodatkową opłatą można nabyć wersję źródłową systemu. Zajmuje on jedną dyskietkę, na której zawarty jest sam pakiet oraz kilka przykładowych baz wiedzy, a także podręcznik użytkownika [4], z którego zaczerpnięto większość przytoczonych tutaj informacji.

Wymagania sprzętowo-programowe są następujące: mikrokomputer IBM PC z minimalną pamięcią 128 KB, system operacyjny DOS 2.0 (lub nowszy) oraz edytor tekstów (do tworzenia bazy wiedzy).

EXSYS

Pakiet EXSYS jest rozpowszechniany przez firmę EXSYS Inc. Jest to narzędzie tworzenia systemów ekspertowych, obejmujące edytor bazy wiedzy oraz główny program systemu ekspertowego, dokonujący wnioskowania na podstawie dostępnej bazy wiedzy. Istotną zaletą pakietu jest rozbudowany edytor, umożliwiający szybkie stworzenie bazy wiedzy. Edytor wykorzystuje technikę menu, wymuszając na użytkowniku kolejne kroki projektowania reguły. Umożliwia on także szybkie „podglądanie" już wykorzystywanych w systemie atrybutów (tzw. kwalifikatorów) oraz zmiennych i formuł. Pozwala także na szybkie usuwanie, przesuwanie lub modyfikowanie reguły (ciągu reguł).

Reprezentacja wiedzy i mechanizm wnioskowania

Wiedza jest reprezentowana w postaci reguł. Pierwszy człon reguły (tj. człon IF) składa się z tzw. kwalifikatora oraz jego wartości. Reguła ma następującą postać:
IF ... THEN ... ELSE ...
Ponadto w jej skład może wchodzić tzw. notatka, zawierająca dodatkowe komentarze związane z regułą oraz tzw. referencje, określające źródło informacji wykorzystywanych w regule. System ma możliwość uwzględniania prawdopodobieństw zdarzeń w części THEN i ELSE. Prawdopodobieństwa mogą być określane jako liczby z przedziału 0—10 lub z przedziału 0—100. Wykorzystywany jest zarówno mechanizm wnioskowania do przodu, jak i do tyłu.

Współpraca z użytkownikiem i implementacja

Współpraca systemu z użytkownikiem jest realizowana w dwóch etapach: projektowania bazy wiedzy i konsultacji z wykorzystaniem systemu. Projektowanie jest znacznie ułatwione dzięki edytorowi bazy wiedzy. Etap konsultacji jest realizowany w sposób typowy dla systemów regułowych; również typowe są dostępne mechanizmy „podglądania" procesu wnioskowania. Konsultacja kończy się podaniem wyników wnioskowania w postaci listy rozwiązań wraz z ich względnymi ocenami. Istnieje możliwość powtórzenia konsultacji po zmianie wartości niektórych danych lub parametrów. Łatwość współpracy z systemem podnosi znacznie wyświetlanie na ekranie wykazu dostępnych działań (zarówno przy pracy z edytorem, jak i w trakcie konsultacji) oraz możliwość korzystania z wyjaśnień systemu (tzw. samouczek, ang. help). Inne możliwości systemu to:

- dołączanie zewnętrznych programów, zarówno wejściowych jak i wyjściowych,
- przekazywanie zmiennych do programów zewnętrznych.

Pakiet EXSYS działa na mikrokomputerach IBM PC/XT i AT. Jest napisany w języku C. Wymaga minimum 192 KB pamięci operacyjnej. Cały system zajmuje cztery dyskietki: program główny, edytor bazy wiedzy, pięć lekcji oraz dyskietkę pokazową, zawierającą kilka systemów stworzonych przy pomocy pakietu EXSYS [5].

INNE NARZĘDZIA DOSTĘPNE NA IBM PC

W krajach zachodnich dostępnych jest wiele innych narzędzi umożliwiających tworzenie systemów ekspertowych na mikrokomputerach IBM PC. Ich zestawienia można znaleźć w [6] i [11]. Trzy spośród nich zasługują na krótkie omówienie.

GURU [8, 12] jest pakietem zintegrowanym, obejmującym oprócz tradycyjnych funkcji przetwarzania tekstów, bazy danych, arkusza obliczeniowego i grafiki — generator systemów ekspertowych, mający dostęp zarówno do bazy danych, jak i poszczególnych arkuszy. Firma MDBS oferująca pakiet GURU reklamuje go jako pierwsze zastosowanie sztucznej inteligencji do celów businessu i zarządzania.

FLOPS (Fuzzy Logic Production System) [10] stanowi rozwinięcie systemu regułowego OPS5. Różni się on od innych narzędzi następującymi cechami:

- uwzględnieniem niepewności zarówno w zakresie faktów, jak i liczb (FLOPS ma m.in. takie struktury danych jak zbiory rozmyte i liczby rozmyte),
- wykorzystaniem nie-arystotelesowskiej logiki rozmytej (ang. fuzzy logic), w której złożone zdania mogą być prawdziwe w różnym stopniu,
- zastosowaniem złożonego mechanizmu nawracania, umożliwiającego przeszukiwanie różnych ścieżek rozwiązań przy jednoczesnym pamiętaniu poprzednio ustalonych faktów.

Inne specyficzne cechy systemu FLOPS to możliwość łączenia z programami napisanymi w innych językach (głównie C), możliwość pisania reguł generujących nowe reguły oraz istnienie współbieżnej wersji FLOPS, emulującej procesory współbieżne.

GoldWorks [9] jest oferowany przez firmę Gold Hill na komputery wykorzystujące procesory 80286 i 80386. Pakiet obejmuje podsystem współpracy z użytkownikiem przy wykorzystaniu techniki menu, podsystem współpracy z językiem implementacji (Golden Common LISP) oraz zapewnia możliwość integracji systemu ekspertowego

z dBASE III, Lotus 1-2-3 oraz programami i bibliotekami języka C. GoldWorks oferuje trzy metody reprezentacji wiedzy: ramy, reguły oraz obiekty i programowanie obiektowe (ang. object oriented programming).

* * *

Narzędzia tworzenia systemów ekspertowych, takie jak opisane w niniejszym artykule, umożliwiają stosunkowo łatwe, szybkie i tanie opracowywanie prostych systemów ekspertowych. Szczegółowe wymagania stawiane konkretnemu narzędziu powinny wynikać z klasy i właściwości rozwiązywanego problemu.

Listę charakterystyk narzędzi tworzenia systemów ekspertowych, w odniesieniu do trzech opisanych pakietów, przedstawiono w poniższej tabeli. Jak wynika z tego zestawienia, systemy EXSYS i M.1 są znacznie bardziej rozbudowane i mają znacznie więcej możliwości od systemu ESIE. Spośród tych dwóch, łatwiejszy w użytkowaniu — dzięki własnemu edytorowi bazy wiedzy, jak i wielu funkcjom dostępnym bezpośrednio w postaci menu — jest EXSYS. Dodatkowo dysponuje on bogatszymi mechanizmami wnioskowania i umożliwia tworzenie większych (pod względem liczby reguł) baz wiedzy. Brak dokumentacji uniemożliwia dokonanie pełnego opisu możliwości M.1, jednak według oceny [7] może on być zastosowany przede wszystkim do projektowania pokazowych systemów ekspertowych. Natomiast EXSYS może być wykorzystany — według dotychczasowych doświadczeń autora — do praktycznej budowy małych systemów z bazą wiedzy.

Charakterystyka narzędzi tworzenia systemów ekspertowych

Charakterystyka	M.1	EXSYS	ESIE
Rodzaj problemu	diagnoza	planowanie	diagnoza
Reprezentacja wiedzy	pary: atrybut-wartość reguły: if-then, if-then-else, reguły zmienne, czynniki pewności	reguły: if-then, if-then-else, wartości zmienne, czynniki pewności	if-then-else
Rodzaj wnioskowania	do tyłu	do przodu do tyłu	do tyłu
Współpraca z użytkownikiem	wyjaśnianie śledzenie	ekran z menu w trakcie konsultacji	wyjaśnianie
Edytor bazy wiedzy	zewnętrzny	wewnętrzny	zewnętrzny
Niepewność i prawdopodobieństwo	tak	tak	nie
Współpraca z innymi językami	tak (C)	tak	nie
Funkcje arytmetyczne	tak	tak	nie
Grafika	nie	nie	nie
Okna	w zakr. śledz. wniosk.	nie	nie
Liczba reguł	ok. 200	5000	100—200
Język implementacji	Prolog	C	Pascal
Cena (w dolarach)	5000	395	wg uznania

LITERATURA

[1] Ambroziak J.: Systemy ekspertowe, cz. 1 i 2. Informatyka, nr 11—12, 1986 i 1, 1987
[2] Buchanan B. G., Shortliffe E. H.: Rule-based Expert Systems — The MYCIN Experiments of the Stanford Heuristic Programming Project. Addison-Wesley, Reading (MA), 1984
[3] D'Ambrosio B.: Building Expert Systems with M.1. Byte, June 1985
[4] ESIE — The Expert System Inference Engine. Knowledge Engineer's Manual. Lightwave Consultants, August 1985
[5] EXSYS. Expert Systec Developcent Package. Exsys Inc., 1985
[6] Gold J.: Do it yourself expert system. Computer Decision, 14 January 1986
[7] Harmon P., King D.: Expert Systems. Artificial Intelligence in Business. John Wiley & Sons, New York, 1985
[8] Jimenez J., King S.: Guru. Data Based Advisor, June 1986
[9] PC Technical Journal, Vol. 5, No. 5, p. 49, May 1987 (firmowa ulotka reklamowa)
[10] Siller W., Tucker D.: FLOPS User's Manual Version 1.2. Carraway Medical Center, Birmingham, 1986
[11] Systemy ekspertowe na IBM PC. Informatyka, nr 6, 1987
[12] Tello E. R.: Guru. Byte, August 1986.

Języki wysokiego poziomu do przetwarzania napisów...

dokończenie ze s. 3

nie danych do obszaru roboczego itp. COMIT posiada liczne możliwości formatowania przy wczytywaniu i drukowaniu.

Część reguły zwana **skokiem** steruje kolejnością działań w programie. Sterowanie może być przekazane do reguły opatrzonej nazwą, w szczególności z powrotem do tej samej, w celu powtórnego jej zinterpretowania, do następnej itd.

Pętle warunkowe mogą być pisane na kilka sposobów. Jedną z operacji warunkowych jest dopasowywanie lewej połowy, które może się nie spełniać. Na przykład, lewa połowa $10 nie daje się dopasować do obszaru roboczego z uprzednio określoną zawartością z powodu niewystarczającej, mniejszej od dziesięciu, liczby składowych w obszarze roboczym. Jeśli lewa połowa nie zostaje dopasowana, to reszta reguły nie podlega wykonywaniu i podejmuje się analizowanie następnej reguły.

W celu ułatwienia programowania pętli dla skoków i odpowiadających im nazw stosuje się specjalny zapis. Przyjmuje się, że gwiazdka pełni zastępczo rolę nazwy dla tych reguł, których nie opatrzono nazwami w sposób jawny. Gwiazdka umieszczona w polu skoku symbolizuje przekazanie sterowania do następnej reguły w kolejności (po wykonaniu bieżącej reguły). Ukośna kreska w polu skoku oznacza nawrót do wykonania bieżącej reguły. Przykładowo, następująca reguła może zostać użyta do usunięcia wszystkich wystąpień IF z obszaru roboczego:

* **IF** = /

Przejście do kolejnej reguły nastąpi tu z chwilą, gdy warunek dopasowania lewej połowy do obszaru roboczego nie spełni się. Pętle mogą być także organizowane za pomocą obliczeń arytmetycznych dokonywanych na liczbowych wartościach wskaźników.

Mechanizm zwany „spedytorem" (ang. dispatcher) zawiaduje wyborem alternatywnych dróg w programie, posługując się wskaźnikami logicznymi i ich zgodnością z nazwami reguł — w ten sposób dochodzi do interpretacji tzw. podreguł. Spedytor może wybierać jedną z kilku podreguł w sposób pseudolosowy. Możliwość wyboru drogi w programie na zasadzie przypadku jest przydatna w operacjach podobnych do tasowania kart. Z kolei reguły spisowe (ang. list rules) pozwalają na wyszukiwanie alternatyw w kolejności alfabetycznej. Przyśpiesza to działanie programu i pomaga przy przeglądaniu słowników i sortowaniu.

Aktualną wersją języka COMIT jest COMIT II, ukończony w roku 1971. COMIT II posiada kilka nowych właściwości, jednak wersja oryginalna jest z nim zgodna na zasadzie podzbioru. Istnieje implementacja języka COMIT II dla komputerów IBM 360/370, niemniej, w chwili obecnej język COMIT wychodzi z użycia.

Tłumaczył i opracował:
ZDZISŁAW PŁOSKI

LITERATURA

[1] Yngve V. H.: The COMIT System for Mechanical Translation. Proc. Intern. Conf. on Information Processing, 1959. R. Oldenbourg, Munich, 1960
[2] Yngve V. H.: Computer Programming with COMIT II. MIT Press, Cambridge (MA), 1972.

ZBYSZKO KRÓLIKOWSKI
Instytut Automatyki
Politechnika Poznańska

Mikrokomputerowe systemy zarządzania bazami danych

W niniejszym artykule przedstawiono syntetycznie podstawowe mikrokomputerowe systemy zarządzania bazami danych oraz charakterystykę tendencji, jakie aktualnie obserwuje się na rynku mikrokomputerowych systemów zarządzania bazami danych.

W chwili obecnej jest handlowo dostępnych około 150 różnych pakietów określanych przez producentów mianem mikrokomputerowych systemów zarządzania bazami danych (MSZBD). Jest to znaczny postęp w porównaniu do sytuacji sprzed zaledwie kilku lat, gdy dostępny był praktycznie tylko jeden system zarządzania bazą danych dBase II (firmy Asthon-Tate). W chwili obecnej do najbardziej popularnych MSZBD należą [4]: PSP:File/PFS:Report, C.I.P., Condor 3, R:base 4000, R:base 5000, R:base 6000, dBase II, dBase III, dBase III Plus, KnowledgeMan, Informix-ESQL/C, Progress, Unify, Reflex, Tadpol, Cardbox II, Archive, MicroRim, Smart Data Manager, Delta, Superfile, Dataflex i in. Zdecydowana większość z wymienionych MSZBD pracuje pod kontrolą systemów operacyjnych PC-DOS lub MS-DOS (lub kompatybilnych z nimi), przy czym jedynie dla kilku z tych systemów baz danych istnieją wersje pracujące pod kontrolą innych systemów operacyjnych, a mianowicie:

- dBase II — pod kontrolą CP/M, CP/M86,
- Delta — pod kontrolą CP/M, CP/M86, MPM,
- Cardbox II — pod kontrolą CP/M,
- Superfile — pod kontrolą CP/M86, Xenix, Unix,
- Informix — pod kontrolą Xenix, Unix.

Istnieje również wiele MSZBD przeznaczonych dla konkretnych typów mikrokomputerów, dysponujących specyficznymi systemami operacyjnymi. Z tej grupy do najbardziej znanych MSZBD należą Filevision i Helix, przeznaczone do pracy w systemie operacyjnym Apple Macintosh.

Charakteryzując dostępne handlowe MSZBD należy stwierdzić, że tylko nieliczne z nich spełniają wymogi profesjonalnych użytkowników. Podstawową cechą zdecydowanej większości MSZBD jest to, iż zapewniają one dostęp do mikrokomputerowej bazy danych tylko jednemu użytkownikowi, a więc obsługują środowisko jednozadaniowe. Co więcej, większość starszych systemów umożliwia użytkownikom pracę wyłącznie z pojedynczym plikiem danych. Innym problemem jest kwestia języka manipulacji danymi (języka zapytań). MSZBD dostarczają zwykle specyficznych języków zapytań (niekiedy jedynie w postaci zbiorów instrukcji), pozwalających na manipulowanie danymi na niskim poziomie. Z tego względu języki te znacznie odbiegają jakością od takich znanych języków zapytań, jak SQL (ang. Structured Query Language) czy QBE (ang. Query by Example).

Podsumowując, większość sprzedawanych obecnie mikrokomputerowych systemów zarządzania bazą danych to systemy reklamowane jako oparte na relacyjnym modelu danych. W rzeczywistości jednak są to raczej systemy zarządzania quasi-relacyjną bazą danych, bowiem nie spełniają one podanych przez Codda [1] wymogów, określanych jako minimalne właściwości relacyjne.

PRZEGLĄD MSZBD

Niżej zostaną krótko scharakteryzowane najciekawsze i najpopularniejsze w ostatnich latach mikrokomputerowe systemy zarządzania bazami danych.

Pfs:file/Pfs:report

Jest to system, w którym w danej chwili czasu może być przetwarzany tylko jeden plik danych. Umożliwia wykonywanie wszystkich podstawowych operacji na bazie danych, tj. wprowadzanie i usuwanie danych oraz ich aktualizację. Zaletą tego systemu jest prosty język zapytań typu QBE. System stwarza użytkownikowi możliwość operowania rekordami zmiennej długości. Ma rozbudowany, o wielu możliwościach, mechanizm drukowania raportów o zawartości bazy danych. Inną zaletą tego systemu jest możliwość wyszukiwania informacji według słów kluczowych, co predysponuje system do ważnej klasy zastosowań bibliograficznych. Indeksowanie rekordów w pliku danych jest wykonywane w tym systemie automatycznie (zawsze według pierwszego pola w rekordzie). Wadą systemu Pfs:file/Pfs:report jest ograniczenie do pracy z pojedynczym plikiem danych oraz niska wydajność systemu dla plików danych powyżej 180 KB.

C.I.P.

Podobnie jak system omówiony wyżej, C.I.P. (ang. Concentric Information Processor) może w danej chwili czasu pracować tylko z jednym plikiem danych. Jego zaletą jest ciekawie rozbudowany system wspomagania użytkownika — tzw. menu. Operowanie w bazie danych zaczyna się od założenia słownika danych, w którym definiuje się typy i atrybuty. W systemie C.I.P. istnieje możliwość obliczenia zawartości pól danych na podstawie innych pól. Wadą tego systemu jest brak języka zapytań — wyszukiwanie rekordów z bazy danych odbywa się na podstawie instrukcji oferowanych przez menu.

Condor 3

W odróżnieniu od systemów omówionych powyżej, Condor 3, umożliwia użytkownikowi jednoczesną pracę z kilkoma plikami. Praca rozpoczyna się od definiowania nazw, typów i długości pól. System Condor 3 umożliwia wykonywanie w bazie danych podstawowych operacji relacyjnych, tj. selekcji, projekcji i połączenia. Aktualizacja danych w systemie Condor 3 może być dokonywana z wykorzystaniem monitora (kursorem). Pliki mogą być indeksowane (jeden indeks na plik); indeksy mogą być tworzone na podstawie kombinacji pól. Mocną stroną systemu jest organizacja i wydruk raportów. Wadą systemu jest słabo zorganizowane i ubogie menu, co w konsekwencji wymaga od użytkownika pełnej znajomości poleceń systemu.

R:base 4000 i R:base 5000

Jest to jeden z najpopularniejszych systemów zarządzania mikrokomputerową bazą danych. Zasadniczą zaletą systemu R:base 4000 jest język zapytań typu SQL. Ponadto, programy przygotowane przez użytkownika mogą być kompilowane. Wadą systemu R:base 4000 jest brak możliwości odzyskiwania przestrzeni na dysku po usunięciu pliku (relacji) oraz stosunkowo słabo rozbudowany generator raportów.

Pewne niedogodności systemu R:base 4000 zostały poprawione w systemie R:base 5000, dla którego opracowano bardzo efektywne mechanizmy dostępu do danych (w tym również dostęp do plików przygotowanych przez system dBase II) oraz możliwość modyfikacji struktury bazy danych za pomocą pojedynczej instrukcji. Poza systemami

jednozadaniowymi R:base 4000 i R:base 5000, na rynku dostępna jest wersja systemu zarządzania wielodostępną bazą danych R:base 6000.

W kontekście systemów R:base należy zwrócić uwagę na dodatkowy pakiet tzw. oprogramowania przyjaznego dla użytkownika (ang. user friendly interface) o nazwie Clout, opracowany przez firmę Soft Sel. Umożliwia on realizację zapytań w postaci poleceń wyrażonych w języku naturalnym. Polecenia te są poddawane translacji na instrukcje języka zapytań odnoszące się do bazy danych R:base. Nowa wersja pakietu Clout umożliwia również dostęp do plików danych przygotowanych przez inne MSZBD, a mianowicie: dBase II, PFS:File i Lotus 1-2-3.

Reflex

Jest to system zarządzania bazą danych z pakietem graficznym, umożliwiającym użytkownikowi prezentowanie danych w różnej postaci. Jest to system przeznaczony dla użytkowników dokonujących częstych statystycznych analiz danych. System ten charakteryzuje się prostotą generowania raportów i wykresów ilustrujących analizę danych. Wymaga dużej pamięci (384 KB).

Tadpole

Podobnie jak system Reflex, Tadpole jest systemem zarządzania analityczną bazą danych. Zapewnia szerokie możliwości z zakresu analizy statystycznej, w tym: obliczanie odchylenia standardowego, regresji i korelacji liniowej, wyszukiwanie maksimum i minimum, test Studenta, analizę częstości wystąpień (na diagramach).

RefSys

Jest to system przeznaczony do organizacji i przechowywania danych bibliograficznych. Daje możliwość aktualizacji, wyszukiwania i drukowania raportów o danych bibliograficznych. W systemie RefSys użytkownik ma możliwość indeksowania plików danych według nazwisk autorów i tytułów prac. System daje możliwość drukowania listy wybranych pozycji literaturowych, uporządkowanych alfabetycznie.

Inne systemy

System Cardbox II jest tanim, małym systemem zarządzania bazą danych przeznaczonym dla początkujących użytkowników. Do grupy systemów stosunkowo tanich, lecz mających duże możliwości funkcjonalne można także zaliczyć systemy Achive, MicroRim, The Smart, Data Manager. Odnośnie tego ostatniego systemu warto wspomnieć o zastosowanej technice okienkowania przy prezentacji danych.

SYSTEMY dBase

Poniżej omówiono najpopularniejsze w ostatnich latach systemy zarządzania bazami danych dBase II i dBase III.

dBase II

W grupie systemów zarządzania baz danych pracujących pod kontrolą systemu operacyjnego CP/M zdecydowanie największą popularnością w ostatnich latach cieszył się system dBase II. Umożliwia on użytkownikowi wykonywanie w prosty sposób wszystkich podstawowych operacji w bazie danych, takich jak zakładanie bazy danych, aktualizację i usuwanie danych, podstawowe operacje relacyjne i kopiowanie plików danych. Wadą tego systemu w opinii użytkowników jest długi czas reakcji systemu przy wykonywaniu operacji sortowania i indeksowania plików danych.

KnowledgeMan

Możliwości i obsługa systemu zarządzania bazą danych KnowledgeMan są bardzo podobne jak w systemie dBase II. Jednakże w KnowledgeMan usunięto szereg istotnych ograniczeń systemu dBase II. Z punktu widzenia użytkowników najistotniejszymi ograniczeniami w dBase II są: stosunkowo mała liczba pól w rekordzie oraz możliwość działania tylko na dwóch plikach danych jednocześnie. System KnowledgeMan jest wyposażony w język zapytań typu SQL, jednakże w opinii użytkowników jest on mało efektywny. Kontakt użytkownika z bazą danych wydaje się być łatwiejszy w systemie dBase II. Również modyfikacja struktury bazy danych w systemie KnowledgeMan jest trudniejsza niż w systemie dBase II.

dBase III i dBase III Plus

Systemy dBase III i dBase III Plus są w chwili obecnej uważane za swego rodzaju standard w klasie mikrokomputerowych systemów zarządzania bazami danych. Ich duże możliwości eksploatacyjne oraz ogromna popularność wśród użytkowników, skłaniają ku temu, by potraktować te systemy nieco szerzej.

System zarządzania bazą danych dBase III [3], następca dBase II, jest przeznaczony dla użytkowników IBM PC/XT lub innych kompatybilnych z nim mikrokomputerów 16-bitowych. Jest napisany w języku C, wymaga pamięci operacyjnej — 256 KB. W dBase III wejście do systemu jest możliwe za pomocą dwóch funkcji pomocniczych HELP i ASSIST. Funkcja HELP umożliwia uzyskanie objaśnień dotyczących wywoływanych poleceń lub funkcji. Funkcja ASSIST pokazuje możliwości przetwarzania bazy danych. W porównaniu z dBase II, system dBase III ma znacznie większe możliwości dotyczące zakładania plików danych (do 20 plików) oraz jednoczesnego operowania plikami (do 10 plików może być jednocześnie otwartych w dBase III, a w dBase II tylko 2 pliki). Jeden plik danych może zawierać miliard rekordów (w dBase II — 65 535 rekordów) o długości 4000 znaków (w dBase II — 1000 znaków). Rekord może zawierać 128 pól (w dBase II — 32). W dBase III może występować pięć rodzajów pól: znakowe (maksymalnie 254 znaki), numeryczne, logiczne, pole daty i pole MEMO. Dwa ostatnie pola są nowością w porównaniu z dBase II. Pole MEMO umożliwia utworzenie wewnątrz pliku danych pola tekstowego o długości 4000 znaków. Jest ono przechowywane w osobnym pliku i może być przetwarzane osobno przez edytor tekstu. System dBase III pozwala na utrzymywanie dodatkowych 256 zmiennych (tzw. memory variables). Na tych zmiennych mogą być wykonywane operacje matematyczne (+, —, /, ÷, EXP, LOG, ROUND, SORT). Zmienne te mogą przyjmować postać łańcuchów. Łańcuchy można łączyć ze sobą.

Nowością w stosunku do dBase II jest polecenie RUN (wywoływanie programów zewnętrznych) oraz SET PROCEDURE TO — dające możliwość zamiany programów zewnętrznych na procedury. W porównaniu z dBase II znacznie powiększono możliwości i poprawiono parametry eksploatacyjne operacji sortowania i indeksowania. System dBase III umożliwia wykonywanie w prosty i przejrzysty sposób następujących operacji:

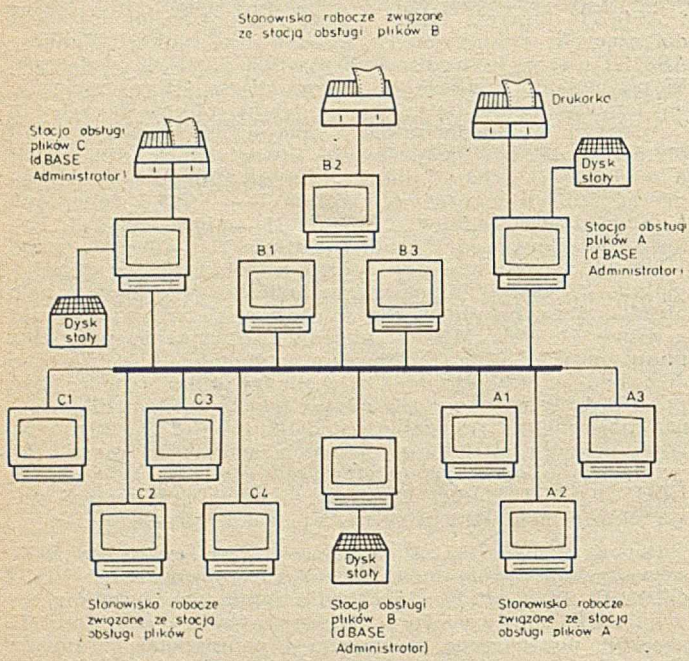
- zakładanie plików (CREATE),
- wyszukiwanie danych (DISPLAY, LIST, BROWSE, FIND), w tym przeszukiwanie według zadanych wartości pola znakowego,
- usuwanie i poprawianie danych (EDIT, DELETE),
- dodawanie nowych rekordów (APPEND, BROWSE),
- wykonywanie operacji relacyjnych: selekcji (DISPLAY... ...FOR), projekcji (BROWSE FIELDS ⟨— —⟩), połączenia dwu relacji (również w kombinacji z operacją projekcji i selekcji),
- sortowanie (SORT) i indeksowanie plików (INDEX ON), także według kombinacji znaków i liczb,
- kopiowanie i kombinowanie plików, także z możliwością wykonywania operacji projekcji i selekcji,
- automatyczne wykonywanie kilku poleceń na plikach (DO, IF..ELSE, DO WHILE),
- obliczanie wartości średniej i sumy z pól numerycznych, zliczanie rekordów spełniających dane warunki,
- tworzenie raportów o zawartości pól,
- nagrywanie przebiegu sesji.

Wersja systemu oznaczona dBase III Plus [2, 6] zawiera pewne dodatkowe funkcje wspomagające w szerszym zakresie wykonywania obliczeń statystycznych i finansowych. System zarządzania bazą danych dBase III Plus umożliwia pracę z bazą danych wielu użytkownikom (rys.), których stanowiska są połączone lokalną siecią komputerową IBM PC Network lub Novell Network (z systemem operacyjnym PC DOS wersja 3.1 lub następne).

W tabeli 1 zestawiono parametry techniczne i eksploatacyjne niektórych spośród omówionych systemów zarządzania bazami danych. W tabelach 2 i 3 zestawiono, na zasadzie porównania z systemem dBase II, możliwości techniczne i niektóre parametry eksploatacyjne systemu zarządzania bazą danych dBase III.

Tabela 1. Parametry techniczne i eksploatacyjne wybranych MSZBD; podane czasy uzyskano w przebiegach testowych na plikach zawierających 5000 rekordów 115-znakowych oraz — czasy w nawiasach — na plikach zawierających 6233 rekordy 27-znakowe

Operacje i parametry	Condor 3	dBase II	R : base 4000	KnowledgeMan	C.I.P.
Czas sortowania	18 min	2 godz (15 min)	25 min	27 min	—
Przeszukiwanie sekwencyjno-indeksowe	1 min (16 s)	1 min (39 s)	1 min (30 s)	4 min (30 s)	35 min
Operacja połączenia	1 min (40 s)	7 min (30 s)	5 min (34 s)	26 min	—
Maks. liczba rekordów na plik	65 534	65 535	nieograniczona	65 535	65 000
Maks. liczba znaków w rekordzie	1024	1000	1530	65 535	2000
Maks liczba pól w rekordzie	127	32	400	255	40
Maks. liczba znaków w polu	127	254	1500	65 535	50
Wymagania pamięciowe	128 KB	128 KB	256 KB	192 KB	192 KB



Wykorzystywanie systemu dBase III Plus w lokalnej sieci komputerowej

Tabela 2. Czasochłonność niektórych operacji na danych dla dBase II i dBase III (wyniki testów dla pliku zawierającego 5000 rekordów 20 znakowych)

Operacja	dBase II	dBase III
Sortowanie	70,45 min	7,35 min
Wyszukiwanie z pliku rekordów posortowanych	12,35 min	9,03 min
Wyszukiwanie z pliku rekordów nieposortowanych	63,09 min	48,50 min

Tabela 3. Parametry techniczne dBase II i dBase III

Parametry	dBase II	dBase III
Liczba rekordów w pliku danych	65 535	1 000 000 000
Liczba znaków w rekordzie	1000	4000
Liczba pól w rekordzie	32	128
Liczba znaków w polu	254	254
Liczba plików danych możliwych do założenia w systemie	10	20
Liczba jednocześnie otwartych plików danych	2	10
Liczba indeksów w pliku danych	7	7
Liczba zmiennych	64	256

KIERUNKI ROZWOJU

W rozwoju MSZBD można zaobserwować następujące trzy ogólne tendencje:

1. Dążenie do standaryzacji języka zapytań dla MSZBD lub opracowania pakietów programowych, umożliwiających danemu MSZBD dostęp do plików przygotowanych przez inne MSZBD.
2. Dążenie do wprowadzenia nowych języków wysokiego poziomu, z wbudowanymi mechanizmami języka dostępu do bazy danych.
3. Dążenie do zwiększenia stopnia współbieżności dostępu do mikrokomputerowej bazy danych (pierwszym etapem jest coraz szersze wprowadzenie sieciowych wersji MSZBD.

Tendencje standaryzacyjne w zakresie języka zapytań do baz danych obserwuje się już od kilku lat. Dotyczy to w szczególności systemów baz danych na dużych komputerach. Podobna tendencja daje się zaobserwować również w odniesieniu do MSZBD. Standard języka zapytań najprawdopodobniej zostanie także oparty na języku SQL. W chwili obecnej w szeregu MSZBD przyjęto SQL jako język dostępu. Równocześnie obserwuje się stosowanie innych rozwiązań, a mianowicie opracowywanie uniwersalnych pakietów umożliwiających różnym MSZBD dostęp do plików przygotowanych przez inne MSZBD. Najbardziej znanym tego rodzaju pakietem jest Clout firmy MicroRim Inc [5].

Dążenie do opracowania nowych języków wysokiego poziomu do współpracy z MSZBD wynika z założenia, że użytkownikami MSZBD są w większości wypadków użytkownicy nie będący profesjonalnymi informatykami. Nie dysponują oni zwykle odpowiednią praktyką i wiedzą, która umożliwiałaby im przygotowywanie efektywnych programów na podstawie języków stosowanych obecnie. Jako przykład nowego rozwiązania można podać system Progress (firmy Data Lanquages Corp.), pracujący zarówno w trybie jednozadaniowym jak i wielodostępnym (pod kontrolą systemu Unix). W systemie tym możliwe jest korzystanie z języka wysokiego poziomu, który charakteryzuje się prostotą i łatwością korzystania z niego. Język ten nie wymaga praktycznie żadnej wiedzy informatycznej. Inne rozwiązanie, Dataease 2.5, o podobnym charakterze przedstawiła ostatnio firma Soft.Solution.

Jak wspomniano stosunkowo niewielka grupa MSZBD pracuje w trybie wielodostępnym: dBase III Plus, R:base 6000, Superfile, Progress, Informix (najczęściej pod kontrolą systemu operacyjnego Unix). Opracowanie nowej wersji systemu operacyjnego MS-DOS dla potrzeb systemów wielomikrokomputerowych pracujących w sieci lokalnej stworzyło przesłanki do tworzenia tzw. sieciowych wersji MSZBD dostępnych jednocześnie przez sieć wielu użytkownikom (p. rys.): dBase III Plus-Lan, Fox-dBase, R:base 6000, Informix, Oracle Net itp. Jest to pierwszy etap na drodze do zwiększenia stopnia współbieżności dostępu do mikrokomputerowej bazy danych. Wydaje się, że w następnym etapie należy oczekiwać pojawienia się systemów zarządzania rozproszoną mikrokomputerową bazą danych oraz systemów umożliwiających zintegrowany dostęp do różnych, heterogenicznych MSZBD.

LITERATURA

[1] Codd E. F.: Relational database — a practical foundation for productivity. Comm. of the ACM, Vol. 25, pp. 109—117, 1982
[2] dBase Tools Series — The Programmer's Library. Ashton-Tate Publishing Group, 1986
[3] Jones E.: Using dBase III. Osborne McGraw-Hill, Berkeley (CA), 1985
[4] Krugliński D.: Database Management Systems. Byte Guide to the IBM PC, pp. 187—196, 1984
[5] Test — Rbase 4000 und Clout. Chip, Nr 2, 1985
[6] Using dBase III Plus. Osborne McGraw-Hill, Berkeley (CA), 1986.

KRZYSZTOF ZIELIŃSKI
JADWIGA INDULSKA
PAWEŁ MAGURA-WITKOWSKI
TOMASZ WALASEK
Instytut Informatyki
Akademia Górniczo-Hutnicza
Kraków

Architektura oprogramowania lokalnej sieci komputerowej UMMLAN-2

Ostatnio obserwuje się w kraju coraz powszechniejsze zrozumienie znaczenia, jakie dla budowy nowoczesnych systemów informatycznych mają lokalne sieci komputerowe. Przejawia się to zarówno w tematyce konferencji naukowych i planów badawczo-rozwojowych, dotyczących różnych zagadnień konstruowania tych systemów, jak i w coraz liczniejszych publikacjach popularyzatorskich.

Wielki wpływ na wzrost zainteresowania lokalnymi sieciami komputerowymi wywarło pojawienie się na krajowym rynku dużej liczby komputerów osobistych oraz lokalnych instalacji komunikacyjnych dla tych komputerów, oferowanych przez firmy polonijno-zagraniczne (np. produkty Transnet, D-Link i inne). Instalacje te należą do klasy wolnych lokalnych sieci komputerowych o szybkości transmisji rzędu 1 Mb/s. Jednocześnie w kilku ośrodkach akademickich w Polsce trwają prace nad budową lokalnych sieci komputerowych [2].

Podstawowe problemy techniczne konstruowania sprzętu dla szybkich sieci lokalnych, tzn. sieci o szybkości od 10 do 100 Mb/s, zostały już rozwiązane w krajach przodujących technologicznie, a obecnie trwają badania nad sieciami lokalnymi o szybkości powyżej 100 Mb/s. Jednakże w warunkach krajowych konstrukcja sprzętu zarówno dla wolnych, jak i szybkich sieci stanowi nadal trudne przedsięwzięcie organizacyjno-techniczne. Jest to spowodowane między innymi brakiem odpowiednich elementów scalonych LSI i VLSI oraz środków uruchomieniowych dla wysokiej klasy systemów mikroprocesorowych. Poważnym utrudnieniem jest też ograniczenie dostępu do najnowszej zagranicznej literatury fachowej.

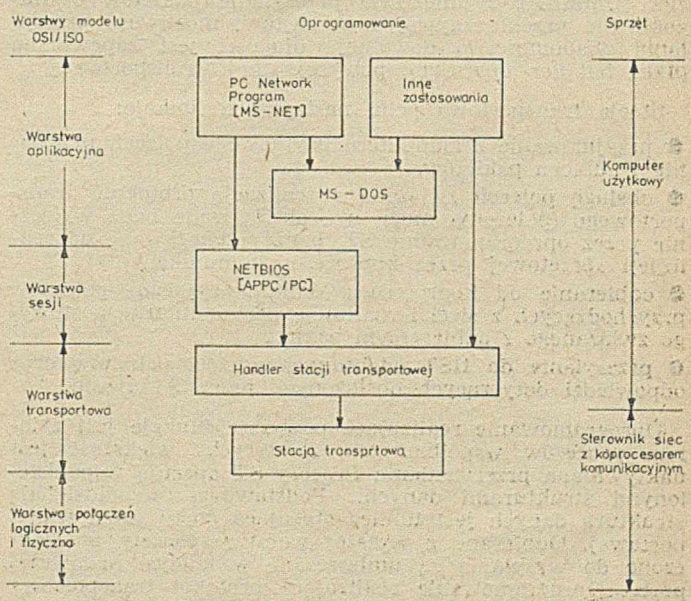
W Instytucie Informatyki Akademii Górniczo-Hutniczej w Krakowie, w latach 1985—1986, skonstruowano w wersji modelowej podsystem transmisji typu Ethernet, pracujący z szybkością do 10 Mb/s, przeznaczony dla sieci lokalnej o nazwie UMMLAN-2 (University of Mining and Metallurgy Local Area Network) [11]. Dalsze prace doprowadziły do uruchomienia prototypowych sterowników tej sieci dla komputerów IBM PC/XT i AT oraz MK-45. Umożliwiło to budowę eksperymentalnej instalacji sieci dla potrzeb naukowo-dydaktycznych, złożonej z kilkunastu komputerów.

Celem artykułu jest przedstawienie architektury oprogramowania sieci UMMLAN-2. W odróżnieniu od zagadnień sprzętowych, oprogramowanie lokalnych sieci komputerowych jest — niezależnie od szybkości transmisji — wciąż dyscypliną, w której prowadzi się liczne prace badawcze i normalizacyjne [5, 7, 8, 10]. Budowa bardzo nowoczesnego jak na warunki krajowe systemu komputerowego, w skład którego wchodzi kilkanaście komputerów IBM PC/XT i AT, stworzyła podstawę dla prac badawczych w dziedzinie rozproszonego przetwarzania informacji oraz opracowywania oprogramowania użytkowego dla określonych zastosowań sieci lokalnych.

STRUKTURA OPROGRAMOWANIA SIECI UMMLAN-2

Struktura oprogramowania sieci UMMLAN-2 odpowiada tylko w ograniczonym zakresie modelowi OSI/ISO. Pod względem funkcjonalnym można w niej wyróżnić podobne warstwy jak w modelu OSI, jednak świadomie zrezygnowano z zastosowania protokołów zalecanych przez ISO, dążąc do osiągnięcia maksymalnie efektywnego rozwiązania. Jest to dosyć często stosowane podejście przy tworzeniu większości istniejących sieci uniwersyteckich realizowanych od podstaw przez zespoły naukowo-badawcze. Wynika to częściowo z niedoskonałości istniejących standardów, które w wielu zastosowaniach są nieefektywne. Przykładowo, przyjęty w projektach MAP i TOP [8, 10] protokół transportowy ISO klasy 4 jest niepotrzebnie skomplikowany w odniesieniu do sieci lokalnych i — jak wykazały badania [9] — nie pozwala osiągnąć większej efektywnej szybkości transmisji niż 2 Mb/s. Podważa to sensowność stosowania tego protokołu w bardzo szybkich sieciach lokalnych. Podobne stanowisko w dziedzinie konstruowania oprogramowania sieciowego prezentują również firmy komputerowe, w tym IBM i Xerox, które tylko w ograniczonym stopniu opierają się na protokołach ISO. Należy podkreślić, że podejście takie nie oznacza rezygnacji ze zgodności programowej z uznanymi i ogólnie stosowanymi systemami sieciowymi, którą osiąga się w wyższych warstwach sieci UMMLAN-2. W szczególności warunek, iż sieć musi być zdolna do współpracy z oprogramowaniem sieciowym firm IBM i Microsoft uznano za jedno z wymagań obligatoryjnych przy projektowaniu oprogramowania.

Na strukturę oprogramowania UMMLAN-2 duży wpływ wywarła też przyjęta architektura sterownika sieci [3, 4, 11]. Sterownik jest dedykowanym komputerem komunikacyjnym, a więc zawiera oprócz specjalizowanych układów VLSI oddzielny procesor i inne układy tworzące urządzenie działające niezależnie od komputera użytkowego. Analizę zalet takiego rozwiązania można znaleźć w wielu pracach, m.in. [3, 4], i dlatego tutaj ją pominięto. Ogólną strukturę oprogramowania sieci UMMLAN-2 oraz jej odniesienie do architektury sprzętu sieci i modelu OSI/ISO przedstawiono na rysunku.



Architektura oprogramowania sieci UMMLAN-2

Oprogramowanie sieci można podzielić na dwie zasadnicze części: pierwsza rezyduje w sterowniku sieci, a druga składa się z systemowych programów współpracujących ze sterownikiem, ale wykonywanych przez komputer użytkowy. Oprogramowanie sterownika sieci, nazwane umownie stacją transportową (ST), odpowiada pod względem zakresu funkcji warstwie transportowej modelu OSI/ISO, przy czym niektóre funkcje transportowe są realizowane przez handler stacji transportowej (HST) wbudowany w system operacyjny komputera użytkowego.

Rozbudowany zestaw operacji HST umożliwia komunikację międzyprocesową oraz zarządzanie zasobami stacji transportowej. Pod względem architektury systemu operacyjnego HST odpowiada fragmentowi jądra systemu operacyjnego wykonującemu operacji komunikacji w sieci. Powyżej tego poziomu oprogramowania przewiduje się implementację specjalizowanych modułów sieciowych, jak NETBIOS lub APPC/PC. Można również wywoływać operacje handlera wprost z oprogramowania użytkowego. W tym zakresie skonstruowane oprogramowanie, chociaż powstało niezależnie, jest zgodne pod względem koncepcji z oprogramowaniem sieci Token Ring [7].

OPROGRAMOWANIE STEROWNIKA SIECI

Do pełnej charakterystyki oprogramowania sterownika sieci UMMLAN-2 wymagane jest omówienie:

- funkcji ST,
- sprzężenia ST z warstwą wyższą,
- przyjętej struktury oprogramowania,
- sposobu rozwiązania problemów implementacyjnych wynikających z założonych dla sieci wskaźników efektywności, przy występujących wymaganiach ze strony zastosowanego koprocesora komunikacyjnego (Intel 82586) oraz ograniczeniach dostępnej pamięci i szybkości procesora sterownika (Z-80).

Poniżej omówiono funkcje ST oraz sprzężenia ST z warstwą wyższą, przedstawiając w miarę pełną charakterystykę funkcjonalną rozwiązania. Dokładniejszy opis pozostałych zagadnień znajduje się w pracach [3, 4].

Stacja transportowa realizuje protokół bezpołączeniowy i połączeniowy warstwy transportu. Protokół połączeniowy stosuje się w celu zwiększenia niezawodności przesyłania pakietów danych przez wykrywanie zagubień pakietów, utraty sekwencji, duplikacji, przekłamania zawartości pakietu oraz przez kontrolę przepływu. W sieci UMMLAN-2 zastosowano protokół połączeniowy z niejawnym nawiązywaniem połączenia, uznawany za bardziej efektywny [1]. Wśród operacji transportowych nie ma zatem operacji nawiązywania i likwidowania połączeń.

Zastosowano metodę adresowania pakietów przez porty, tzn. oprogramowanie zapewnia przesłanie pakietów między portami zdalnych stacji transportowych. Port stacji transportowej jest dwukierunkowy, tzn. umożliwia zarówno nadawanie, jak i odbieranie pakietów. Stacja transportowa sieci UMMLAN-2 umożliwia przesyłanie pakietów o długości nie przekraczającej 1518 bajtów; możliwość przesyłania wiadomości o dowolnej długości jest zapewniana przez handler operujący powyżej stacji transportowej.

Stacja transportowa pełni następujące funkcje:

- przyjmowanie z komputera użytkowego poleceń przesłania i odbioru pakietów,
- obsługę poleceń związaną z realizacją protokołu transportowego (polecenia mogą być obsługiwane bądź wyłącznie przez oprogramowanie ST, bądź przekazywane do realizacji sprzętowej przez koprocesor komunikacyjny),
- odbieranie od koprocesora komunikacyjnego pakietów przychodzących z sieci i realizację protokołu transportowego związanego z odbieraniem pakietów,
- przesyłanie do HST odebranych z sieci pakietów oraz odpowiedzi dotyczących obsłużonych przez ST zleceń.

Oprogramowanie realizujące powyższe funkcje jest zbiorem procesów współbieżnych, z których większość jest uaktywniana przerwaniami. Procesy ST operują współdzielonymi strukturami danych. Podstawową współdzieloną strukturą danych jest tablica opisująca porty stacji transportowej. Odbierane z wyższej warstwy pakiety przeznaczone do wysyłania są umieszczane w kolejce nadawczej portu. Oprogramowanie realizujące protokół transportowy zleca koprocesorowi komunikacyjnemu wysłanie tych pakietów do sieci. W odwrotnym kierunku ST umieszcza w kolejce odbiorczej portu pakiety odbierane z sieci przez koprocesor komunikacyjny. Z kolejki tej, na żądanie procesu realizowanego w komputerze użytkowym, pakiety są przesyłane do przestrzeni adresowej procesu. W wypadku protokołu połączeniowego zastosowano potwierdzenie pakietu po jego przekazaniu do bufora odbierającego procesu. Należy zaznaczyć, że po stronie nadawczej ST nie wysyła następnego pakietu z danego portu do chwili odebrania potwierdzenia ostatnio wysłanego pakietu albo przekroczenia czasu kontrolnego. Ten rodzaj kontroli przepływu pakietów jest szczególnie korzystny w sieci typu Ethernet, charakteryzującej się krótkim czasem propagacji sygnału i dość długimi pakietami przesyłanych danych [6].

Stacja transportowa udostępnia swoje usługi przez sprzężenie z warstwą wyższą. Definicja sprzężenia obejmuje zbiór dopuszczalnych operacji i formaty przekazywanych poleceń i danych. Ze względu na fakt, że ST jest implementowana w samodzielnym sterowniku (mikrokomputerze komunikacyjnym), oprogramowanie musi też obsługiwać sprzęg między sterownikiem a komputerem użytkowym. Sprzęg ten tworzą linie sygnałowe oraz dwa kanały DMA. Jego funkcją jest umożliwienie transmisji informacji w obu kierunkach, natomiast sprzężenie programowe (zbiór operacji ST) ma zapewnić dostęp do wszystkich usług transportowych.

Poniżej przedstawiono jedynie charakterystykę sprzężenia programowego, ze względu na jego duże znaczenie w

Tabela 1. Operacje stacji transportowej sieci UMMLAN-2 (n — odpowiedź natychmiastowa a — odpowiedź asynchroniczna)

Operacja	Typ odpowiedzi	Znaczenie
Konfigurowanie i inicjowanie ST		
Init	n	Inicjowanie ST
Address	n	Przypisanie węzłowi sieci adresu indywidualnego albo adresów grupowych
Configure	n	Zdefiniowanie parametrów ST i koprocesora komunikacyjnego
Start	n	Umożliwienie nadawania i odbioru danych (uaktywnienie ST)
Zarządzanie portami		
Create_Port	n	Utworzenie portu
Destroy_Port	n	Likwidacja portu
Test_Port	n	Testowanie stanu portu
Wysyłanie pakietów		
Send	n	Wysyłanie pakietów w trybie bezpołączeniowym
Send/Ack	n, a	Przesłanie pakietu w trybie połączeniowym
Zarządzanie odbiorem pakietów		
Receive	n, a	Odbiór pakietu ze stacji transportowej (z kolejki odbiorczej portu)
Preview	n	Pobranie z portu parametrów pakietu czekającego na odbiór
Signal_Rcv	n, a	Żądanie od ST sygnalizowania przybycia pakietu do części odbiorczej portu
Remove	n	Usunięcie pakietu z kolejki odbiorczej portu
Diagnostyka		
Diagnose	n	Diagnostyka ST i koprocesora
	n	Odczytanie stałych i zmiennych ST

funkcjonowaniu sieci UMMLAN-2. Sprzęg sprzętowy został rozwiązany w sposób typowy dla połączeń między magistralami mikrokomputerów i omówiony w pracy [12].

Wszystkie operacje ST są obsługiwane w trybie polecenie — realizacja — odpowiedź natychmiastowa. Następna operacja może być zlecana ST dopiero po odebraniu odpowiedzi natychmiastowej na polecenie poprzednie. ST realizuje dwa rodzaje operacji: operacje lokalne, kończone definitywnie odpowiedzią natychmiastową, oraz operacje o dłuższym czasie wykonania wynikającym z konieczności realizowania protokołu transportowego między zdalnymi stacjami transportowymi. W operacjach drugiego rodzaju odpowiedź natychmiastowa jest informacją o przyjęciu operacji do realizacji (co umożliwia warstwie wyższej przesłanie następnego polecenia), a ostateczne zakończenie operacji jest sygnalizowane handlerowi za pomocą tzw. odpowiedzi asynchronicznej.

W tabeli 1 przedstawiono pełny zbiór operacji realizowanych przez ST. Na uwagę zasługuje elastyczność dostarczonych mechanizmów odbioru pakietów. Oprócz normalnego polecenia odbioru (Receive) przewidziano możliwość sprawdzania parametrów pakietu oczekującego w porcie (Preview), a także możliwość zażądania sygnalizowania przez ST przybycia pakietu do portu (Signal_Rcv). Ponadto, polecenie Remove zapewnia użytkownikowi eliminowanie skutków błędnego działania innych użytkowników sieci przez usunięcie pakietu z kolejki odbiorczej portu.

Opisany zbiór operacji stacji transportowej umożliwia implementację — w wyższej warstwie sieci UMMLAN-2 — bogatego zestawu operacji komunikacji i synchronizacji międzyprocesowej dla rozproszonych systemów komputerowych.

CHARAKTERYSTYKA HANDLERA SIECI UMMLAN-2

HST ściśle współpracuje ze stacją transportową, dokonując podziału wiadomości na pakiety o określonej długości. Poza tą typowo transportową funkcją, HST nie realizuje żadnego protokołu, dostarczając tylko mechanizmów do realizacji komunikacji międzyprocesowej opartej na usługach ST. Operacje HST są wywoływane analogicznie do funkcji systemu operacyjnego, a ich parametry są przekazywane w postaci 20-bajtowych rekordów zwanych NOB (ang. network operation block).

Podział zbioru operacji HST i krótki opis ich znaczenia przedstawiono w tabeli 2. Przyjęte skróty literowe określają następujące typy operacji:

N — natychmiastowa; wykonanie operacji odbywa się w trybie wywołania procedury, są to operacje lokalne, nie angażujące stacji transportowej,

Z — zawieszająca; wykonanie operacji powoduje zawieszenie procesu użytkowego do chwili jej zakończenia,

W — współbieżna; operacja jest wykonywana współbieżnie w stosunku do wywołującego ją procesu.

O zakończeniu operacji typu W wywołujący ją proces może zostać poinformowany o trzech sytuacjach:

— przy sprawdzaniu stanu tej operacji w bloku NOB,

— przy zawieszeniu się do czasu wykonania operacji typu W przez wywołanie specjalnie w tym celu wprowadzonej operacji Wait,

— przy dostarczeniu adresu procedury (tzw. Post routine), która jest wywoływana, gdy HST zakończy zleconą operację typu W.

Zamieszczone w tabeli 2 operacje Reset oraz Unlink są typowe i nie wymagają komentarza. Operacja Test_Station dostarcza informacji o stanie podsystemu komunikacji.

Operacje Enable_Posts oraz Disable_Post umożliwiają użytkownikowi odpowiednio blokowanie i odblokowanie kolejki procedur Post, czekających na wykonanie i znajdujących się pod kontrolą HST. Umożliwia to zaimplementowanie sekcji krytycznych. Cel wprowadzenia operacji Wait podano wcześniej. Operacja Kill umożliwia zaprzestanie operacji typu W.

Operacje Add_Address oraz Delete_Address umożliwiają manipulowanie adresami grupowych stacji transportowej, co jest bardzo przydatne do implementacji wyższych warstw oprogramowania.

HST udostępnia również elastyczne operacje na zegarze. Operacja Signal_Timeout inicjuje współbieżny proces odmierzania czasu. Program użytkowy może zainicjować kilkanaście takich procesów, których liczba jest ograniczona długością wewnętrznej kolejki HST. Do zaprzestania procesów odmierzania czasu służy operacja Kill_Timeouts. Zakończenie procesu odmierzania czasu jest sygnalizowane zgodnie z zasadami przyjętymi dla operacji typu W.

Grupa operacji na portach ST dostarcza typowych usług ich przydzielania, testowania i usuwania. Liczba portów jest parametrem konfiguracyjnym stacji transportowej.

Operacje nadawcze umożliwiają realizowanie przesłań w trybie bezpołączeniowym (Send_Datagram) oraz połączeniowym (Send_Message oraz Send_Message_Part). Operacja Send_Datagram jest typu natychmiastowego. Dwie pozostałe operacje są typu W i inicjują niezależne procesy nadawcze. Tak więc operacje nadawania długich wiadomości (do 64 KB) są realizowane współbieżnie pod kontrolą HST i nie blokują procesu wysyłającego wiadomość. Operacja Send_Message_Part służy do przesyłania wiadomości będącej częścią dłuższej wiadomości. Dzięki temu jest możliwe przesyłanie długich wiadomości łańcuchowanych, przechowywanych w nieciągłym obszarze adresowym

Tabela 2. Operacje handlera stacji transportowej

Operacja	Typ	Znaczenie
Operacje ogólnego przeznaczenia		
Reset	N	Zeruje stan HST i ST
Unlink	N	Dezaktywuje HST
Test_Station	N	Dostarcza parametry HST i ST
Add_Address	N	Definiuje nowy adres grupowy
Delete_Address	N	Usuwa adres grupowy
Enable_Posts	N	Odblokowuje wywołania procedury Post
Disable_Posts	N	Blokuje wywołania procedur Post
Wait	Z	Zawiesza proces użytkowy
Kill	N	Powoduje zaprzestanie wykonania operacji typu W
Operacje na zegarze		
Signal_Timeout	W	Inicjuje proces zegarowy
Kill_Timeouts	N	Powoduje zaprzestanie procesów zegarowych
Operacje na portach		
Allocate (_Any)_Port	N	Przydziela port
Delete_Port	N	Zwalnia port
Test_Port	N	Podaje stan portu
Operacje nadawcze		
Send_Message (Part)	W	Inicjuje proces przesłania wiadomości w trybie połączeniowym
Send_Datagram	N	Wysyła wiadomość w trybie bezpołączeniowym
Kill_Sends	N	Powoduje zaprzestanie procesów nadawania
Operacje odbiorcze		
Receive	W	Inicjuje proces odbioru
Kill_Receives	N	Powoduje zaprzestanie procesu oczekiwania
Signal_Receive	W	Inicjuje proces oczekiwania
Preview	N	Sprawdza nagłówek pakietu
Remove	N	Usuwa pakiet z portu

lub pobieranych fragmentami z pamięci zewnętrznej. Do zaprzestania operacji nadawania wprowadzono operacje Kill_Sends.

Operacje grupy odbiorczej powodują inicjowanie współbieżnych procesów odbioru wiadomości (Receive) lub oczekiwanie na wiadomość (Signal_Receive). Przybycie wiadomości do portu lub jej odbiór do bufora procesu użytkowego mogą być sygnalizowane zgodnie z zasadami przyjętymi dla operacji typu W (również przez wywołanie procedury Post). Dodatkowo wprowadzono możliwość odczytania nagłówka wiadomości znajdującej się w porcie (operacja Preview), co pozwala na podjęcie decyzji o odbiorze wiadomości (Receive) lub jej usunięciu z portu (Remove).

Jak wynika z zamieszczonego bardzo skrótowego przeglądu funkcji HST, oprogramowanie to stanowi dobrą podstawę do implementacji różnego rodzaju systemów sieciowych i przetwarzania rozproszonego.

PERSPEKTYWY ROZWOJU OPROGRAMOWANIA SIECI UMMLAN-2

Uniwersalność operacji HST jest kluczowym elementem umożliwiającym szybką adaptację dowolnego oprogramowania sieciowego do pracy w sieci UMMLAN-2. Pierwszym krokiem w tym kierunku jest implementacja programu systemowego emulującego — przy użyciu operacji HST — funkcje standardowego modułu NETBIOS firmy IBM. Umożliwia to zastosowanie w sieci UMMLAN-2, na przykład, zestawu programów IBM PC Network Program oraz bogatego oprogramowania firmy Microsoft, pracującego z systemem operacyjnym MS-DOS.

W przedstawionym ujęciu stosowanie firmowego oprogramowania jest alternatywą w stosunku do możliwości pisania własnego, specjalizowanego oprogramowania sieciowego. Operacje HST są dostępne z języków programowania C i Pascal, tworząc bardzo silne narzędzie wspomagające tworzenie takiego oprogramowania.

Kolejnym etapem prac jest włączenie HST do systemu operacyjnego Xenix. Stworzy to nowe możliwości w dziedzinie przetwarzania rozproszonego w sieci UMMLAN-2, dzięki udostępnieniu wielozadaniowości systemu Xenix i nowych funkcji systemu bardziej zaawansowanego w porównaniu dość prymitywnym systemem MS-DOS.

LITERATURA

[1] Birrell A. D., Nelson B. J.: Implementing Remote Procedure Calls. Report CS1-83-7, December 1983

[2] Borzemski L., Grzech A., Kasprzak A.: Stan i kierunki rozwoju sieci lokalnych dla potrzeb szkolnictwa wyższego. IX Szkoła Mikroprocesorowa „Lokalne Sieci Komputerowe", Łódź, październik 1986

[3] Indulska J., Nogieć J., Walasek T., Zieliński K.: The Impact of VLSI Technology on Communication Software for LAN — Practical Consideration of the UMMLAN-2 Project. Proc. Int. Conf. Systems Science IX, Wrocław, 1986

[4] Krasowski R., Zieliński K.: Wpływ układów VLSI na konstrukcję podsystemów transmisji i architektury oprogramowania sieci. IX Szkoła Mikroprocesorowa „Lokalne Sieci Komputerowe", Łódź, październik 1986

[5] Krumrey A.: Netware in Control. PC Technical Journal, November 1985

[6] Lee T. A., Linton A.: UDSTP — a Transport Protocol for a Uniform Datagram Service. University of Newcastle Technical Report 208, November 1985

[7] Mc Master D. S.: IBM's Token-Ring — What are the Alternatives. Mini-Micro Systems, September 1986

[8] MAP Specification, Version 2.1. General Motors Technical Center, 31 March 1985

[9] News Technology. Communication Systems Worldwide, p. 12, June 1987

[10] Technical Office Protocols Specification. Version 1.0. Boeing Corp. Seattle (WA), November 1985

[11] Zieliński K., Indulska J., Krasowski R.: Lokalna sieć komputerowa UMMLAN-2. Informatyka, nr 11—12, 1986

[12] Krasowski R., Zieliński K.: Design Alternatives of Distributed Computer System Communication Hardware. Microcomputer'86 — Design, Practice and Education. Bierutowice, Poland, 1986.

towarzystwo konsultantów polskich

tkp Oddział w Łodzi

ul. Suwalska 25/27, 93-176 Łódź, tel. 81-36-20 wew. 293

Pracownia Mikrokomputerowa TKP oferuje:

1. Programator pamięci EPROM typu 2716-27256 — cena: 180 tys. zł
2. Programator pamięci EPROM typu 2716-27512 — cena: 240 tys. zł
3. Programator układów 8748/49 — cena: 240 tys. zł
4. Emulatory pamięci EPROM w 9 wersjach od 2716-2732 do 2716-27512 — cena: od 180 tys. zł

Wszystkie w/w urządzenia są wykonywane w wersjach umożliwiających współpracę z komputerem za pośrednictwem interfejsu szeregowego RS-232C lub równoległego.

Wewnętrzne zabezpieczenia chronią układ przed uszkodzeniem w razie nieprawidłowego włożenia układu w podstawkę.

Ponadto oferujemy nasze usługi w zakresie projektowania specjalizowanych układów elektronicznych oraz opracowywania oprogramowania.

EO/87/87

MACIEJ M. SYSŁO
Instytut Informatyki
Uniwersytet Wrocławski

POLSKIE TOWARZYSTWO INFORMATYCZNE

Algorytmy kombinatoryczne i ich efektywność (2)

Problem plecakowy

W drugiej części artykułu omówiono **problem plecakowy** (ang. knapsack problem), zwany także problemem załadunku. Problem ten jest najprostszą wersją całkowitoliczbowego programowania liniowego, ma mianowicie tylko jedno ograniczenie liniowe, a zmienne mogą przyjmować jedynie wartości 0 lub 1.

BINARNY PROBLEM PLECAKOWY (KNAPSACK)
Dane:

Liczby naturalne n, a_1, a_2, ..., a_n, b_1, b_2, ..., b_n i B.
Cel:

Znaleźć wektor $x = (x_1, x_2, ..., x_n)$ o współrzędnych 0 lub 1 spełniający nierówność:

$$b_1x_1 + b_2x_2 + \ldots + b_nx_n \leqslant B,$$

i maksymalizujący funkcję:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n.$$

Mimo swej prostoty, problem plecakowy jest w pewnym sensie reprezentantem wszystkich problemów programowania liniowego w liczbach całkowitych. Każdemu zadaniu programowania całkowitoliczbowego można bowiem przyporządkować równoważne mu zadanie plecakowe. Zależność ta nie ma jednak zbyt wielkiego znaczenia praktycznego, ponieważ zadania plecakowe powstające w ten sposób, mają niewspółmiernie dużą liczbę zmiennych. Poniżej przedstawiono natomiast dwie inne wersje problemu plecakowego — decyzyjny problem plecakowy oraz problem podziału. Na ich przykładzie zilustrowano kilka typowych faktów. Ponieważ klasyczne, dokładne metody rozwiązywania problemu plecakowego są opisane w łatwo dostępnych opracowaniach, skupiono się przede wszystkim na przybliżonych metodach rozwiązywania. Decyzyjny problem plecakowy ilustruje sposób formułowania problemów decyzyjnych dla problemów optymalizacyjnych.

DECYZYJNY BINARNY PROBLEM PLECAKOWY (DEC-KNAPSACK)

Dane:
Liczby naturalne n, a_1, a_2, ..., a_n, b_1, b_2, ..., b_n, B i K.

Pytanie:
Czy istnieje wektor $x = (x_1, x_2, ..., x_n)$ o współrzędnych 0 lub 1 spełniający następujące nierówności:

$$b_1x_1 + b_2x_2 + \ldots + b_nx_n \leqslant B,$$
$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \geqslant K.$$

Jedną z wersji problemu DEC-KNAPSACK jest następujący problem podziału zbioru liczb na dwa podzbiory o tej samej sumie wartości ich elementów.

PROBLEM PODZIAŁU (PARTITION)

Dane:

Liczby naturalne n, a_1, a_2, ..., a_n, gdzie $\sum_{i=1}^{n} a_i = 2b$.

Pytanie:
Czy istnieje podzbiór $I' \subset I = \{1,2,\ldots,n\}$ taki, że

$$\sum_{i \in I'} a_i = \sum_{i \in I - I'} a_i = b?$$

Rozwiązanie każdego zadania problemu podziału może być sprowadzone do rozwiązania odpowiedniego zadania problemu DEC-KNAPSACK, prawdziwe jest bowiem następujące stwierdzenie: Zadanie problemu PARTITION o danych n, a_1, a_2, ..., a_n, gdzie $\sum_{i=1}^{n} a_i = 2b$ ma rozwiązanie TAK wtedy i tylko wtedy, gdy zadanie problemu DEC-KNAPSACK o danych n, a_1, a_2, ..., a_n, b, b ma rozwiązanie TAK.

Zatem każde zadanie problemu PARTITION może być łatwo przetransformowane na zadanie problemu DEC-KNAPSACK, czyli każdy algorytm rozwiązywania tego drugiego problemu może być zastosowany do rozwiązywania pierwszego problemu. Ta zależność między problemami PARTITION i DEC-KNAPSACK ilustruje często spotykane relacje między problemami kombinatorycznymi (w szczególności) i matematycznymi (w ogólności), gdy rozwiązywanie jednego problemu może być sprowadzone do rozwiązywania innego problemu. Transformacje takie odgrywają podstawową rolę w złożoności obliczeniowej.

Przestrzeń możliwych rozwiązań powyższych problemów składa się z wektorów o współrzędnych 0 lub 1. Zatem zadanie plecakowe z n zmiennymi ma co najwyżej 2^n możliwych rozwiązań. Dla $n = 10$, może ich być $2^{10} = 1024$, a więc znacznie mniej (aż o 5 rzędów), niż graf o 10 wierzchołkach ma drzew rozpinających. Niestety, jeśli chodzi o metody rozwiązywania obu problemów, to sytuacja jest do pewnego stopnia odwrotna. Problem SST może być rozwiązywany algorytmem o złożoności $O(n^2)$, nie jest natomiast znany żaden algorytm rozwiązywania problemu plecakowego, którego koszt byłby ograniczony przez wielomian zmiennej rozmiaru problemu; przypuszcza się, że taki algorytm nie istnieje.

Przed opisaniem przybliżonej metody rozwiązywania problemu plecakowego warto podać dokładny algorytm dla problemu podziału, oparty na schemacie programowania dynamicznego. Algorytm ten, mimo iż teoretycznie jest metodą nieefektywną, z powodzeniem może być stosowany do rozwiązywania praktycznych zadań, w których parametry należą do ograniczonego przedziału zmienności.

ALGORYTM DOKŁADNY

Algorytm programowania dynamicznego, który rozwiązuje problem podziału, wyznacza wartości funkcji boolowskiej (tj. funkcji o wartościach TAK lub NIE) T (i, j) dla $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant b$. Wartością T (i, j) jest T (TAK), jeśli wśród liczba a_1, a_2, ..., a_i istnieje podzbiór, którego suma jest równa j, natomiast N (NIE) — w przeciwnym wypadku. Inaczej mówiąc, T(i,j) jest T wtedy, gdy $\sum_{k=1}^{i} a_k x_k = j$ dla x_1, x_2, ..., x_i przyjmujących wartości 0 lub 1, a N — w przeciwnym razie. Zatem zadanie problemu podziału o danych n, a_1, a_2, ..., a_n ma rozwiązanie T wtedy i tylko wtedy, gdy T (n,b) ma wartość T. Wartość T (n,b) wyznacza się z zależności rekurencyjnej $T(n, b) = T(n-1, b) \cup T(n-1, b-a_n)$, w której pierwszy człon po prawej stronie odpowiada rozwiązaniu zadania, gdy $x_n = 0$, a drugi — gdy $x_n = 1$. Ogólnie, prawdziwa jest następująca zależność:

$$T(i, j) = T(i-1, j) \cup T(i-1, j-a_i),$$

dla $j = 1, 2, ..., b$, $i = 2, 3, ..., n$. W przypadku $i = 1$, przyjmuje się, że $T(1,a_1)$ jest T i T(1,j) jest N dla $j \neq a_1$. Ponadto określa się, że T(i,j) jest N dla $j \leqslant 0$. Powyższą zależność rekurencyjną stosuje się dla $i = 1$, $j = 1, 2, ..., b$,

Tabela 1. Ilustracja rozwiązania zadania podziału o parametrach: $n=4$, $a_1=2$, $a_2=3$, $a_3=5$, $a_4=6$

i \ j	1	2	3	4	5	6	7	b=8
1	N	T	N	N	N	N	N	N
2	N	T	T	N	T	N	N	N
3	N	T	T	N	T	N	T	T
n=4	N	T	T	N	T	N	T	T

$i = 2$, $j = 1, 2, ..., b, ..., i = n$, $j = 1, 2, ..., b$. W tabeli 1 zilustrowano zastosowania tej zależności do rozwiązania zadania podziału o parametrach $n = 4$, $a_1 = 2$, $a_2 = 3$, $a_3 = 5$, $a_4 = 6$. Element T(4,8) jest T, zatem zadanie to ma rozwiązanie, np. a_1+a_2 i a_2+a_3.

Rozwiązanie problemu podziału powyższą metodą wymaga wypełnienia nb elementów tablicy o n wierszach i b kolumnach, zatem czas potrzebny do wykonania wszystkich operacji z tym związanych jest ograniczony przez 0(nb). Wyrażenie nb jest wielomianem zmiennych n i b, nie jest jednak ograniczone przez żaden wielomian rozmiaru problemu, przy czym za **rozmiar problemu** przyjmuje się liczbę bitów potrzebnych do zapamiętania danych. Warto obliczyć, jaki jest rozmiar problemu podziału, tzn. ile potrzeba bitów do zapamiętania danych n, a_1, $a_2, ..., a_n$. Każda z liczb a_i wymaga co najwyżej $[\log_2 a_i]+1$ bitów, zatem do zapamiętania całego układu danych potrzeba $\sum_{i=1}^{n} [\log_2 a_i]+n$ bitów. Ponieważ $a_i \leqslant b$ dla każdego i, więc rozmiar problemu podziału jest co najwyżej $cn\log_2 b$, gdzie c jest pewną stałą. Wielomian nb nie jest zaś ograniczony przez żaden wielomian zmiennej $n\log_2 b$, gdyż b nie jest ograniczone przez $\log_2^k b$ dla żadnej stałej naturalnej k (bowiem $b/\log_2^k b \to \infty$, gdy $b \to \infty$, dla każdej stałej k).

Powyższy schemat wyznaczania T(n,b) nie jest więc algorytmem wielomianowym względem rozmiaru problemu, ale ta niewielomianowość wynika z występowania w obliczeniach liczb o nieograniczonych wartościach. Algorytm ten jest jednak wielomianowy, jeśli wartości parametrów są z góry ograniczone przez wielomian zmiennej rozmiaru problemu. Takie algorytmy są nazywane **pseudowielomianowymi.** Dla problemu kolorowania grafów, omawianego w następnej części, nie jest znany nawet algorytm pseudowielomianowy. W tym właśnie sensie problem plecakowy jest łatwiejszy do rozwiązywania niż problem kolorowania.

ALGORYTM PRZYBLIŻONY

Mówi się, że algorytm rozwiązywania problemu KNAPSACK wyznacza **rozwiązania absolutnie ε-przybliżone,** jeśli dla dowolnego zadania tego problemu algorytm ten generuje rozwiązanie o wartości K' różniącej się od wartości optymalnej K_{opt} o nie więcej niż ε, czyli $K_{opt}-K' \leqslant \varepsilon$.

W celu wykazania, że wyznaczanie rozwiązań absolutnie przybliżonych jest tak samo trudne jak wyznaczanie rozwiązań optymalnych, wraz z zadaniem o danych n, a_1, $a_2, ..., a_n$, b_1, $b_2, ..., b_n$, B, rozpatruje się zadanie o danych n, $(k+1)a_1$ $(k+1)a_2, ..., (k+1)a_n$, b_1, $b_2, ..., b_n$, B. Rozwiązanie optymalne drugiego z tych zadań ma wartość (k+1) razy większą niż wartość pierwszego zadania. Niech K_{opt} i K_{opt}^{k+1} oznaczają wartości rozwiązań optymalnych tych zadań, a K i K^{k+1} — wartości rozwiązań absolutnie ε-przybliżonych, dla $\varepsilon = k$. Prawdziwa jest nierówność:

$$K_{opt}^{k+1} - K^{k+1} \leqslant k,$$

skąd wynika:
$(k+1)(K_{opt}-K) \leqslant k$, czyli

$$K_{opt} - K \leqslant \frac{k}{k+1} < 1.$$

Ponieważ wszystkie liczby występujące w obu zadaniach są całkowite, z ostatniej nierówności wynika, że $K_{opt} = K$. Można stąd wywnioskować, że ewentualny algorytm wyznaczający rozwiązania absolutnie k-przybliżone dla zadań zmodyfikowanych wyznaczałby jednocześnie rozwiązania optymalne dla wszystkich zadań plecakowych. Istnienie wielomianowego algorytmu generującego rozwiązania absolutnie przybliżone jest więc mało prawdopodobne.

Poniżej przedstawiono ogólny schemat metody, wyznaczającej dla dowolnego zadania problemu KNAPSACK i ustalonego $\varepsilon > 0$ rozwiązanie **ε-przybliżone** o wartości K_ε, której **względny błąd** nie jest większy od ε, czyli:

$$\frac{K_{opt} - K_\varepsilon}{K_{opt}} \leqslant \varepsilon$$

Metoda ta jest rodziną n algorytmów (gdzie n jest liczbą zmiennych), z której dla każdego $\varepsilon > 0$ można zawsze wybrać algorytm generujący rozwiązania ε-przybliżone. Niech będzie dane zadanie programowania liniowego stowarzyszone z zadaniem problemu KNAPSACK, a powstałe przez zastąpienie ograniczenia $x_i = 0$ lub 1 ograniczeniem $0 \leqslant x_i \leqslant 1$ $(i = 1, 2, ..., n)$. By rozwiązać zadanie stowarzyszone, najpierw ustawia się zmienne według nierosnących ilorazów a_i/b_i, a następnie rozpatrując je w takiej kolejności przydziela się im możliwie największe wartości, aż do napięcia ograniczenia $\sum_{i=1}^{n} b_i x_i \leqslant B$. Idea ta przeniesiona na problem plecakowy sugeruje, że do osiągnięcia najcenniejszej zawartości należy wypełniać plecak towarami w kolejności nierosnących ilorazów a_i/b_i, które można uważać za względne zyski. Taka strategia nie zawsze jednak gwarantuje rozwiązanie optymalne, co zilustrowano poniższym przykładem.

Zmaksymalizować:
$60x_1+35x_2+30x_3$
przy ograniczeniach:
$3x_1+2x_2+2x_3 \leqslant 4$
$x_1, x_2, x_3 = 0$ lub 1.

Biorąc pod uwagę względne zyski, towary powinny być rozpatrywane w kolejności x_1, x_2, x_3, co prowadzi do rozwiązania (1,0,0). Takie samo rozwiązanie otrzymuje się rozpatrując zmienne w nierosnącej kolejności bezwzględnych zysków, tj. współczynników funkcji celu. Rozwiązaniem optymalnym powyższego zadania jest zaś wektor (0,1,1). Chociaż zachłanne sposoby wypełniania plecaka nie zawsze zapewniają optymalną zawartość, podejście to z powodzeniem jest stosowane w algorytmie wyznaczającym rozwiązania względnie przybliżone.

Dla wygody można założyć, że zmienne w zadaniu plecakowym są już uporządkowane tak, by były spełnione nierówności $a_1/b_1 \geqslant a_2/b_2 \geqslant ... \geqslant a_n/b_n$. Niech $I \subset N = \{1, 2, ..., n\}$ będzie podzbiorem towarów mieszczących się w plecaku, tj. $\sum_{i \in I} b_i \leqslant B$. Uzupełnienie zbioru I do możliwie najcenniejszego ładunku polega, zgodnie z powyższą ideą, na dokładaniu do plecaka towarów w kolejności wzrastania ich numerów. Niech L(I) oznacza wartość dodatkowej zawartości otrzymanej w ten sposób dla początkowego ładunku I. Wartość L(I) oblicza się według algorytmu:

begin

$L \leftarrow 0$; $C \leftarrow B - \sum_{i \in I} b_i$;

for $i \leftarrow 1$ **to** n **do**

if $i \notin I$ **and** $b_i \leqslant C$ **then**

begin $L \leftarrow L + a_i$; $C \leftarrow C - b_i$ **end**

end

Niech k będzie liczbą naturalną z przedziału (1,n). Algorytm Knapsack(k) wyznacza rozwiązanie binarnego zadania plecakowego, najlepsze jakie może być otrzymane przez wybór dowolnych k towarów mieszczących się w plecaku, a następnie uzupełnienie ich metodą zachłanną do najcenniejszego załadunku. Zysk (k) oznacza wartość rozwiązania otrzymanego algorytmem Knapsack (k).

ALGORYTM KNAPSACK (k)
Dane:

Parametry zadania plecakowego n, a_1, $a_2, ..., a_n$ b_1, $b_2, ... b_n$, B.
Wyniki:

Zysk — wartość najlepszego rozwiązania otrzymanego metodą zachłanną z podzbioru co najwyżej k towarów mieszczących się w plecaku.

begin

Zysk←0;

for każdego podzbioru $I \subset \{1, 2, \ldots, n\}$ takiego, że $|I| \leqslant k$ **and** $\sum_{i \in I} b_i \leqslant B$ **do**

$$\text{Zysk} \leftarrow \max \{\text{Zysk}, \sum_{i \in I} a_i + L(I)\}$$

end

Autor prezentowanej metody, S. Sahni, udowodnił, że wartość rozwiązania Zysk (k) spełnia nierówność:

$$\frac{K_{opt} - \text{Zysk}(k)}{K_{opt}} \leqslant \frac{1}{k+1}$$

Zatem, aby otrzymać rozwiązanie ε-przybliżone należy wybrać k_ε spełniające $1/(k_\varepsilon + 1) \leqslant \varepsilon$, czyli $k_\varepsilon \geqslant 1/\varepsilon - 1$. Łatwo sprawdzić, że algorytm Knapsack (k) może być zrealizowany w czasie $0(kn^{k+1})$, jest to więc algorytm wielomianowy (zmiennej n, gdyż k jest stałą). Dla ustalonej dokładności $\varepsilon > 0$, czas działania algorytmu Knapsack (k_ε) wyznaczającego ε-przybliżone rozwiązania jest ograniczony przez wielomian zmiennej n. Zależność od ε (a raczej od 1/ε) jest jednak niewielomianowa. W pełni wielomianowy (zmiennych n i 1/ε) schemat aproksymacyjny dla problemu plecakowego można znaleźć w [1].

Powyższą klasę algorytmów przybliżonych można zastosować do rozwiązania następującego zadania plecakowego:
Zmaksymalizować:
$200x_1 + 155x_2 + 115x_3 + 90x_4$
przy warunkach:
$50x_1 + 40x_2 + 30x_3 + 25x_4 \leqslant 95$
i $x_1, x_2, x_3, x_4 = 0$ lub 1.

W tabeli 2 zilustrowano działanie algorytmu Knapsack (k). Rozwiązanie optymalne (0,1,1,1) jest generowane przez Knapsack (2).

Tabela 2. Wyniki przykładowego działania algorytmów Knapsack(k), k=1, 2, 3

k=1	I	{1}	{2}	{3}	{4}		
	Zysk	355	355	315	290		
	Rozwiązanie	(1100)	(1100)	(1010)	(1001)		
k=2	I	{1,2}	{1,3}	{1,4}	{2,3}	{2,4}	{3,4}
	Zysk	355	315	290	360	360	360
	Rozwiązanie	(1100)	(1010)	(1001)	(0111)	(0111)	(0111)
k=3	I	{2,3,4}	Pozostałe grupy trzech towarów są niedopuszczalnymi ładunkami				
	Zysk	360					
	Rozwiązanie	(0111)					

Najefektywniejsze dokładne algorytmy rozwiązywania problemu plecakowego składają się najczęściej z dwóch etapów. Pierwszy etap, będący algorytmem wielomianowym, polega na redukcji zmiennych, drugi zaś jest realizacją metody podziału i oszacowań. Liczba zmiennych eliminowanych z losowo generowanych zadań problemu waha się w granicach 60—80%. Etap pierwszy ma więc duży wpływ na rozmiar zadań rozwiązywanych w drugim etapie przez algorytm, którego czas działania nie można oszacować przez żaden wielomian. Wyniki eksperymentów obliczeniowych [2] pokazują, że zadania plecakowe z n = 500 mogą być rozwiązywane w czasie 0,5 s.

LITERATURA

[1] Ibarra O. H., Kim C. E.: Fast approximation algorithms for the Knapsack and sum of subset problems. Journal of the ACM, Vol. 22, pp. 463—468, 1975

[2] Sysło M. M., Deo N. Kowalik J. S.: Discrete Optimization Algorithms with Pascal Programs. Prentice-Hall, Englewood Cliffs (NJ), 1983.

Nowe książki

L. Bolc (Ed.): Computational Models of Learning, Springer-Verlag, 1987.

Maszynowe uczenie się jest jedną z aktywniej rozwijanych obecnie dziedzin sztucznej inteligencji i nauk poznawczych (ang. cognitive science). Książka stanowi cenne uzupełnienie artykułów z dziedziny maszynowego uczenia się, pojawiających się coraz liczniej w pismach naukowych. Zawiera kilka szeroko i przystępnie omówionych tematów badawczych takich jak: strategie uczenia się i automatyczne przyswajanie wiedzy, heurystyka w odkryciach empirycznych, rola treningu w uczeniu proceduralnym, zdobywanie wiedzy koncepcyjnej w procesach poszukiwania, rozwój poznawczy jako optymalizacja.

L. Bolc (Ed.): Natural Language Parsing Systems, Springer Verlag, 1987

Jest to zbiór artykułów poświęconych zagadnieniom badawczym w dziedzinie rozbioru gramatycznego języka naturalnego. Książka zawiera szczegółowe opisy prac — zarówno teoretycznych, jak i praktycznych — prowadzonych przez badaczy w różnych krajach.

A. K. Wong, A. Pugh: Machine Intelligence and Knowledge Engineering for Robotic Applications, Springer-Verlag, 1987.

Jest to zbiór referatów wygłoszonych na konferencji roboczej NATO poświęconej inteligencji maszynowej i inżynierii wiedzy w robotyce, która odbyła się w 1986 roku we Włoszech. Artykuły przygotowane przez przedstawicieli przemysłu, agencji rządowych i instytucji akademickich dotyczą takich zagadnień jak widzenie i czucie maszynowe (ang. machine sensing), reprezentacja wiedzy i rozpoznawanie obrazów, systemy z bazami wiedzy, planowanie zadań, planowanie trajektorii, śledzenie, systemy sterujące, sprzęt i oprogramowanie.

Zakłady Tworzyw Sztucznych

„ERG"

w Pustkowie

zatrudnią natychmiast

INŻYNIERÓW INFORMATYKÓW

z praktyką

Przedsiębiorstwo zapewnia mieszkanie, możliwość osiągania zarobków od 30 do 50 tys. złotych.

Zgłoszenia przyjmuje **i udziela informacji**

Dział Kadr i Szkolenia,

39-206 Pustków 3,
tel. Dębica 40-61 wew. 244, 597, 508.

EO/268/88

ARTUR KRĘPSKI
Instytut Podstaw Informatyki
Polska Akademia Nauk
Warszawa

Język Smalltalk-80 (2)

Niniejszy artykuł stanowi drugą część całości poświęconej językowi Smalltalk-80; dokończono w niej przegląd konstrukcji języka (bloki, metody) oraz zaprezentowano przykład klasy. W części pierwszej przedstawiono intuicyjnie elementarne pojęcia tego języka oraz wprowadzono podstawowe wyrażenia (stałe, zmienne, komunikaty). Całość artykułu zawiera też podsumowanie i gramatykę języka.

Bloki

Blok określa ciąg wyrażeń, który jest wartościowany nie w miejscu jego tekstowego wystąpienia a dopiero wtedy, gdy do bloku wyśle się specjalny komunikat. Można powiedzieć, że blok stanowi jakby definicję metody bez nazwy (tj. bez selektora).

Składniowo, ciąg wyrażeń bloku jest zamknięty w nawiasach kwadratowych. Identyfikatory parametrów bloku występują na jego początku (każdy poprzedzony znakiem dwukropka) i są zakończone znakiem pionowej kreski. Wartością bloku jest wartość ostatniego wyrażenia w bloku (lub stała nil, jeśli blok nie zawiera wyrażeń).

Wyrażenie w bloku może być poprzedzone znakiem strzałki ↑. Wartościowanie takiego wyrażenia powoduje zakończenie wykonywania bloku i metody, która go zawiera — jego wartość staje się wartością bloku i metody.

Ponieważ blok nie ma nazwy (selektora), to selektor komunikatu, który go uaktywnia ma specjalną (standardową) postać — albo jest unarnym selektorem value (gdy blok nie ma parametrów), albo jest selektorem kluczowym będącym tylokrotnym złożeniem klucza value, ile argumentów liczy ten blok (value: — jeden argument, value:value: — dwa argumenty itd.).

Bloki wykorzystuje się głównie do zapisywania **struktur sterowania**. Najprostsze takie struktury — to wybór warunkowy (na podobieństwo instrukcji if-then-else, np. w Pascalu) oraz iteracja (instrukcja while).

Wybór warunkowy jest zrealizowany w postaci następujących metod w klasie Boolean:
ifTrue:, ifFalse:, ifTrue:ifFalse:, ifFalse:ifTrue:

Przykładowo, wartością następującego wyrażenia będzie 1, 0, —1 w zależności od znaku liczby x:

```
x > 0
  ifTrue: [1]
  ifFalse: [x = 0
              ifTrue: [0]
              ifFalse: [—1]]
```

W powyższym wyrażeniu, jeśli warunek x > 0 będzie spełniony, tj. wynikiem będzie obiekt oznaczony przez stałą true, to metoda:
ifTrue:ifFalse:

wysłana do niego spowoduje wysłanie komunikatu value do bloku [1] ([1] value), i ostatecznie wartością całego wyrażenia będzie 1.

Iteracja jest zrealizowana jako komunikat
whileTrue:

lub

whileFalse:

wysłany do dowolnego bloku udostępniającego jako wartość obiekt klasy Boolean, z argumentem będącym blokiem wykonywanym tyle razy, ile razy odbiorca udostępnia odpowiednio wartość stałej true lub false. Przykładowo, wartościowanie następującego wyrażenia powoduje wyzerowanie tablicy list (komunikat size wysłany do tablicy powoduje obliczenie jej długości):

```
[index <= list size]
  whileTrue: [list at: index put: 0. index := index + 1]
```

Stosunkowo łatwo definiuje się inne struktury sterowania niż dotąd wymienione. Przykładowo, iterowanie określoną liczbę razy bloku aBlock można zapisać jako komunikat o selektorze:

timesRepeat:

i argumencie aBlock wysyłany do liczby całkowitej — krotności iteracji. Oto — metoda (p. punkt następny) reagująca na ten komunikat:

```
timesRepeat: aBlock
  | index |
index := 1.
[index <= self] whileTrue:
  [aBlock value. index := index + 1]
```

W powyższej metodzie mamy do czynienia z trzema blokami — jeden, o nazwie aBlock, jest parametrem metody; dwa pozostałe służą do zapisania iteracji:

whileTrue:

Blok aBlock jest wartościowany w każdym kroku iteracji w wyniku wysłania komunikatu unarnego o selektorze value (aBlock jest z założenia blokiem bez parametrów). Zmienna self oznacza odbiorcę komunikatu, tzn. liczbę całkowitą będącą krotnością iteracji.

Metody

Składniowo, metoda składa się z następujących części:

— szablon (selektor z listą parametrów metody),
— deklaracja zmiennych roboczych (wyszczególnienie ich identyfikatorów ujęte w znaki pionowych kresek),
— ciąg wyrażeń (z których ostatnie może być poprzedzone znakiem strzałki ↑).

Metoda jest funkcją, której wykonanie polega na wartościowaniu zadanego ciągu wyrażeń. Wynikiem metody jest wartość wyrażenia poprzedzonego znakiem strzałki, bądź (gdy żadne wyrażenie nie jest poprzedzone znakiem strzałki) odbiorca komunikatu, który uaktywnił metodę (co jest równoważne dodaniu wyrażenia ↑ self na koniec metody). Obiekt będący wynikiem metody jest udostępniany nadawcy tego komunikatu. Zauważmy, że metody nie mogą być zagnieżdżane.

Uaktywnienie metody następuje zawsze w reakcji na wysłanie komunikatu, tj. zastosowanie selektora z pewnymi argumentami do obiektu będącego odbiorcą komunikatu. Odszukanie metody, która pasuje do komunikatu (selektory metody i komunikatu są identyczne) odbywa się na etapie wykonania programu, tzn. wiązanie komunikatu z metodą jest dynamiczne (ang. late binding), a nie statyczne (wykonywane przez kompilator albo program łączący).

Poszukiwanie metody pasującej do komunikatu zaczyna się od klasy odbiorcy komunikatu i przebiega przez wszystkie jej nadklasy, aż do klasy Object (szczytowej w hierarchii klas). Jeżeli taka pasująca metoda nie zostanie znaleziona, to jest sygnalizowany błąd („Niezrozumiały komunikat").

Wyjątkiem w tak opisanym procesie poszukiwania jest przypadek wysłania komunikatu do odbiorcy wyznaczonego przez nazwę zastrzeżoną super — wówczas poszukiwanie metody pasującej do komunikatu rozpoczyna się od nadklasy klasy zawierającej metodę, w której występuje

wysłanie komunikatu (a nie jak zwykle — również w przypadku self — od klasy jego odbiorcy). Umożliwia się w ten sposób odwołanie do metody oryginalnej (z nadklasy) w podklasie przedefiniowującej metodę.

Przykład

Oto jedna z kilkunastu metod podanych w następnym punkcie:

```
<| aVector
  | teta |

teta := self * aVector / self length / aVector length.
teta := teta arccos
self //*) aVector >=0

  ifTrue: [↑ teta]
  ifFalse: [↑ teta negated]
```

Powyższa metoda ma szablon składający się z selektora binarnego <| i parametru (wektora) o nazwie aVector. Zastosowana do odbiorcy — wektora (w metodzie jest on wartością zmiennej self) daje kąt skierowany między obomą wektorami.

Po deklaracji zmiennej roboczej teta występują trzy wyrażenia — najpierw przypisanie zmiennej teta wartości kąta nieskierowanego, w notacji geometrycznej:

arccos (self * aVector / (| self | * | aVector |))

a następnie wybór warunkowy:

ifTrue:ifFalse:

w zależności od znaku wyznacznika wektorów self i aVector (wyznacznik jest jedną z metod zdefiniowanych w następnym punkcie).

Niewielka część metod -- rzędu stu, dwustu — z różnych klas jest zrealizowana jako tzw. **metody pierwotne.** Są one zakodowane w języku maszyny docelowej i są znane (przez swoje numery) bezpośrednio procesorowi języka Smalltalk.

Składniowo, metodę pierwotną wyróżnia się w ten sposób, że po szablonie metody występuje dodatkowo oznaczenie metody pierwotnej w postaci:

⟨primitive: numer-metody-pierwotnej⟩

Wartościowanie dalej zadanych wyrażeń (zwykle sygnalizujących sytuację błędną) następuje tylko wtedy, gdy podprogram metody pierwotnej nie wykonuje się poprawnie.

Metody pierwotne są wykorzystywane do zakodowania najczęściej wykonywanych metod systemowych, od efektywności których może istotnie zależeć wydajność całego systemu (np. arytmetyka liczbowa, elementarne operacje na obiektach).

Przykład

Metodę pierwotną o numerze 29 [7] można wykorzystać w metodzie mnożenia liczb całkowitych, np. w następujący sposób:

```
* aNumber

  ⟨primitive: 29⟩
  self error: 'Nadmiar całkowitoliczbowy'
```

Przykład klasy

Jako przykład zapiszemy zestaw metod dla klasy Vector (podklasa klasy Object), której reprezentantami będą wektory (swobodne) na płaszczyźnie euklidesowej (wydruk 1). Zmienne reprezentatywne tej klasy o nazwach x oraz y są współrzędnymi wektora.

Jedyna (przykładowa) metoda klasowa (o selektorze kluczowym newX:newY) służy tworzeniu nowego wektora; wszystkie pozostałe — to metody reprezentatywne. Są one pogrupowane i odpowiednio zatytułowane w komentarzach (komentarz to ciąg znaków ujętych w cudzysłów).

Powyższą klasę Vector wykorzystamy do rozwiązania przykładowego zadania z geometri analitycznej na płaszczyźnie. Dane są współrzędne wierzchołków trójkąta A (—2,2), B (4,—2), C (0,3) — oblicz miary kątów wewnętrznych trójkąta ABC i jego pole. Odpowiedni program w języku Smalltalk-80 przedstawiono na wydruku 2.

* * *

Język Smalltalk-80 jest zunifikowanym, wyrażeniowym, rozszerzalnym, beztypowym językiem programowania obiek-

```
newX: xValue  newY: yValue
"tworzenie reprezentanta - nowego wektora [xValue,yValue]"

^self new  x: xValue
           y: yValue

"--------------------"
"atrybuty wektora"
"--------------------"

x: xCoordinate  y: yCoordinate  "nowe współrzędne"
x := xCoordinate. y := yCoordinate.

x  "współrzędna x"
^x

y  "współrzędna y"
^y

length  "długość wektora"
^ ((x*x) + (y*y)) sqrt

"--------------------"
"rachunki wektorowe"
"--------------------"

+ aVector  "suma wektorów"
^ Vector newX: x + aVector x
         newY: y + aVector y

@ alfa  "mnożenie przez skalar"
^ Vector newX: alfa * x
         newY: alfa * y

"--------------------"
"rachunki skalarne"
"--------------------"

* aVector  "iloczyn skalarny"
^ (x * aVector x) + (y * aVector y)

/\ aVector  "wyznacznik"
^ (x * aVector y) - (y * aVector x)

"--------------------------"
"relacje między wektorami"
"--------------------------"

= aVector  "równość"
^ x = aVector x and: [y = aVector y]

|| aVector  "równoległość"
^ self /\ aVector = 0

|= aVector  "prostopadłość"
^ self * aVector = 0

"----------------"
"kąt skierowany"
"----------------"

(| aVector  "kąt skierowany między wektorami"
  | teta |
teta := self * aVector / self length / aVector length.
teta := teta arcCos.
self /\ aVector )= 0
  ifTrue: [^teta]
  ifFalse: [^ teta negated]
```

Wydruk 1

```
"-----"
'"Dane"
"-----"

Ax := -2. Ay := 2.
Bx := 4. By := -2.
Cx := 0. Cy := 3.

"----------"
"Rozwiązanie"
"----------"

AB := Vector newX: (Bx-Ax) newY: (By-Ay).
AC := Vector newX: (Cx-Ax) newY: (Cy-Ay).
BC := Vector newX: (Cx-Bx) newY: (Cy-By).
"Miary kątów wewnętrznych"
Alfa := AB (| AC.
Beta := BC (| (AB@(-1)).
Gamma := AC (| BC.
"Pole"
PoleABC := AB length * AC length * Alfa sin / 2.
```

Wydruk 2

towego z mechanizmami dziedziczenia (prefiksowania) klas oraz dynamicznego wiązania jako narzędziami abstrakcji (modularyzacji). Zawiera wbudowany, bogaty zestaw mechanizmów programowania, m.in. wyrafinowane struktury danych i struktury sterowania (w tym także mechanizmy programowania równoległego).

*) W tym miejscu powinna być wstawiona kreska pochylona w lewo, lecz drukarnia nie ma takiego znaku *(przyp. red.)*

Najważniejsze cechy języka można podsumować następująco:

1) **unifikacja mechanizmów** — wprowadza się bardzo małą liczbę pojęć podstawowych (obiekt, klasa, reprezentant, metoda, komunikat), z których buduje się cały system; każdy element języka i systemu jest obiektem,

2) **wyrażeniowość** — każda konstrukcja języka ma określoną wartość (obiekt), w szczególności metoda jest właściwie funkcją,

3) **rozszerzalność** — każda klasa staje się elementem języka (jej nazwa ma zasięg globalny); język Smalltalk-80 z tego punktu widzenia jest tylko pewnym (minimalnym) zbiorem wyróżnionych obiektów służących do zapisu metod,

4) **beztypowość** — nie występuje pojęcie typu (odpowiednikiem typu obiektu jest jego klasa; dla stałych, ich składnia jednoznacznie określa klasę, której są reprezentantami),

5) **mechanizm dziedziczenia (prefiksowania)** — pozwala wiązać klasy w hierarchię, wyróżniając dla danej klasy jej nadklasę i podklasy; otrzymuje się w ten sposób możliwość — odpowiednio abstrahowania i specjalizowania klas,

6) **dynamiczne wiązanie** — odnajdywanie metody pasującej do danego komunikatu odbywa się na etapie wykonania na podstawie hierarchii klas.

DODATEK. Gramatyka języka Smalltalk-80

Do opisu składni języka Smalltalk-80 użyto rozszerzonej notacji BNF (ang. EBNF). W porównaniu z oryginalnym formalizmem BNF wprowadzono następujące zmiany:

1) każda produkcja kończy się kropką,

2) symbole terminalne są ujęte w cudzysłów (nieterminalne bez metanawiasów ”⟨” i ”⟩”),

3) użyto następujących metanawiasów, aby skrócić zapis produkcji:

(x|y) — wystąpienie x lub y
[x] — 0 lub 1 wystąpienie x
{x} — 0 lub więcej wystąpień x

Symbolem początkowym gramatyki języka Smalltalk-80 jest metoda.

```
1. blok = "[" [":" identyfikator {":" identyfikator} "|"]
          [ciąg-wyrażeń] "]".

2. ciąg-wyrażeń = wyrażenie {"." wyrażenie} ["." "^" wyrażenie] |
                  "^" wyrażenie.

3. cyfra = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".

4. cyfra-uogólniona = cyfra | duża-litera.

5. cyfry = cyfra {cyfra}.

6. cyfry-uogólnione = cyfra-uogólniona {cyfra-uogólniona}.

7. duża-litera = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
                 "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
                 "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z".

8. identyfikator = litera {litera | cyfra}.

9. klucz = identyfikator ":".

10. komunikat = komunikat-unarny | komunikat-binarny |
                komunikat-kluczowy.

11. komunikat-binarny = selektor-binarny odbiorca-komunikatu-unarnego.

12. komunikat-kluczowy = klucz odbiorca-komunikatu-binarnego
                         {klucz odbiorca-komunikatu-binarnego}.

13. komunikat-unarny = selektor-unarny.

14. liczba = [cyfry "r"] ["-"] cyfry-uogólnione          ["." cyfry-uogólnione]
                                                         ["e" ["-"] cyfry].

15. litera = duża-litera |
             "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |
             "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |
             "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z".

16. metoda = szablon-metody  [oznaczenie-metody-pierwotnej]
                             [zmienne-robocze] [ciąg-wyrażeń].

17. napis = "'" {znak | "'"} "'".

18. odbiorca-komunikatu-binarnego = odbiorca-komunikatu-unarnego |
                                    wysłanie-komunikatu-binarnego.

19. odbiorca-komunikatu-unarnego = wyrażenie-proste |
                                   wysłanie-komunikatu-unarnego.

20. oznaczenie-metody-pierwotnej = "<" "primitive:" cyfry ">".

21. selektor-binarny = "-" | znak-specjalny [znak-specjalny].

22. selektor-kluczowy = klucz {klucz}.

23. selektor-unarny = identyfikator.

24. stała = liczba | stała-znakowa | napis | stała-symboliczna |
            stała-tablicowa.

25. stała-symbolicza = "#" symbol.

26. stała-tablicowa = "#" tablica.

27. stała-znakowa = "$" {znak | " "}.

28. symbol = identyfikator | selektor-binarny | selektor-kluczowy.

29. szablon-metody = selektor-unarny |
                     selektor-binarny identyfikator |
                     klucz identyfikator {klucz identyfikator}.

30. tablica = "(" {liczba | stała-znakowa | napis | symbol |
                   tablica} ")".

31. wyrażenie = {zmienna ":="} (wyrażenia-proste |
                                wysłanie-komunikatu ("|" komunikat)).

32. wyrażenie-proste = stała | zmienna | blok | "(" wyrażenie ")".

33. wysłanie-komunikatu = wysłanie-komunikatu-unarnego |
                          wysłanie-komunikatu-binarnego |
                          wysłanie-komunikatu-kluczowego.

34. wysłanie-komunikatu-binarnego =
        odbiorca-komunikatu-binarnego komunikat-binarny.

35. wysłanie-komunikatu-kluczowego =
        odbiorca-komunikatu-binarnego komunikat-kluczowy.

36. wysłanie-komunikatu-unarnego =
        odbiorca-komunikatu-unarnego komunikat-unarny.

37. zmienna = identyfikator.

38. zmienne-robocze = "|" {identyfikator} "|".

39. znak = cyfra | litera | znak-specjalny |
           " " | "'" | "#" | "+" | "," | "." | "-" | "/" |
           ":" | ";" | "<" | "=" | ">" | "@" | "\" | "_" | "~".

40. znak-specjalny = "!" | "%" | "&" | "*" | "+" | "," | "/" | "?" |
                     "=" | "?" | "?" | "@" | "\" | "_" | "~".
```

Wydruk 3

Nowe książki

P. Schnupp, C. T. Nguyen Huu: Expertensystem, Springer-Verlag, 1987.

W książce opisano budowę systemów ekspertowych i programów użytkowych z bazami wiedzy, opierając się głównie na przykładach zaczerpniętych z systemów dostępnych na rynku komputerowym. Zamieszczone programy prologowe mogą posłużyć czytelnikom jako dobry punkt wyjściowy do budowy własnych systemów. Przykłady te ilustrują w szczególności takie elementy systemów ekspertowych jak: wyjaśnianie przebiegu rozumowania, optymalizacja heurystyczna i reprezentacja wiedzy za pomocą ram. Książka może posłużyć zarówno jako podstawa samodzielnych studiów, szkoleń, seminariów i wykładów, jak również jako kompendium wiedzy przydatnej w budowie systemów ekspertowych.

S. C. Shapiro (Ed.): Encyclopedia of Artificial Intelligence, John Wiley, 1987.

Jest to kompendium wiedzy z zakresu szeroko pojętej sztucznej inteligencji. W realizacji przedsięwzięcia wzięło udział 200 badaczy i 180 recenzentów. Encyklopedia zawiera ponad 450 ilustracji oraz ponad 5000 odsyłaczy do literatury. Wszystkie hasła zostały starannie poindeksowane w układzie tematycznym. Na podkreślenie zasługuje interdyscyplinarny charakter encyklopedii. Encyklopedia obejmuje następujące dziedziny: chemia, fizyka i optyka, filozofia, psychologia, informatyka, lingkwistyka i gramatyka, muzyka, medyczne systemy doradcze i protetyka, edukacja i heurystyka, rozwiązywanie problemów i rozpoznawanie obrazów, planowanie i procesy twórcze, teoria gier i teoria podejmowania decyzji, robotyka i automatyka przemysłowa. uczenie się maszynowe i przyswajanie pojęć, systemy ekspertowe i systemy regułowe, interakcja człowiek-komputer, zagadnienia społeczne oraz sztuczna inteligencja.

JAN BIELECKI
Instytut Informatyki
Politechnika Warszawska

Język C

Coraz bliżej normy (2)

W bieżącym numerze kontynuuję omówienie nowego projektu normy ANSI języka C.

Typ void

Nie istnieją dane typu (void). Konwersja (void) jest używana jedynie, gdy chodzi o jawne odrzucenie danej, albo wtedy, gdy jest deklarowana funkcja nie udostępniająca rezultatu. Słowo kluczowe void może także występować w miejscu pustej listy parametrów funkcji.

```
#include <stdio.h>
main ()
{
  printf ("To be or not to be");
  wait ();
}
void wait (void)
{
  while (!kbhit ());
  (void) getch ();
}
```

Wykonanie funkcji wait powoduje oczekiwanie na wprowadzenie z konsoli dowolnego znaku i pominięcie go. Wywołanie funkcji getch poprzedzono jawnym operatorem konwersji w celu podkreślenia, że rezultat funkcji zostanie odrzucony.

Typ void *

Dane typu (void *) są wskazaniami adresowymi. W odróżnieniu od pozostałych danych wskazujących, wskazują one nie obiekty, lecz lokalizują obszary pamięci operacyjnej. Ponieważ dane typu (void) nie istnieją, wyrażenie typu (void *) nie może być argumentem operatora wyłuskania.

```
#include <stdio.h>
main ()
{
  void *calloc (unsigned,unsigned);
  void *Ref = calloc (3,1) ;
  ((char*)Ref) [0] = 'j';
  ((char*)Ref) [1] = 'b';
  printf ("%s", (char*)Ref);
}
```

Wykonanie programu powoduje wyprowadzenie napisu jb. Rezultatem funkcji callos jest wskazanie adresowe.

Identyczność typów

Dwa typy są identyczne, jeśli są zgodne tożsamościowo. Typy zgodne tożsamościowo są również zgodne strukturalnie (ale nie odwrotnie).

```
typedef
  struct CPLX{
    double Re,Im;
  } COMPLEX;
struct CMPLX{
  double Re,Im;
};
```

Typ (struct CPLX) jest identyczny z typem (COMPLEX) ale nie jest identyczny z typem (struct CMPLX). Typy (struct CPLX) i (struct CMPLX) są zgodne strukturalnie, ale nie są zgodne tożsamościowo.

Deklaracje przesłaniające

Deklaracja o postaci:

```
struct tag;
albo
union tag;
```

w której tag jest oznacznikiem, powoduje przesłonięcie deklaracji oznacznika struktury lub unii i zatrzymanie jej zasięgu na średniku przytoczonej deklaracji.

```
#include <stdio.h>
struct STR2 {
  char Field;
};
main ()
{
  struct STR2;
  struct STR1 {
    struct STR2 *Ref;
  } Str;
  struct STR2 {
    long Field;
  };
  printf ("%d", sizeof *Str.Ref);
}
```

Wykonanie programu powoduje wyprowadzenie liczby o wartości sizeof (long). Gdyby z programu usunięto deklarację zatrzymującą, to nastąpiłoby wyprowadzenie liczby o wartości sizeof (char).

Pola bitowe

Pole bitowe może być typu (signed int) albo (unsigned int). O tym, czy pole bitowe typu (int) jest typu (signed int) czy typu (unsigned int), decyduje implementacja.

```
#include <stdio.h>
union{
  unsigned Pos : 1;
  signed Neg : 1;
} Fix;
main ()
{
  Fix.Pos = 1;
  printf ("%d", Fix.Neg);
}
```

Wykonanie programu powoduje wyprowadzenie liczby —1.

Operatory

Zestaw operatorów został rozszerzony o jednoargumentowy operator + (plus). Użycie tego operatora zapobiega potraktowaniu wyrażenia takiego jak np.:

a + (b + c)

jak wyrażenie:

(a + b) + c

Rezultatem operacji + (plus) jest dana reprezentowana przez jej argument.

```
#include <stdio.h>
main ()
{
  double Big = 2e20,
      Small = 1e—5;
  printf ("%f", —Big + + (Big + Small));
}
```

Wykonanie programu powoduje wyprowadzenie liczby bliskiej 0. Gdyby z programu usunięto jednoargumentowy operator + (plus), to mogłoby nastąpić wyprowadzenie liczby bliskiej 0.00001.

Punkty charakterystyczne

Punktem charakterystycznym jest takie miejsce w programie, w którym ustają skutki uboczne. Punktami charakterystycznymi programu są: miejsce wywołania funkcji (po opracowaniu wszystkich jej argumentów), miejsca wystąpienia operatora koniunkcji (&&), alternatywy (||), warunku (?) i połączenia (,) oraz miejsce, w którym zakończono opracowywanie kompletnego wyrażenia nie będącego argumentem funkcji.

```
#include <stdio.h>
main()
{
  int i = 2,j = 1;
  printf ("%d%d", j = i + +, i + j);
}
```

Wykonanie programu powoduje wyprowadzenie jednego z napisów: 23, 32, 33. Wynika to m.in. stąd, że program zawiera tylko jeden punkt charakterystyczny, i o tym że zmiennej i przypisano daną o wartości 3, wiadomo dopiero tuż przed wykonaniem prologu funkcji printf, a więc już po opracowaniu wszystkich jej argumentów.

Wyrażenia strukturowe

Wyrażenie strukturowe reprezentuje strukturę albo unię. Ma ono zazwyczaj postać identyfikatora, ale może mieć również postać przypisania albo wywołania funkcji. Wyrażenie strukturowe może być jedynie lewym argumentem operatora strukturowego, prawym argumentem operatora przypisania i może być ujęte w nawiasy okrągłe.

```
#include <stdio.h>
struct COMPLEX{
  double Re,Im;
} Str = {—3.0,4.0};
main()
{
  struct COMPLEX Cut (struct COMPLEX);
  struct COMPLEX Temp;
  Temp = Cut (Str);
  printf ("%0.1f,%0.1f)", Temp.Re, Temp.Im);
}
struct COMPLEX
Cut (struct COMPLEX Par)
{
  Par.Re = fabs (Par.Re);
  Par.Im = fabs (Par.Im);
  return Par;
}
```

Wykonanie programu powoduje wyprowadzenie napisu (3.0,4.0).

Wywołanie funkcji ze zmienną liczbą argumentów

Funkcja, która może być wywoływana ze zmienną liczbą argumentów, charakteryzuje się tym, że w miejscu jej ostatniego parametru znajduje się parametr zwany trzykropkiem (zapisywany za pomocą wielokropka „..."). Zabrania się, aby taka funkcja została wywołana z liczbą argumentów mniejszą od liczby parametrów poprzedzających ten symbol.

Odwołania do argumentów reprezentowanych przez parametr trzykropek powinny odbywać się za pośrednictwem makrowywołania va_arg, którego pierwszym argumentem jest nazwa pomocniczej zmiennej typu va_list, a drugim jest nazwa typu argumentu. Pierwsze (w danej funkcji) odwołanie do makrodefinicji va_arg powinno być poprzedzone wywołaniem makrodefinicji va_start, a po ostatnim powinno występować makrowywołanie va_end. Pierwszym argumentem va_start i va_end jest nazwa wspomnianej zmiennej pomocniczej, a drugim argumentem va_start jest nazwa tego parametru funkcji, który poprzedza parametr trzykropek. Wszystkie omówione makrodefinicje są zdefiniowane w zbiorze nagłówkowym stdarg.h.

```
#include <stdio.h>
enum Type {Char,Int,Float} ;
main()
{
  void Out (int,enum Type, ...);
  Out (3,Char,'E','w','a');
  Out (2,Int,1,3);
  Out (1,Float,4.8f);
}
#include <stdarg.h>
void
Out (int Count, enum Type Sort,...)
{
  va_list Area;
  va_start (Area,Sort);
  printf ("(");
  while (Count——)
    switch (Sort){
      case Char:
        printf ("%c", va_arg (Area,int));
        break;
      case Int:
        printf ("%d", va_arg (Area,int));
        break;
      case Float:
        printf ("%f", va_arg (Area,double));
    }
  va_end (Area);
}
```

Wykonanie programu powoduje wyprowadzenie napisu:
Ewa
13
4.800000

LITERATURA

[1] Bielecki J.: Język C — interpretacja standardu. WNT, Warszawa 1987
[2] Bielecki J.: Wprowadzenie do języka C. WNT, Warszawa, 1988
[3] Bielecki J.: Oprogramowanie mikrokomputerów. WKiŁ, Warszawa, 1987
[4] Bielecki J.: Turbo C. Numerica, Warszawa, 1987
[5] Bielecki J.: Turbo C dla programistów. WKiŁ, Warszawa, 1988
[6] Bielecki J.: Sam na sam z językiem C. WKiŁ, Warszawa, 1988
[7] Bielecki J.: Język C w systemie CP/M. WKiŁ, Warszawa, 1988

Nowe książki

G. K. H. Pang, A. G. J. MacFarlane: An Expert System Approach to Computer-Aided Design of Multivariable Systems, Springer-Verlag, 1987.

Książka jest adresowana do czytelników zainteresowanych wykorzystaniem technik systemów ekspertowych do analizy i projektowania systemów sterowania. Omówione podejście opiera się na metodach uogólnionej funkcji odpowiedzi częstotliwościowej. Zdefiniowano pełny zestaw wskaźników, za pomocą których można ocenić podstawowe cechy systemu ze sprzężeniem zwrotnym, tzn. jego stabilność, wydajność i odporność na zakłócenia. Opisano prototypowy system ekspertowy wspomagający projektowanie systemów sterowania. Wiedza dotycząca procesu projektowania została zdefiniowana z wykorzystaniem reprezentacji ram.

Uprzejmie informujemy Czytelników, że egzemplarze INFORMATYKI — bieżące i archiwalne — można kupić nie tylko w kioskach Ruchu, Klubie NOT SIGMY, Zakładzie Kolportażu i Dziale Handlowym (szczegóły podano w WARUNKACH PRENUMERATY), ale również

w lokalu naszej redakcji
ul. Mickiewicza 18 m. 17 w Warszawie, tel. 39-14-34

oraz

w specjalistycznej księgarni PP „Domu Książki"
ul. Mokotowska 51/53 w Warszawie, tel. 28-16-14

Zapraszamy wszystkich zainteresowanych.

ANDRZEJ WARDA
Instytut Maszyn Matematycznych
Warszawa

Pamięci dyskowe Mazovii (2)

W drugiej części artykułu opisano sterowanie dyskiem stałym (ang. hard disk).

DYSK STAŁY

Dysk stały jest pamięcią masową o dostępie swobodnym. W jednostce dyskowej są dwa sztywne, niewymienialne, 5-calowe talerze. Każda powierzchnia wymaga oddzielnej, ruchomej głowicy, obsługującej 306 ścieżek. Całkowita, sformatowana pojemność czterech powierzchni dysków wynosi 10 megabajtów. Na tę pojemność składają się 1224 ścieżki, zawierające po 17 sektorów; w każdym sektorze można umieścić 512 bajtów danych. Pozostałe ważniejsze parametry wymieniono w tabeli 1.

Tabela 1. Parametry napędu dysku stałego stosowanego w Mazovii 1016

Nazwa parametru	Wartość
Gęstość ścieżek	345 ścieżek/cal
Czas dostępu	3 ms (ze ścieżki na ścieżkę)
Czas dostępu na ścieżce	8,33 ms
Stopa błędów:	
— korygowalny błąd odczytu	$1/10^{10}$ bitów odczytywanych
— niekorygowalny błąd odczytu	$1/10^{12}$ bitów odczytywanych
— błąd pozycjonowania	$1/10^{6}$ szukanych ścieżek
Prędkość obrotowa dysku	3600 obr/min ±1%
Szybkość transmisji	5 Mb/s
Metoda kodowania	MFM
Zasilanie	+12 V ±5% — 1,8 A (maksymalnie 4,5 A)
	+5 V ±5% — 0,7 A (maksymalnie 1,0 A)

Zwarta obudowa chroni głowicę, dyski i mechanizm napędzający przed uszkodzeniami mechanicznymi i przed wpływem zanieczyszczeń. Wbudowany system recyrkulacji wymusza przepływ powietrza oczyszczonego przez filtr. Filtr ten eliminuje zanieczyszczenia o średnicy powyżej 0,3 μm. Termiczna izolacja silnika krokowego i silnika napędzającego dyski od komory dysków zapewnia dużą stabilność temperatury wewnątrz komory dysków. Umożliwia to poprawną pracę dysku stałego bezpośrednio po włączeniu zasilania, bez konieczności stosowania opóźnień na stabilizację termiczną urządzenia.

STEROWNIK DYSKU STAŁEGO

Sterownik dysku stałego może sterować maksymalnie dwiema jednostkami napędów dysków stałych, dołączonych płaskim kablem. Jest wyposażony w pamięć buforową i wykorzystuje bezpośredni dostęp do pamięci komputera. Jest też wykorzystywane przerwanie, które sygnalizuje zakończenie operacji i możliwość odczytania stanu sterownika.

W sterowniku dysku stałego jest zastosowana kontrola i korekcja błędów ECC, wykorzystująca do tego celu 32 nadmiarowe bity, dołączone do pola danych. Na pakiecie sterownika znajduje się pamięć stała, traktowana jako część BIOS-a, w której znajduje się program sterujący.

Sterownik jest oparty na zintegrowanym mikroukładzie WD2010 (firmy Western Digital). Do prekompensacji danych przy zapisie i separacji danych przy odczycie zastosowano układ WD10C20. Do sterowania pracą całego pakietu i do celów diagnostycznych zastosowano mikroprocesor jednoukładowy Intel 8035.

PROGRAMOWANIE

Sterownik dysku stałego ma dwa rejestry dostępne dla głównego procesora: rejestr stanu i rejestr danych. Ośmiobitowy rejestr stanu zawiera informacje o stanie sterownika i jest dostępny w dowolnej chwili. Ośmiobitowy rejestr danych składa się z kilku rejestrów ułożonych w stos, z jednym tylko rejestrem bezpośrednio dostępnym w określonych momentach. W rejestrze tym są przechowywane dane, kody rozkazów, parametry, a także można z niego pobrać niektóre informacje o stanie sterownika. Dane są odczytywane z rejestru danych lub zapisywane do niego w celu zaprogramowania operacji lub otrzymania wyników po wykonaniu danej operacji. Rejestr główny stanu można tylko odczytywać i służy on do kontrolowania poprawności wykonania operacji.

Rejestr danych

Jednostka sterująca inicjuje operację przez przesłanie do rejestru danych 6-bitowego bloku danych (DCB, ang. device control block). W tabeli 2 przedstawiono układ tego bloku i znaczenie poszczególnych bajtów.

Rejestr stanu

Po zakończeniu każdej operacji sterownik dysku stałego przesyła do systemu operacyjnego bajt stanu. W bajcie tym jest zawarta informacja, czy operacja została zakończona poprawnie, czy też wystąpił błąd. W tabeli 3 przedstawiono format tego bajtu. Jeżeli są dozwolone przerwania, to sterownik wysyła przerwanie, gdy jest gotowy do przesłania bajtu stanu i usuwa sygnał zajętości.

Jeżeli w rejestrze stanu jest sygnalizowany błąd (bit 1 jest ustawiony), to sterownik dysków podaje kolejne 4 bajty opisujące rodzaj błędu. Format tych czterech bajtów przedstawiono w tabeli 4.

Zestawienie wymagań programowych

Dwa najmniej znaczące bity szyny adresowej są doprowadzone do dekodera portów we-wy, który ma dwie czę-

Tabela 2. Struktura bloku danych

Bity	7	6	5	4	3	2	1	0
Bajt 0	Klasa rozkazu			Kod rozkazu				
Bajt 1	0	0	Dysk	Numer głowicy				
Bajt 2	Numer cylindra			Numer sektora				
Bajt 3	Numer cylindra							
Bajt 4	Licznik przeplotów lub bloków							
Bajt 5	Pole kontrolne							

Tabela 3. Format bajtu stanu

Bity	Znaczenie
0, 2, 3, 4, 6, 7	Ustawienie w stan logiczny 0
1	Wskazuje, że podczas ostatniej operacji wydarzył się błąd
5	Wskazuje numer logiczny napędu dysku stałego

Tabela 4. Format bajtów opisu błędu

Bity	7	6	5	4	3	2	1	1
Bajt 0	Ważny adres	0	Typ błędu		Kod błędu			
Bajt 1	0	0	Dysk	Numer głowicy				
Bajt 2	Numer cylindra			Numer sektora				
Bajt 3	Numer cylindra							

Tabela 5. Adresy portów we-wy sterownika dysku stałego

Odczyt lub zapis	Adres portu	Funkcja
Odczyt	320H	Odczyt danych (ze sterownika do jednostki centralnej)
Zapis	320H	Zapis danych (z jednostki centralnej do sterownika)
Odczyt	321H	Odczyt stanu sterownika
Zapis	321H	Zerowanie sterownika
Odczyt	322H	Zarezerwowane
Zapis	322H	Generowanie impulsu wyboru sterownika
Odczyt	323H	Niewykorzystywane
Zapis	323H	Zapis wzorca do rejestru maski przerwań i DMA

ści. Jedna część jest uaktywniana przez sygnał odczytu we-wy (IOR), a druga przez sygnał zapisu we-wy (IOW). W rezultacie na pakiecie są dostępne cztery porty odczytu-zapisu.

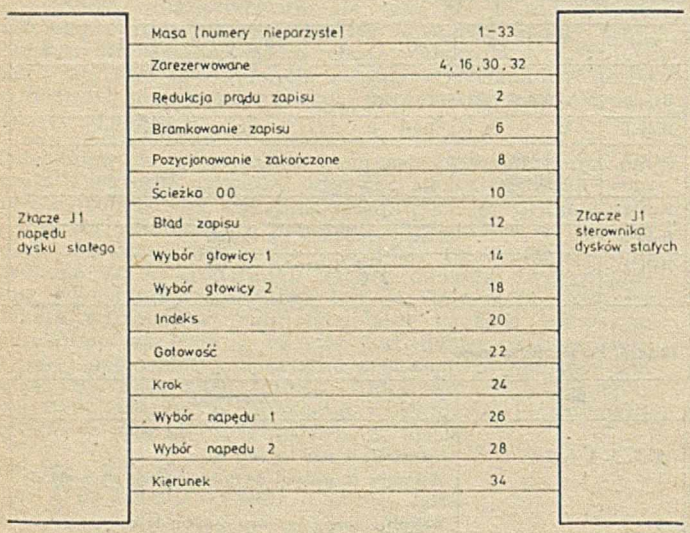
Sygnał zezwolenia adresu (AEN) jest wysyłany z pakietu procesora, gdy przesyłanie danych następuje przez kanał bezpośredniego dostępu do pamięci. Kiedy sygnał AEN jest aktywny, dekoder portów we-wy jest zablokowany.

W tabeli 5 przedstawiono dostępne porty we-wy.

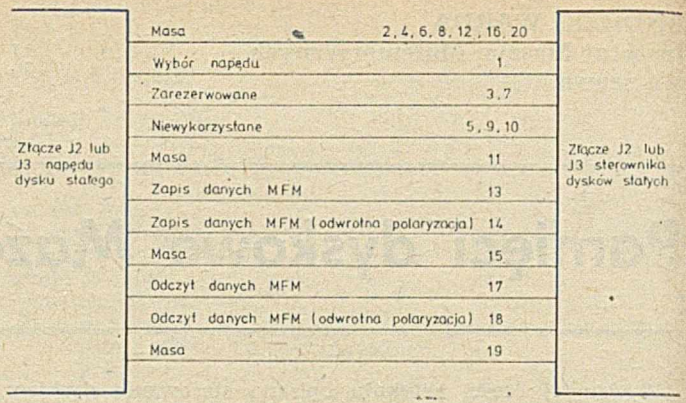
Lista rozkazów

Sterownik dysku stałego umożliwia wykonanie wielu elementarnych operacji, spośród których najważniejszymi są:

- formatowanie ścieżki,
- formatowanie złej ścieżki,
- zapis sektora,
- odczyt sektora,
- zapis sektora z dołączoną korekcją ECC do pola danych,
- odczyt sektora z dołączoną korekcją ECC do pola danych,



Rys. 1. Linie sprzęgu złącza J1



Rys. 2. Linie sprzęgu złącz J2 i J3

- odczyt stanu,
- ustawienie parametrów napędu,
- pozycjonowanie głowicy,
- testowanie gotowości napędu,
- wewnętrzne testowanie sterownika.

Na rysunkach 1 i 2 przedstawiono linie sprzęgu sterownika i napędu.

Ceny ogłoszeń

Od 1 lipca 1987 r. obowiązują następujące ceny materiałów reklamowych publikowanych na łamach INFORMATYKI:

Ogłoszenia

— ogłoszenia czarno-białe, artykuły reklamowe i informacje naukowo-techniczne (biuletyny) zależnie od objętości: cała strona — 50 tys., 3/4 str. — 45 tys., 2/3 str. — 40 tys., 1/2 str. — 35 tys., 1/3 str. — 30 tys., 1/4 str. — 25 tys., 1/8 str. — 20 tys., poniżej 1/8 str. — 200 zł za cm².
— ogłoszenia drobne (zależnie od liczby słów) jedno słowo — 50 zł.

Dodatki do ceny podstawowej:

— za każdy dodatkowy kolor + 30%,
— za każdy specjalny kolor (nie wynikający z podstawowych kolorów) + 30%,
— za pełny kolor (grafika wielobarwna, zdjęcia kolorowe) + 120%
— za zamieszczenie ogłoszenia na I lub IV stronie okładki + 100%
— za zamieszczenie ogłoszenia na II i III stronie okładki + 50%

Zniżki

dotyczą ogłoszeń — całkowitych powtórzeń
— za ogłoszenia 3—5-krotne — 10%
— za ogłoszenia 6—10-krotne — 20%
— za ogłoszenia 11-krotne i powyżej — 30%
— za artykuły i wkładki reklamowe wykonane przez zleceniodawcę — 40%
— za biuletyny i bloki reklamowe — 60%

W uzasadnionych wypadkach stosuje się zniżki specjalne dla ogłoszeń nie będących powtórzeniami — za zgodą Dyrektora — Naczelnego Redaktora Wydawnictw NOT SIGMA.
Ponadto Biuro Ogłoszeń świadczy usługi w zakresie wykonywania zdjęć czarno-białych i barwnych oraz nadbitek wkładek reklamowych.

Ceny wkładek

— wkładka 2 str. o formacie A4
nakład do 500 egz. 20 tys. zł
nakład 500—1000 egz. 35 tys. zł
— wkładka 4 str. o formacie A4
nakład 500 egz. 40 tys. zł
nakład 500—1000 egz. 70 tys. zł

Przy wkładkach o nakładzie powyżej 1000 egz. stosuje się wielokrotność powyższych cen, a dodatki za kolory oblicza się jak dla ogłoszeń.

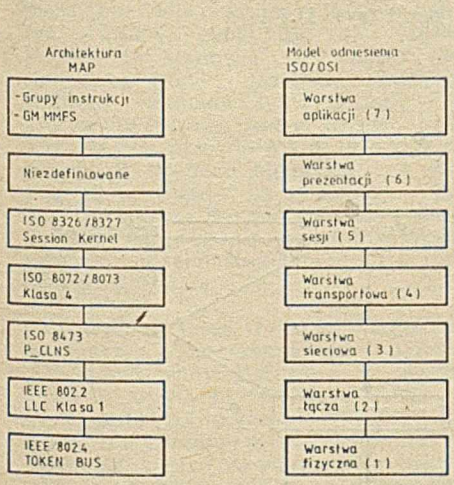
Ogłoszenia przyjmowane są przez:
Dział Ogłoszeń i Reklamy WCiKT NOT SIGMA
ul. Świętojerska 5/7, 00-236 Warszawa
adres do korespondencji: skrytka pocztowa 1004, 00-950 Warszawa
telefony: 31-93-65 lub 31-22-21 w. 196 i 291

Co to jest MAP?

Chociaż obecnie w produkcji panują „sztywne" linie montażowe, to celem fabryki 2000 roku jest połączenie komputerów i automatów w elastyczne zadania produkcyjne, mogące tworzyć ze sobą prawie dowolne kombinacje. Owe elastyczne zadania produkcyjne wymieniają ze sobą informacje i przedmioty obrabiane oraz przekazują do komputerów sterujących produkcją i montażem, na przykład informacje o załóceniach w produkcji lub dane eksploatacyjne.

Stanowiska produkcyjne lub automatyzacyjne, odpowiadające obecnemu stanowi techniki, umożliwiają wymianę danych jedynie w bardzo ograniczonym zakresie. O ile w dziedzinie sprzęgów sprzętowych stosowane są powszechnie normy międzynarodowe (np. V24), to prawie każdy producent oferujący wyroby do komputerowo zintegrowanego wytwarzania (ang. Computer Intergrated Manufacturing, CIM) opracowuje własne protokoły komunikacyjne. W wyniku tego, stworzenie sieci w całej fabryce jest możliwe tylko wtedy, gdy wszystkie urządzenia zostały zakupione i zainstalowane przez jednego producenta. Jednak z powodu różnej funkcjonalności urządzeń (np. urządzenia sterowania procesem technologicznym, różnią się od wyposażenia komputerów) jest to możliwe jedynie w ograniczonym stopniu.

Problem ten doprowadził w 1980 roku do połączenia, pod kierownictwem najbardziej wpływowego użytkownika — firmy General Motors, siedemnastu wielkich producentów z przemysłu komputerowego i urządzeń automatyki (Allen Bradley, DEC, IBM, Siemens i in.), w celu opracowania architektury komunikacji w dziedzinie produkcji przemysłowej oraz w celu ustalenia protokołów transmisji (ang. manufacturing automation protocol, MAP). Celem projektu MAP jest umożliwienie wymiany danych między różnymi systemami przy uwzględnieniu istniejących norm i zaleceń Międzynarodowej Organizacji Normalizacyjnej ISO.



Hierarchia protokołów MAP w porównaniu z modelem ISO/OSI

W 1982 roku opublikowano pierwszą oficjalną specyfikację MAP, którą 31 marca 1985 roku zastąpiła wersja 2.1 [3]. Nowsza wersja powinna być wiążąca i jednoznaczna — co najmniej dla zakładów General Motors — przez okres dwóch lat, aby w ten sposób uniknąć również niebezpieczeństwa błędnych inwestycji w przemyśle oprogramowania, wskutek ciągłych zmian specyfikacji MAP.

ARCHITEKTURA PROTOKOŁÓW MAP

Architektura protokołów MAP jest oparta na modelu odniesienia ISO/OSI [4], który składa się z siedmiu poziomów (fizyczna warstwa transmisji, łącza, sieciowa, transportowa, sesji, prezentacji oraz aplikacyjna). Na rysunku zestawiono model odniesienia ISO/OSI z architekturą MAP.

Fizyczna warstwa transmisji jest oparta na szerokopasmowej magistrali TOKEN BUS (zgodnie z normą IEEE 802.4), a warstwa łącza — na protokole LLC Class 1, zgodnie z normą IEEE 802.2. W dokumentacji MAP bardziej szczegółowo wyspecyfikowane są jedynie warstwy od trzeciej do siódmej.

Warstwa sieciowa steruje transmisją pakietów danych, zgodnie z zaleceniem ISO DIS 8473 „Data Communications Protocol for Providing the Connectionless-Mode Network Service" (w skrócie — P_CLNS).

W warstwie transportowej odpowiedzialnej za połączenie procesów stosowany jest protokół transportowy ISO Klasa 4 (ISO 8072/8073). Według ISO rozróżnia się pięć różnych klas protokołów (klasa 0 ... klasa 4), przy czym klasa 4 jako najsilniejsza obejmuje wykrywanie i usuwanie błędów.

Zadaniem warstwy sesji jest sterowanie przebiegiem komunikacji. MAP opiera się w tej warstwie na minimalnej konfiguracji ISO Session Kernel (ISO 8326/8327) dla pracy dupleksowej.

Kolejna warstwa prezentacji w architekturze MAP jest nieznormalizowana. Z powodu rozpowszechnienia monitorów graficznych, które są już prawie standardowym wyposażeniem urządzeń sterowania numerycznego za pomocą komputerów (CNC), można spodziewać się, że w niedalekiej przyszłości protokoły MAP zostaną rozszerzone w kierunku prezentacji obrazów.

PROTOKOŁY APLIKACYJNE MAP

Podstawą dla protokołów aplikacyjnych specyficznych dla MAP są dokumenty „Common Application Service Elements" określone przez ISO (DP 8649/8650). Zapewniają one tę samą funkcjonalność pod względem usług, jaka występuje dla warstwy sesji, przy czym usługi są bezpośrednio przeniesione z warstwy 7 na warstwę 5. Na tę funkcjonalność składają się następujące elementy [2]:

- zestawienie połączenia (warstwa 5: S-CONNECT; warstwa 7: ASSOCIATE),
- transmisja danych (S-DATA; A--TRANSFER),
- zwykłe rozłączenie połączenia (S--RELEASE, A-RELEASE),
- przerwanie połączenia przez jednego użytkownika (S-U-ABORT; A-U--ABORT),
- przerwanie połączenia przez świadczącego usługę (S-P-ABORT; A-P--ABORT).

Właściwym zadaniem projektu GM--MAP było umożliwienie sterowania i nadzoru nad komputerami oraz urządzeniami technologicznymi (ang. Robotic control, RC; numerical control, NC; programmable logic control, PLC). W celu zdefiniowania funkcji niezbędnych dla użytkownika, w procesie produkcyjnym stworzono własną składnię tzw. MMFS (ang. manufacturing message format standard). Jedną ze spodziewanych zmian specyfikacji MAP jest to, że nastąpi odejście od składni MMFS i funkcje technologiczne określone w protokole zostaną opisane według notacji ASN.1 (ang. abstract syntax notation number one) ustalonej przez ISO i prawie identycznej z zaleceniem X.409 ustalonym przez CCITT.

Grupa pierwsza funkcji dla automatów sterowanych numerycznie

Funkcja	Opis
CYCLE START	Aktywacja lub zakończenie aktualnego cyklu maszynowego
PART	Identyfikacja poszczególnych przedmiotów obrabianych
AXIS OFFSET	Manipulacja osiowa (np. przy sterowaniu robotów)
UNITS	Definicja określająca stosowane jednostki
EXCHANGE	Manipulacja paletami z przedmiotami obrabianymi
LIFT	Podniesienie określonego urządzenia

Specyfikacja technologicznych protokołów aplikacyjnych, opracowana pod nazwą EIA 1393A przez Electronic Industries Association [1], dzieli funkcje na różne tzw. grupy instrukcji (ang. instruction groups). Na przykład w grupie pierwszej, dla automatów sterowanych numerycznie, definiuje się m.in. funkcje przedstawione w tabeli. Inne ważne funkcje zdefiniowane w dokumencie EIA 1393A dotyczą ładowania skrośnego (Upload/Download) programów do sterowania numerycznego oraz operacji na plikach (Transmit, Select, Queue itp.).

WŁAŚCIWOŚCI DOTYCZĄCE CZASU RZECZYWISTEGO

Protokoły technologiczne MAP są protokołami warstwy aplikacyjnej. Każdy pakiet protokołów MAP musi więc być dopasowany do wszystkich warstw komunikacyjnych, w wyniku czego szybkie działania, jakie są potrzebne, na przykład w sterowaniu procesami, nie mogą zostać wykonane.

To ograniczenie, tzn. brak właściwości charakterystycznych dla systemów czasu rzeczywistego, zostanie prawdopodobnie usunięte w następnej specyfikacji MAP oczekiwanej w 1988 roku. W szczególności powinny zostać uwzględnione prace grupy PROWAY, opracowującej i badającej protokoły komunikacyjne w aspekcie wymagań systemów czasu rzeczywistego. Bez możliwości, jakie daje szybki czas dostępu oraz szybkości transmisji typowe dla systemów czasu rzeczywistego, MAP byłby stosowany jedynie jako podrzędna sieć do transmisji dużych ilości danych.

Pomito tego, że architektura MAP opiera się na magistrali TOKEN BUS umożliwiającej transmisje o różnych priorytetach, to MAP nie przewiduje żadnych priorytetów. Również funkcja transmisji danych EXPEDITED przewidziana standardowo w protokołach ISO do stosowania co najmniej dwóch klas priorytetów, nie jest uwzględniona w specyfikacji MAP 2.1.

ZNACZENIE MAP

Pomimo poważnych braków obecnie ograniczających zakres stosowalności protokołów MAP przewidzianych dla całej sfery produkcji, przewiduje się, że MAP będzie szeroko stosowany w przyszłości. O ile w dziedzinie „otwartej komunikacji" konkuruje ze sobą wiele koncepcji, np. model odniesienia ISO/OSI z systemem SNA firmy IBM, to w dziedzinie produkcji przemysłowej istnieją dytychczas jedynie propozycje protokołów MAP. Wprawdzie w ramach projektu ESPRIT, koordynowanego i wspieranego przez Komisję EWG, prowadzone są również prace nad opracowaniem protokołów do zastosowań w przemyśle, pod nazwą CNMA (ang. communication networks for manufacturing application), to jednak z powodu pozycji rynkowej, jaką zajmują firmy i użytkownicy uczestniczący w projekcie GM-MAP, to nowe propozycje nie mogą ograniczyć jego dominacji.

Implementacje GM-MAP obejmujące całą sferę produkcji oraz technologii, jeszcze nie istnieją. Jednakże w wielu ośrodkach prowadzone są intensywne prace w tej dziedzinie. Dotychczasowe częściowe implementacje przeprowadzone głównie w zakładach koncernu General Motors bazowały jeszcze na przestarzałej specyfikacji MAP 2.0.

Opracował
A. RAZNOWIECKI

LITERATURA

[1] Electronic Industries Association: Manufacturing Message Service for Bidirectional Transfer of Digitally Encoded Information. EIA Working Document 1393A, October 1985

[2] Fong K. L., Amaranth P. A.: MAP Application Layer Interface and Application Management Structure. Comput. Commun. Rev., Vol. 15, No. 2, April 1985, No. 3, July 1985

[3] General Motors: MAP Specification Version 2.1, March 1985

[4] International Organization for Standardization: Information Processing Systems — Open Systems Interconnection — Basic Reference Model ISO 7498. November 1983

Podejście do tworzenia kompilatorów Ady

W ramach europejskiego projektu kompilatora Ady (realizowanego przez firmy Alsys, Bull i Siemens) napisano translator programów w języku Ada na język PL/I. Program ten pozwolił na implementację kompilatora Ady w języku będącym podzbiorem Ady. Dopuszczalny dla translatora podzbiór zawierał jedynie klasyczne konstrukcje programowe, nie obejmując m.in. obsługi sytuacji wyjątkowych, mechanizmów czasu rzeczywistego i jednostek rodzajowych. Poniżej opisano implementowany podzbiór języka, strukturę translatora oraz sposób użycia translatora do realizacji kompilatora Ady.

Stworzenie kompilatora dla języka Ada jest problemem złożonym. Jednym z głównych celów jego autorów było zapewnienie przenośności większości modułów kompilatora. Starano się również o przybliżenie twórcom kompilatora charakterystycznych właściwości języka. Dlatego zdecydowano się na napisanie kompilatora w języku Ada, po uprzednim stworzeniu programu tłumaczącego oprogramowanie z Ady na język zaimplementowany na komputerze użytym do realizacji projektu.

Zaimplementowany podzbiór języka Ada był zdeterminowany przez możliwości języka GPL (jest to podzbiór PL/I dostępny na komputerach Bull-DPS7) oraz przez potrzeby kompilatora. Do napisania kompilatora nie była potrzebna implementacja jednostek rodzajowych, typów zadaniowych i specyfikacji reprezentacji. Z kolei, w GPL niemożliwe jest odwzorowanie liczb rzeczywistych oraz implementacja procedur obsługi sytuacji wyjątkowych. Niektóre cechy charakterystyczne Ady zostały zaimplementowane częściowo, np. w czasie operacji przepisywania nie jest wykonywane żadne badanie spełniania zawężeń.

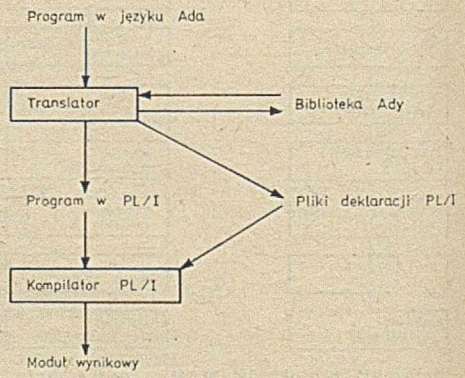
Powstały translator jest w rzeczywistości zredukowaną formą kompilatora. Fakt ten ułatwił weryfikację niektórych koncepcji. W szczególności, oddzielenie fazy kompilacji od translacji z Ady na PL/I pozwoliło na zweryfikowanie znacznej części kompilatora. Tekst źródłowy kompilatora okazał się przy tym dobrym tekstem dla samego kompilatora. Przypuszczalnie było to najpoważniejsze zastosowanie języka Ada do końca 1984 roku.

STRUKTURA TRANSLATORA

Translacja programu w języku Ada przebiega w trzech etapach. Pierwszą fazę stanowi analiza syntaktyczna programu i konstruowanie postaci pośredniej AIL (ang. abstract intermediate language), która jest postacią wejściową dla pozostałych dwu faz.

W czasie drugiej fazy, dla każdej deklaracji napotkanej w tekście programu tworzona jest tablica symboli, a każdemu odwołaniu przyporządkowane jest wskazanie na odpowiadający mu element w tablicy symboli.

Trzecia faza jest aktywowana, gdy całe poddrzewo (może być nim deklaracja, instrukcja prosta lub instrukcja złożona) jest całkowicie zidentyfikowane. Wówczas generowany jest odpowiadający mu tekst w języku PL/I. Plik z wygenerowanym tekstem jest następnie poddany kompilacji za pomocą kompilatora PL/I, łączone są kody wynikowe poszczególnych jednostek, a utworzony kod wynikowy programu jest przekazywany do konsolidacji (rys. 1).



Rys. 1. Translacja programu w języku Ada

Te trzy fazy mogą operować dużymi zbiorami danych, dlatego dla wszystkich faz wspólne są programy obsługi. Są to przede wszystkim programy zarządzające pamięcią wirtualną i umożliwiające w ten sposób translację obszernych programów w ograniczonym obszarze pamięci. Zarządzają one także strukturą drzewa programu oraz tablicy symboli. Rozłączna kompilacja jednostek programowych wymaga przechowywania między fazami pewnych informacji, np. tablicy symboli lub zależności między poszczególnymi jednostkami. Wszystkie te informacje są przechowywane w bibliotece Ady. W celu odzwierciedlenia w PL/I mechanizmu rozłącznej kompilacji zachowuje się również pliki z tekstem w języku PL/I, będące wynikiem translacji deklaracji widocznych przez inne jednostki programowe.

W każdym z opisanych etapów możliwe jest oczywiście wykrywanie błędów w translowanym programie. W wypadku napotkania błędu, biblioteka Ady nie jest modyfikowana. W przeciwnym razie, z biblioteki ładowana jest następna jednostka do translacji. W wypadku powtórnej kompilacji niszczone są wszystkie podjednostki.

Mechanizm pamięci wirtualnej zaimplementowano w celu umożliwienia zarządzania dużymi zbiorami danych translatora. Każdy obiekt ze zbioru obiektów wirtualnych ma swój adres wirtualny. Obiekty są grupowane w tzw. strony stałej wielkości. Odwołanie się do obiektu powoduje, w wyniku zdekodowania jego adresu wirtualnego, identyfikację strony, do której należy dany obiekt. Jeżeli strona ta nie jest zapisana w żadnym z buforów programu zarządzającego zasobami wirtualnymi, to jest ona odczytywana z pliku. Strona, która znajdowała się w buforze, jest przedtem powtórnie zapisywana w pliku, jeżeli była modyfikowana.

Znaczna część analizatora syntaktycznego została utworzona przy użyciu narzędzia SYNTAX, będącego produktem francuskiego instytutu INRIA (Institut National de Recherche d'Informatique et d'Automatique). Na podstawie opisu gramatyki i semantyki SYNTAX produkuje zbiór tablic. Tablice te stanowią parametry dla programu, który następnie zostanie poddany analizie syntaktycznej i z którego utworzone będzie drzewo AIL. Elementy w drzewie mają adresy wirtualne. Z każdego elementu jest dostęp do elementu na poziomie o jeden wyższym, do pierwszego z elementów na poziomie o jeden niższym oraz do następnego elementu na tym samym poziomie.

W trzeciej fazie translacji przez rekurencyjne przeszukiwanie drzewa AIL generowany jest tekst w języku PL/I. W czasie translacji tworzone są — zależnie od potrzeb — jeden lub dwa pliki zawierające ten tekst. Pierwszy z nich zawiera zbiór deklaracji odpowiadających deklaracjom programu w Adzie widocznym z innych jednostek. Plik ten jest tworzony podczas translacji specyfikacji lub ciała podprogramu albo pakietu zawierającego wydzielone jednostki. Jest on zachowywany po zakończeniu translacji. Drugi plik zawiera procedurę PL/I, która zostanie przedstawiona kompilatorowi PL/I, a następnie usunięta. Na początku działania tej procedury dołączany jest zbiór plików deklaracji odpowiadający widoczności programu w języku Ada dla klauzul **with**.

Deklaracje w Adzie są reprezentowane za pomocą odpowiadających im deklaracji w PL/I: liczby całkowite 16- lub 32-bitowe, typ logiczny, znaki i wskazania na obiekty typów prostych. Typy złożone Ady są reprezentowane w PL/I jako tablice i zmienne bazowe. Obiekty widoczne dla innych jednostek są deklarowane jako **external**, pozostałe jako **internal**. Zmienne **external** są statyczne, tzn. są alokowane tylko raz dla jednego wykonania programu. Zmienne **internal** mają dodatkowy atrybut **static** lub **automatic**, oznaczający czy mają być alokowane tylko raz, czy też przy każdym wykonaniu zawierających je procedur. Klasa **static internal** jest zarezerwowana dla zmiennych w ciele pakietów, które są jednostkami bibliotecznymi, ale nie są zadeklarowane jako **separate**.

Translacja instrukcji i wyrażeń przebiega w dwóch etapach. Najpierw tworzone jest drzewo tekstu PL/I, odpowiadające danemu wyrażeniu, a dopiero potem jest ono zapisywane w pamięci. Takie rozdzielenie jest konieczne ze względu na możliwość wystąpienia w czasie przetwarzania wyrażenia zmiennych tymczasowych, których inicjowanie i wykorzystanie wartości odbywa się w różnych częściach tego wyrażenia.

Aby uniknąć konfliktów wynikłych z powtarzania się nazw identyfikatorów wprowadzono jednoznaczny sposób kodowania. Każdemu obiektowi Ady przyporządkowany jest jeden element tablicy symboli, odpowiadający jednemu adresowi wirtualnemu. Identyfikatory PL/I składają się z siedmiu znaków: pierwsza litera określa rodzaj obiektu, a sześć pozostałych znaków reprezentuje jednoznacznie adres wirtualny.

INNE NARZĘDZIA

Spośród narzędzi towarzyszących translatorowi warto wymienić trzy: zarządcę bibliotek, program wiążący i narzędzie analizy kompletności.

Zarządzanie bibliotekami Ady

Program zarządzający bibliotekami pozwala sprawdzać i modyfikować stan biblioteki Ady. Dopuszcza on również możliwość usuwania pewnych jednostek (wówczas wszystkie podjednostki są także usuwane). W omawianej implementacji możliwe jest współdzielenie jednostek między bibliotekami. Wprowadzono tu pojęcie więzi. Jeżeli jednostka **X** jest translowana w bibliotece **A**, to posiadacz biblioteki **B** może utworzyć więź z jednostką **X** przez wydanie określonego polecenia do programu zarządzania zasobami bibliotecznymi. Dalsze korzystanie z jednostki **X** odbywa się tak, jakby była ona translowana w bibliotece **B**. Na przykład, więź ze specyfikacją pakietu pozwoli użyć wywołania tego pakietu z użyciem klauzuli **with**, umożliwi także translację innej wersji ciała tego pakietu.

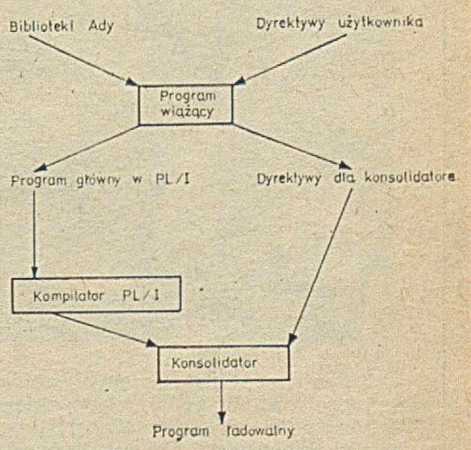
Sposób implementacji kompilatora można przedstawić schematycznie tak, że kompilacja przebiega w kilku fazach określonych przez oddzielne procedury. Sprzężeniem między poszczególnymi fazami, zapewniającym możliwość komunikacji między nimi, są pakiety. Dla sprzężenia i dla poszczególnych faz utworzone są biblioteki procedur. Istnieje także biblioteka zawierająca więź ze sprzężeniem i z przetranslowanymi modułami faz. Zbiór bibliotek jest charakterystyczny dla danej wersji kompilatora. Dodatkowo każdy implementator ma swoją własną bibliotekę, gdzie może tworzyć nową wersję któregoś z jej składników, bez konieczności narzucania jej innym użytkownikom.

Program wiążący

Program wiążący (ang. binder) ustala kolejność przetwarzania poszczególnych jednostek bibliotecznych na podstawie dyrektyw użytkownika. Wiązanie kończy się błędem, jeżeli graf zależności zawiera cykle. Na podstawie ustalonej kolejności program wiążący wytwarza program główny w języku PL/I, który zapoczątkuje opracowywanie pakietów i wywołuje program użytkowy. Generuje on także zbiór dyrektyw dla konsolidatora. Po zakończeniu kompilacji programu głównego wywoływany jest konsolidator systemowy, którego produktem jest moduł ładowalny (rys. 2).

Narzędzie analizy kompletności do testowania kompilatora

Narzędzie do analizy ma na celu sprawdzenie, czy każda linia kompilatora jest wykonana co najmniej raz.



Rys. 2. Tworzenie programu wykonywalnego w Adzie

Zasadą jego działania jest przyporządkowanie licznika każdemu ciągowi instrukcji (np. każdej gałęzi instrukcji IF). Podczas wykonywania programu w języku Ada (gdy następuje odwołanie do kompilatora) liczniki są inkrementowane. Po zakończeniu wykonywania wartości liczników są dodawane do wartości uzyskanych w czasie poprzednich wykonań programu. Oddzielny program usługowy pozwala następnie przeanalizować uzyskane wyniki i przedstawić je w czytelnej formie.

Przedstawione narzędzie przetestowano w czasie działania fazy analizy syntaktycznej kompilatora. Dzięki niemu wykryto ciągi instrukcji, które wcale nie były wykonywane (często były to ciągi instrukcji tworzące obsługę błędów) oraz ciągi instrukcji wykonywane najczęściej. Pozwoliło to na istotną optymalizację programu kompilatora. Taka analiza jedynie nieznacznie wydłuża czas wykonania programu.

Opracowała DOROTA INKIELMAN

na podstawie BIGRE, nr 42

Akademicka sieć superkomputerowa

National Science Foundation (amerykańska Krajowa Fundacja Nauki), działając na zlecenie rządu USA, zakończyła na przełomie lat 1986—1987 pierwszy etap budowy krajowej sieci superkomputerowej NSFNET. Sieć obejmuje sześć ośrodków obliczeniowych wyposażonych w najnowsze superkomutery, połączonych między sobą za pomocą szybkich linii przesyłowych (56 000 bitów/s). Ponad 50 uniwersytetów i instytucji badawczych w USA ma dostęp do NSFNET przez hierarchię sieci podrzędnych. Aktualnie kilka tysięcy projektów badawczych korzysta nieodpłatnie z mocy obliczeniowych oferowanych przez NSFNET.

Cztery z sześciu ośrodków są wyposażone w superkomputery Cray X-MP/48, najpotężniejsze komputery produkowane obecnie przez Cray Research Inc. Są one zlokalizowane w uniwersytetach San Diego (stan Kalifornia), Urbana-Champaign (Illinois) i Pittsburgh (Pensylwania) oraz w Narodowym Centrum Badań Atmosferycznych w Boulder (Kolorado). Piąty ośrodek w Princeton (New Jersey), aktualnie wyposażony w komputer CDC Cyber 205, otrzyma wkrótce superkomputer ETA-10. Szósty ośrodek w uniwersytecie Cornell (Nowy Jork) ma komputer IBM 3090-400 VF połączony równolegle z kilkoma procesorami macierzowymi FPS-164 i FPS-264.

National Science Foundation przeznaczyła ponad 200 milionów dolarów na rozwój ośrodków w ramach pięcioletniego programu budowy sieci NSFNET. Stanowi to około 60% całkowitych kosztów budowy. Pozostała część wydatków zostanie pokryta przez rządy stanowe i sektor prywatny, zwłaszcza przez firmy komputerowe.

Podstawowym zadaniem sieci jest udostępnienie naukowcom w całym kraju najnowocześniejszego i najszybszego sprzętu komputerowego. Przewiduje się ciągłe unowocześnianie sprzętu w wytypowanych ośrodkach. Już obecnie planuje się instalowanie nowych superkomputerów o mocy obliczeniowej komputera ETA-10 lub większej oraz zwiększenie szybkości linii przesyłowych do ponad miliona bitów/s. Wkrótce zostaną również zmodernizowane urządzenia peryferyjne, takie jak interakcyjne stacje graficzne oraz urządzenia pamięci masowej.

Oprac. M. MACHURA

na podst. „Supercomputer", styczeń 1987

Kto jest kim w IFIP

Zmarł profesor Tohru Moto-oka

Z żalem odnotowujemy śmierć jednego z twórców japońskiego Projektu Komputerów Piątej Generacji, profesora Tohru Moto-oki.

Profesor Moto-oka studiował elektronikę na Uniwersytecie Tokijskim, gdzie w 1958 roku obronił pracę doktorską. Tam też rozpoczął pracę, aż do śmierci był pracownikiem jednego z wydziałów tego uniwersytetu.

Jako szef Projektu Komputerów Piątej Generacji walnie przyczynił się do wytyczenia jego kierunków rozwojowych. Uczestniczył również w pracach japońskiego Projektu Superkomputerów do Zastosowań Naukowych, jak również w niezliczonej liczbie komitetów — zarówno japońskich, jak i międzynarodowych. W grudniu 1985 roku został pośmiertnie odznaczony przez cesarza Japonii orderem Największego Skarbu (Zuihosho).

Profesor Moto-oka był przedstawicielem Japonii w Komitecie Technicznym IFIP ds. Projektowania Systemów Cyfrowych i szefem programowym konferencji nt. Architektury Komputerów Piątej Generacji, która odbyła się w lipcu 1985 roku w Manchester. Przewodniczący Komitetu Technicznego ds. Projektowania Systemów Cyfrowych, prof. David Aspinall tak o nim napisał: „Profesor Moto-oka był wzorowym członkiem Komitetu, jego obecność wpłynęła na pełny sukces konferencji grupującej czołowych projektantów komputerów piątej generacji z Japonii, Ameryki i Europy". Profesor Moto-oka był również współprzewodniczącym Komitetu Programowego Siódmej Międzynarodowej Konferencji IFIP nt. Języków Opisu Sprzętu Komputerowego, która odbyła się w Tokio w sierpniu 1985 roku. Był też zaproszonym mówcą na Kongresie IFIP w 1983 roku.

Jego badania naukowe dotyczyły architektury komputerów, systemów rozproszonych, maszyn wnioskujących, przetwarzania równoległego, automatycznego projektowania układów cyfrowych i sztucznej inteligencji.

(MK)

Mikrokomputery z Domu Handlowego Nauki

Od kilku lat polski rynek przeżywa istną inwazję sprzętu mikrokomputerowego. Rozpoczęło się od domowych i osobistych zabawek ośmiobitowych. Od trzech lat przeważają komputery klasy IBM PC, początkowo XT, potem coraz większe AT, a wreszcie ostatnio — różnie zwane komputery z procesorem 386. Wiele firm operuje na tym rynku, pochodzenie i sposób sprowadzania sprzętu bywają rozmaite. Przeważa skup od osób prywatnych, a następnie sprzedaż jednostkom uspołecznionym, być może po pewnym uszlachetnieniu, polegającym głównie na przetestowaniu i obdarzeniu gwarancją. Inne firmy sprowadzają komputery na zasadzie transakcji wymiennych. Jeszcze inne (np. spółka „Mikrokomputery") sprowadzają układy elektroniczne i mechanizmy precyzyjne (np. dyski), a korzystają z zaprojektowanych i wykonanych w kraju płyt, obudów, zasilaczy itp.

Od 1986 roku na rynku mikrokomputerowym działa także Zakład Informatyki i Automatyzacji Badań Domu Handlowego Nauki, stanowiącego spółkę z przeważającym udziałem Polskiej Akademii Nauk. Organizatorzy Zakładu doszli do wniosku, że jedyną metodą zapewnienia odpowiedniej liczby mikrokomputerów klasy IBM PC dla jednostek Akademii jest hurtowy zakup od jednej, solidnej firmy z Dalekiego Wschodu, za pośrednictwem przedsiębiorstwa handlu zagranicznego. Przede wszystkim ogłoszono w Singapurze przetarg między ewentualnymi dostawcami. Oferenci musieli zapewnić odpowienio szeroki zestaw produktów — komputerów i urządzeń zewnętrznych, odpowiednią jakość i całkowitą zgodność z oryginałem IBM PC. Oczywiście, odgrywała też rolę cena i warunki dostaw. Najlepsza okazała się tajwańska firma „Multitech" (obecnie Accer), nie znana poprzednio na rynku polskim. Warto dodać, że kilka innych firm, popularnych na naszym rynku, brało udział w przetargu, ale nie potrafiły one spełnić żądanych warunków.

Równocześnie z wyborem sprzętu, rozpoczęto przygotowywanie zaplecza technicznego. Skorzystano z gościny zakładu Warmet na ul. Jubilerskiej w Warszawie. Trzeba było zorganizować magazyn (jeden transport obejmuje kilka kontenerów) i montownię. Komputery są bowiem dostarczane w częściach, a ich montaż, uruchamianie i testowanie odbywa się w Zakładzie Informatyki. Przygotowano też spójny system polskich liter. Klawiatury mają firmowo naniesione litery w miejscach, w miarę możliwości, zgodnych z krajową normą. Nakładka na system operacyjny (podobna do standardowych nakładek keybxx rozprowadzanych przez „Microsoft") przydziela tym klawiszom kody ze zbioru odpowiadającego znakom narodowym rozszerzonego kodu ASCI. Oczywiście kody te są także poprawnie interpretowane przez wszystkie rozprowadzane karty graficzne oraz przez drukarki. Większość oprogramowania akceptuje polskie litery na równi z angielskimi.

Obecnie oferta Zakładu Informatyki DHN obejmuje trzy rodzaje komputerów. Najprostszym jest **XT-Turbo**, z zegarem o częstotliwości przełączanej między 4,77 MHz a 8 MHz i pamięcią 640 KB na płycie głównej. Drugim jest **Acer 900**, zgodny z AT. Ostatnio, do oferty włączono komputer **Acer 1100**, korzystający z procesora iAPX 386 (80386) taktowanego zegarem o częstotliwości 16 MHz. Odpowiada on najnowszym tendencjom światowym — w zeszłym roku został uznany za najszybszy mikrokomputer tej klasy (zwanej w uproszczeniu Super-AT) na największej amerykańskiej wystawie komputerowej, a na targach w Hanowerze zdobył wyróżnienie.

Każdy komputer jest konfigurowany osobno, zgodnie z żądaniami kupującego. Standardowa jest tylko pełna pamięć na płycie głównej (640 KB w klasie XT, a 1 MB dla AT i AT-386). Koprocesor arytmetyczny (8087, 80287 lub 80387 w zależności od komputera) jest opcjonalny. Można wybrać kartę grafiki (grafika kolorowa — CGA, jednobarwna — Hercules, rozszerzona — EGA) i oczywiście pasujący do niej monitor: jednobarwny firmy BMC lub kolorowy (zwykły lub o zwiększonej rozdzielczości) firmy TVM. Jeśli chodzi o pamięć masową, to oferta obejmuje stacje dyskietek 360 KB i 1,2 MB oraz dyski 20 MB, a ostatnio także dyski 40 MB z czasem dostępu 28 ms i z silnikami liniowymi, sprowadzone głównie z myślą o modelu Acer 1100 i pracy wielodostępnej. Do archiwowania danych można zainstalować kasetową pamięć taśmową (ang. streamer) albo wymienne dyski elastyczne typu Bernoulli (10 MB pojemności). Dyski stałe i urządzenia do archiwowania są dołączane za pośrednictwem specyficznych kart, dostosowanych do szybkości i szerokości magistrali różnych modeli komputerów. Stacje dyskietek są dołączone do popularnej klasy disk I/O, dającej dodatkowo łącze równoległe i jedno lub dwa łącza szeregowe. Są także dostępne rozmaite inne karty: dodatkowe łącza szeregowe i równoległe, rozszerzenia pamięci (AT i AT-386; w tym ostatnim wypadku dodatkowa pamięć jest dostępna przez 32-bitową magistralę), przetworniki a/c i c/a, programatory EPROM-ów itp.

Oferta obejmuje również urządzenia zewnętrzne. Można kupić jedną z dwóch drukarek firmy Epson FX-800 (10 cali) lub FX-1000 (15 cali). Obie stanowią najnowszą generację konwencjonalnych, 9-igłowych, drukarek mozaikowych z pełnymi możliwościami graficznymi, kilkoma krojami pisma (także NLQ) i bardzo wygodnym mechanizmem prowadzenia papieru. Ostatnio pojawiły się drukarki 24-igłowe, a także drukarka laserowa (144 punkty na 1 mm kwadratowy). Na próbę sprowadzono również urządzenie do analizy obrazu, które teoretycznie może działać jako czytnik pisma — niestety, na razie okazało się, że jego sprawność, zwłaszcza dla tekstów z polskimi literami, jest za mała do zastosowań praktycznych. Do zastosowań w projektowaniu sprowadzono plotery i digitizery (firmy Roland i Logitech). Można też kupić myszkę.

Obecnie coraz większe zainteresowanie budzi korzystanie z mikrokomputerów w systemach wielodostępnych i w sieciach. Jeśli chodzi o to pierwsze, to należy sobie zdać sprawę z faktu, że architektura tych komputerów nie była opracowana z myślą o wielodostępie, a więc wszelkie rozwiązania będą zawsze miały charakter protez. Zakład Informatyki DHN oferuje wprawdzie sterowniki terminali produkowane w Polsce, do których można dołączyć standardowe monitory i klawiatury, ale potencjalny nabywca nie powinien oczekiwać od nich zbyt wiele.

Jeśli chodzi o sieci, to początkowo oferowano wyłącznie Transnet. Sieć ta spisuje się nieźle, jeśli łączy kilka komputerów, a korzystanie z niej obejmuje dorywcze przekazywanie kilku bloków danych. W samym Zakładzie Informatyki są nią połączone wszystkie zainstalowane komputery, korzysta się z niej także przy nagrywaniu oprogramowania na dyski stałe nowych komputerów. Transnet zawodzi jednak w większych, bardziej obciążonych konfiguracjach. Zdecydowano więc kupić standardową już obecnie sieć Ethernet, która powinna zaspokoić nawet najbardziej wyszukane potrzeby. W dużych konfiguracjach można będzie zresztą łączyć obie sieci, korzystając z oprogramowania Netware firmy Novell. To oprogramowanie wymaga przeznaczenia jednego komputera AT lub Super-AT na składnicę plików (ang. file server). Zasoby tego komputera (głównie pamięć dyskowa, ale także drukarki) mogą być dzielone między kilka stacji, często po kilka, a nawet kilkanaście. Na większe odległości sprowadzono modemy, zgodnie ze standardem Hayesa, na które otrzymano oficjalną homologację, pozwalającą na włączenie do ogólnodostępnej sieci telefonicznej.

W sumie, do tej pory sprzedano sprzęt do ok. 300 jednostek i instytucji, często po kilka a nawet kilkanaście mikrokomputerów. Ceny wahają się od 2 do 30 milionów zł za system komputerowy. Wśród klientów przeważają jednostki naukowo-badawcze (przeszło 60%, z tego prawie połowa to placówki PAN), reszta — to przemysł (15%), transport (9%) i inne.

Zakup sprzętu od jednego dostawcy, na podstawie rocznych kontraktów, ma liczne zalety. Sprzęt jest jednolity, wiadomo, jak go testować i obsługiwać. Wszelkie postulaty dotyczące jakości, a także uwagi, np. na temat wersji BIOS-a, można od razu zgłaszać producentowi, który stara się je szybko uwzględniać. Poza tym są dostępne najnowsze produkty, często wcześniej niż w Europie Zachodniej. Tak było z drukarkami, a wspomniany model Acer 1100 trafił do Warszawy w dwa miesiące po premierze na targach hanowerskich. Do września 1987 r. sprzedano kilkanaście sztuk.

Przy sprzedaży komputerów na tak dużą skalę trzeba było zorganizować odpowiednio sprawny serwis gwarancyjny (12 miesięcy, zgodnie z ogólnymi warunkami obowiązującymi polskich dostawców uspołecznionych) i pogwarancyjny. Udzielono więc autoryzacji kilku firmom działającym w różnych miastach, powierzając im opiekę nad komputerami w poszczególnych województwach.

W warunkach polskich szczególnie śliskim problemem jest wyposażenie komputerów w oprogramowanie i jego dokumentację. Wiele firm przerabia oryginalne produkty zachodnie, a następnie sprzedaje je jako swoje. W Zakładzie Informatyki postanowiono od początku, że należy wraz z komputerem za darmo rozprowadzać to oprogramowanie, które otrzymano od dalekowschodniego dostawcy. Na rozprowadzenie systemu operacyjnego MS DOS firma Multitech ma zresztą oficjalną licencję (z innymi programami jest gorzej...). Niektóre programy dostosowano specjalnie do współpracy z polskim alfabetem (np. Wordstar 2000). W sumie, zbiór dostępnych programów obejmuje kilkaset dyskietek, w tym cztery systemy operacyjne: MS-DOS/PC-DOS, CP/M-86, Xenix i PC-MOS. Ten ostatni stanowi wieloprogramowy odpowiednik standardu MS-DOS, opracowany dla mikrokomputerów klasy Super-AT. Są też systemy wspomagające (GEM), popularne programy zarządzania bazami danych (m.in. dBase III PLUS), procesory tekstów (Wordstar 2000, Word), arkusze są elektroniczne i pakiety zintegrowane (Lotus 1—2—3, Symphony, Framework), wreszcie liczne kompilatory i programy narzędziowe (C, Pascal, Fotran, Norton, CodeSmith). Dokumentacja programów jest wypożyczana tak, aby użytkownik mógł ją sobie skopiować.

Z każdym zakupionym komputerem jest związane prawo do przeszkolenia dwóch osób. Kursy odbywają się co tydzień w ośrodku szkoleniowym w Warszawie, trwają 5 dni po 6 godzin, a są prowadzone przez pracowników warszawskich uczelni. W sali szkoleniowej stoi 16 komputerów — dla wykładowcy i dla 30 uczestników, po dwóch na komputer. Szkolenie jest podporządkowane zasadzie nauki przez praktykowanie i obejmuje system PC-DOS, programy Wordstar 2000, dBase III, Lotus 1—2—3 oraz korzystanie z kompilatorów Turbo Pascala i MS-Fortranu. Uczestnicy kursów mają także do dyspozycji skrypt, zapisany na dysku. Jest też wydawana seria małych broszurek, zawierających spis poleceń omawianych programów.

Zakład Informatyki w czasie największych spiętrzeń montuje i wydaje klientom ok. 10 komputerów dziennie. Oczywiście w pracy korzysta się z komputerów — przy przetwarzaniu zamówień, kompletowaniu konfiguracji, fakturowaniu, przygotowywaniu umów, pism, prowadzeniu magazynów itp.

Czy metoda przyjęta przez DHN może być łatwo porównana, na przykład z metodą spółki „Mikrokomputery"? Zapewne nie, zwłaszcza jeśli uda się zrealizować obietnicę corocznego zarzucania rynku dwoma tysiącami egzemplarzy Mazovii. Trzeba by przeprowadzić dokładną analizę kosztów, uwzględniając pełny wsad dewizowy w Mazovii i złotówkowe koszty produkcji. Przy rozsądnych warunkach, uczciwej grze ekonomicznej i powstającym rynku konsumenta przyszłość pokaże, które z rozwiązań mających zapełnić polski rynek mikrokomputerowy są do zaakceptowania.

JD

BALTCOM '87

W dniach 17—20 listopada 1987 roku w Gdańsku, odbyła się wystawa komputerowa BALTCOM '87. Część ekspozycji znajdowała się w hali „Olivia", a reszta w Domu Technika NOT w centrum miasta. W Domu Technika odbywały się także wykłady i seminaria organizowane przez wystawców.

Pierwsze wrażenie z wystawy było takie, jak zwykle ostatnimi laty: mikro, mikro, mikro. Prezentowano komputery wzorowane na IBM PC z różnych części świata, głównie z tej części Zachodu, która leży na wschodzie, a także sprzęt krajowy (Mazovia). W tej grupie najciekawsza była oferta firmy **InterAMS**, komputer zgodny z nowym IBM-owskim PS2: procesor 386 z koprocesorem 387, zegar 20 MHz, dyski elastyczne typu Bernoulli 10 MB, dyski stałe 80 MB, dyskietki 3,5-calowe. Typowy zestaw kosztował 20 mln zł. Wkrótce będą dostępne stacje odczytu dysków optycznych o pojemności od 600 do 1400 MB.

W grupie sprzętu mikrokomputerowego wysokiej klasy z dołączonymi specjalizowanymi urządzeniami najciekawiej wypadła firma **Rank Xerox**, która wielu osobom kojarzy się wyłącznie z kserografami. Prezentowano podręczną drukarnię, czyli tzw. desktop publishing. Komputer klasy IBM PC/AT sterował dużym ekranem o wysokiej rozdzielczości (900 * 1280 punktów), na którym można było obejrzeć kolumny tworzonego artykułu lub książki, łącznie z rysunkami. Oprogramowanie Ventura pozwala korzystać z tekstów i grafiki, przygotowanych za pomocą wielu edytorów i programów graficznych. Można także wprowadzać obrazy z zewnątrz, korzystając ze skanera. Każdy dokument jest tworzony według z góry zadanego stylu, określającego rodzaj czcionki, postać tytułów, podział na łamy, sposób wstawiania grafiki, justowanie itp. Gotowy dokument można wydrukować na drukarce laserowej o doskonałej rozdzielczości. Kompletny system kosztuje ok. 20—25 tys. dolarów.

Firma Rank Xerox prezentowała także inne urządzenia, np. ploter elektrostatyczny, kreślący skomplikowany rysunek w formacie A3 — w ciągu 2 minut — z rozdzielczością 400 pikseli na cal, w czterech kolorach, z możliwością zapełniania całych powierzchni. Szkoda, że „Informatyka" nie jest drukowana w kolorze!

Rank Xerox prezentował także pierwszą legalną polską wersję systemu operacyjnego MS-DOS, na licencji firmy Microsoft. Będzie ona rozprowadzana wraz z komputerami Xerox od stycznia 1988 roku. Ciekawe, że producent systemu nie wyraził zgody na spolszczenie nazw poleceń. Czy Niemcy też muszą pisać „dir" aby obejrzeć katalog dyskietki?

Wśród innych urządzeń zewnętrznych moje zainteresowanie wzbudziła nakładka ciekłokrystaliczna na rzutnik pisma, pozwalająca pokazać na wielkim ekranie obraz z monitora komputerowego (DataCo).

Od pewnego czasu modne stały się systemy wielodostępne i sieci. **Computex**, dobrze już zadomowiony na rynku, prezentował system dostępu do baz danych CX-DMOS (Grand Prix w konkursie Baltcomu). Stanowi on narzędzie do konfigurowania stanowisk wprowadzania i przetwarzania danych na komputer XT z ośmioma terminalami lub AT z szesnastoma. Łącza szeregowe do terminali są obsługiwane przez dodatkowy procesor 8088. System operacyjny zapewnia niepodzielność transakcji na danych, nawet zapisanych na kilku fizycznych plikach, zabezpiecza przed blokadą itd.

Jeśli chodzi o sieci, to standardem staje się Ethernet (np. proponowała go firma **Paco**). Pokazywano kilka modemów telefonicznych i teleksowych (**Softlan, Conpol**) — niestety na razie bez homologacji, a więc do użycia tylko w sieciach wewnętrznych. Zresztą jakość i liczba komutowanych łączy,

zwłaszcza międzymiastowych, nie pozwala liczyć na rozwój dużych systemów sieciowych.

Użytkownicy sprzętu powoli przyzwyczajają się do tego, że sam mikrokomputer to nie wszystko. Po raz pierwszy chyba na takiej wystawie pokazano urządzenia zasilające, zabezpieczające przed utratą informacji wskutek awarii sieci energetycznej. Firma **VIP** proponuje urządzenie dostarczające 500 lub 1000 W mocy w czasie od kilkudziesięciu minut do kilku godzin, zależnie od użytych akumulatorów (mogą być takie od ciężarówki). Firma **Comex** oferuje małe zasilacze z wewnętrznymi akumulatorami dającymi 300 W mocy przez 20 minut.

Komputer wymaga obsługi, a obsługa wymaga szkolenia. Firma **Video Technika** zaprezentowała film wideo uczący podstaw korzystania z systemu MS-DOS, programu Sidekick i innych. Jakość filmu była dobra, bardzo dobrze wyglądały zwłaszcza obrazy ekranu komputera. Można zamówić filmy na temat innych programów.

W dziedzinie oprogramowania oczywiście ogromny wybór prezentował „koncern" **Kajkowskich**. Mieliśmy okazję obejrzeć nowy ośrodek szkoleniowo-konsultacyjny firmy **Procom**, zlokalizowany na terenie ujeżdżalni. Zatrudniono tam programistów, którzy także bywają doradcami, nauczycielami i przewodnikami po oprogramowaniu sprzedawanym przez „koncern".

Sprzęt powyżej klasy mikro tradycyjnie prezentowała firma **Hewlett-Packard i jej przedstawicielstwo Zotpan.** Systemy minikomputerowe serii 3000 i 9000 służą do wspomagania projektowania i zarządzania przedsiębiorstwami.

Celowo na koniec zostawiłem dwa przykłady specyficzne i odbijające od mikrokomputerowego tła. Państwowy gigant na rynku oprogramowania, **ZETO-ZOWAR**, przedstawił połączenie komputera XT ze stacją pamięci taśmowej PT-305, pozwalające zapisywać na taśmie i czytać z niej informacje w różnych formatach, stosowanych przez różne duże komputery. Natomiast standardowej wielkości firma **Gambit** proponuje rozbudowę i unowocześnienie istniejących minikomputerów 9150 (Seecheck), przez wymianę terminali, jednostki sterującej terminalami i dodanie dysków typu Winchester. Taki zestaw można połączyć z komputerem IBM (dużym) i wówczas terminale emulują stacje 3270. Pokrewna firma **Annex** proponuje instalowanie dysków Winchester w Odrach i SM-ach.

Tego typu oferty wzbudziły kontrowersje wśród widzów, a także wśród jurorów oceniających eksponaty (oferta Gambitu otrzymała trzecią nagrodę). Wiele osób uważało, że jest to cofanie się w rozwoju, niemalże wywoływanie upiorów przeszłości w epoce jedynie nowoczesnego sprzętu mikrokomputerowego. Inni wyświadczali obu firmom niedźwiedzią moim zdaniem przysługę, argumentując że wprawdzie istotnie są to archaizmy, ale w naszych lokalnych warunkach nikt nie wyrzuci niezamortyzowanych komputerów, a więc trzeba ... itd.

Osobiście uważam, że problem polega nie na tym, czy komputer zamortyzował się czy nie, ale na tym, czy byłoby czym go zastąpić i czy byłoby to opłacalne. W amerykańskim centrum ochrony przestrzeni powietrznej i kosmicznej do lat osiemdziesiątych działał lampowy komputer z lat pięćdziesiątych, podobne przykłady można mnożyć. Oczywiście ktoś, kogo na to stać, mógłby zastąpić Seechecki nowym minikomputerem, lecz na pewno nie mikrokomputerem osobistym klasy PC, choćby i AT 386. Niestety, minikomputerów jest na rynku bardzo mało.

Przedstawiciel firmy Procom, handlującej oprogramowaniem dla przedsiębiorstw, powiedział jasno: mikrokomputerami nie da się kompleksowo skomputeryzować średniego przedsiębiorstwa. Trzeba więc wyręczać duży przemysł dużych komputerów (śladowe ilości zawodnych Riadów i jeszcze bardziej zawodnych SM-ów praktycznie nie liczą się, a doskonałe systemy Hewlett-Packarda pozostają nieosiągalne) i produkować w miarę nowoczesny sprzęt komputerowy i minikomputerowy posiłkując się istniejącymi procesorami, zasilaczami, magistralami, łączami itp. Zresztą trudno uznać stację taśmy magnetycznej za sprzęt archaiczny — katalogi poważnych firm zawierają także takie stacje dla mikrokomputerów (prawda, że na ogół z zapisem 6000 bpi), kosztujące zazwyczaj więcej niż cała reszta komputera. Istnieje bowiem wiele urządzeń, np. w sejsmologii i w zastosowaniach wojskowych, w których stosuje się wyłącznie taśmy do zapisywania zgromadzonych wyników. Taśma jest jedynym rodzajem pamięci w amerykańskim promie kosmicznym, a nikt nie uzna tego produktu za przesadnie przestarzały.

Nie przesadzajmy więc i nie uważajmy, że cały świat przerzucił się na PC i że wstyd już mówić o innym sprzęcie. Zresztą, rozbudowa dotyczy nie tylko sprzętu sprzed kilku lat. Oto wkrótce mają pojawić się na polskim rynku superminikomputery klasy VAX, zapewne wyposażone standardowo w bułgarskie stacje dysków, kiepskie terminale itd. Doświadczenie uczy, że aby w pełni wykorzystać możliwości dobrej elektroniki trzeba będzie jak najszybciej (czyli zaraz po upływie okresu gwarancji) wymieniać terminale na odpowiedniki np. VT 200, dodawać dyski Winchester albo Bernoulli, a być może także rozszerzać pamięć itd. Uważam, że konkretna oferta w tej dziedzinie jest godna nawet pierwszej nagrody na każdej wystawie komputerowej!

Na koniec, akcent wisielczego optymizmu. W czasie wystawy odbyło się spotkanie z sekretarzem stanu w Ministerstwie Finansów. Na pytania o ewentualne ułatwienia dla producentów zaawansowanej technologii padła odpowiedź, że rząd i ministerstwo właśnie uczy się i prosi o sugestie. Przyłączamy się do jednodniówki Baltcomix i życzymy wszystkiego najlepszego!

JAROSŁAW DEMINET

Ze świata — Nowe superkomputery

Komputery o wielkich mocach obliczeniowych znajdują zastosowanie w badaniach naukowych oraz w takich przemysłach, jak: naftowy, samochodowy, lotniczy i kosmiczny. Według analizy wykonanej w marcu 1986 roku, w różnych ośrodkach na świecie pracowały 194 superkomputery. Dominującą rolę odgrywała amerykańska firma Cray (64% instalacji). Na kolejnych miejscach uplasowały się amerykańska firma CDC/ETA (18%) oraz japońskie firmy Fujitsu (11%), Hitachi (5%) i NEC (2%).

Pod koniec 1986 roku na rynku pojawiły się dwa nowe superkomputery: SX-2 firmy NEC (która weszła na rynek amerykański tworząc spółkę z firmą Honeywell) oraz ETA10 firmy ETA (w której CDC ma udziały w wysokości 89%). Obydwie maszyny przewyższają mocą obliczeniową aktualnie produkowane superkomputery. Testy wykazały, że SX-2 wykonuje 1,3 miliarda operacji zmiennoprzecinkowych na sekundę (maksymalna szybkość osiągnięta dla standardowego zestawu testów). Superkomputer ETA10, który zgodnie z założeniami miał wykonywać 10 miliardów operacji zmiennoprzecinkowych na sekundę, osiągnął maksymalną szybkość około 7 miliardów operacji na sekundę.

Pierwszy egzemplarz SX-2 został zainstalowany w Centrum Badawczym Okręgu Houston, założonym przez uniwersytety teksaskie. Pierwszymi kontrahentami firmy ETA są Uniwersytet Stanowy na Florydzie, Uniwersytet w Minnesocie oraz Konsorcjum Obliczeń Naukowych — organizacja ustanowiona przez 13 uniwersytetów i instytucji badawczych w Princeton. ETA planuje instalację 10—14 superkomputerów w 1987 roku. Firma Cray ogłosiła już, że podejmuje wyzwanie rzucone przez firmy NEC i ETA i wkrótce wprowadzi na rynek nowy komputer Cray YMP, którego parametry są jednak jak dotąd nie znane.

Superkomputery firmy Cray kosztują w zależności od mocy obliczeniowej od 4 do 21 milionów dolarów. Ceny komputerów firmy NEC mieszczą się w przedziale od 15 do 25 milionów dolarów. Superkomputer ETA10 kosztuje od 8 do 22 milionów dolarów, w zależności od liczby procesorów (2, 4, 6 lub 8).

M. MACHURA

na podstawie „Electronic Business", wrzesień 1986

Nowe książki

C. W. Holsapple, A. B. Whinston (Eds.): Decision Support Systems — Theory and Applications, Springer-Verlag, 1987.

Jest to zbiór referatów wygłoszonych na konferencji roboczej NATO poświęconej teorii i zastosowaniom systemów wspomagania procesów decyzyjnych, która odbyła się w 1985 roku we Włoszech. Artykuły zostały przygotowane przez badaczy z przemysłu, agencji rządowych i instytucji akademickich. Wszystkie przedstawione w książce systemy wspomagania procesów decyzyjnych wykorzystują techniki sztucznej inteligencji.

A. Kusiak (Ed.): Artificial Intelligence — Implications for CIM, Springer-Verlag, 1987.

Jest to zbiór specjalnie opracowanych artykułów, które stanowią wprowadzenie do tematyki sztucznej inteligencji i jej roli w integracji procesów produkcyjnych z wykorzystaniem komputerów (CIM, ang. Computer Integrated Manufacturing). Omówione techniki obejmują systemy ekspertowe, odczuwanie przez dotyk (ang. tactile sensing), systemy wizyjne oraz rozpoznawanie mowy. Przedyskutowano zastosowanie tych technik w systemach obrotu materiałami, automatycznego magazynowania i wyszukiwania. Znaczna część książki jest poświęcona systemom planowania produkcji.

J. M. Rosenberg: Dictionary of Artificial Intelligence and Robotics, John Wiley, 1986.

Słownik zawiera ponad 4000 haseł i definicji z dziedziny sztucznej inteligencji i robotyki. Jest przeznaczony zarówno dla początkujących w tej dziedzinie, jak i dla praktyków. Podaje alternatywne znaczenia terminów, skróty, akronimy oraz wyrażenia obce. Obejmuje zarówno hasła ogólne, jak i szczegółowe, kładzie nacisk na zależności między sztuczną inteligencją, robotyką i sterowaniem komputerowym oraz grupuje hasła zawierające wspólne pojęcia w porządku alfabetycznym.

CENTRALNA BIBLIOTEKA TECHNICZNA NOT

Wykaz nowych wydawnictw zwartych z dziedziny informatyki

Głuszkowski T.: Zastosowanie grafiki mikrokomputerowej; materiały szkoleniowo-doradcze PC Standard '88. Warszawa 1987, RS NOT Ośrodek Doskonalenia Kadr Technicznych (Koło Użytkowników Komputerów Profesjonalnych) *(46926 A, 46927)*

Kierzkowski Z.: Ujmowanie informacji katalogowej we wspomaganiu komputerowym projektowania; VII Szkoła Metodologii Konstruowania Maszyn — materiały seminaryjne *(46934 A, 46935, 46936)*

Krawiec S.: Komputerowe wspomaganie doboru cech konstrukcyjnych sprężyn śrubowych; VII Szkoła Metodologii Konstruowania Maszyn — materiały seminaryjne. Rydzyna 1987, Centrum Postępu Technicznego SIMP *(46971 A, 46972)*

Kręcicjewski M.: Układy cyfrowe. Warszawa 1988, Wydawnictwo Czasopism i Książek Technicznych NOT-SIGMA *(46934 A, 46935, 46936)*

Oprogramowanie narzędziowe mikrokomputerów profesjonalnych; materiały szkoleniowo-doradcze PC Standard '88. Warszawa 1987, RS NOT Ośrodek Doskonalenia Kadr Technicznych (Koło Użytkowników Mikrokomputerów Profesjonalnych) *(46922 A, 46923)*

Rusiński E.: Zastosowanie mikrokomputera do kształtowania ustrojów nośnych maszyn MES; VII Szkoła Metodologii Konstruowania Maszyn — materiały seminaryjne cz. 6. Rydzyna 1987, Centrum Postępu Technicznego SIMP *(46969 A, 46970)*

Szuniewicz R.: Wspomaganie informatyczne działań przedsiębiorstwa. Warszawa 1987 OPT NOT, RS NOT Ośrodek Doskonalenia Kadr Technicznych *(46918 A, 46919)*

W nawiasach podano numery katalogowe.

WARUNKI PRENUMERATY NA 1988 ROK (dotyczy numerów 7-12, 1988 r.)

Prenumeratorzy zbiorowi — jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłaty wyłącznie na blankiecie „wpłata—zamówienie" (jest to „polecenie przelewu" rozszerzone dla potrzeb Wydawnictwa o część dotyczącą zamówienia).

Blankiety te będą dostarczane dotychczasowym prenumeratorom przez Zakład Kolportażu. Nowi prenumeratorzy otrzymają je po zgłoszeniu zapotrzebowania (pisemne lub telefoniczne) w Zakładzie Kolportażu.

Prenumeratorzy indywidualni — osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty. Wpłacać należy na konto: NBP III Oddział Warszawa 1036-7490-139-11.

Prenumerata ulgowa — przysługuje wyłącznie osobom fizycznym — członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły. Sposób zamawiania prenumeraty ulgowej jest taki sam jak prenumeraty indywidualnej. W prenumeracie ulgowej można zamówić tylko po 1 egzemplarzu każdego czasopisma.

Uwaga! Miesięcznik „Aura" może być zamawiany w prenumeracie ulgowej również przez uczniów szkół ogólnokształcących.

Prenumerata ze zleceniem wysyłki za granicę — zamawia się tak jak prenumeratę indywidualną. Dodatkowo należy podać na blankiecie wpłaty nazwisko i dokładny adres odbiorcy.

Cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa.

Wpłaty na prenumeratę przyjmowane są w terminach:

- do 10 listopada na każdy kwartał, I i II półrocze oraz cały rok następny,
- do 28 lutego na II, III i IV kwartał oraz II półrocze,
- do 31 maja na III i IV kwartał oraz II półrocze,
- do 31 sierpnia na IV kwartał.

Zmiany w prenumeracie można zgłaszać pisemnie tylko w wyżej wymienionych terminach.

Informacji o prenumeracie udziela — Zakład Kolportażu Wydawnictwa NOT-SIGMA (ul. Bartycka 20, 00-716 Warszawa), skr. poczt. 1004, 00-950 Warszawa, tel. 40-00-21 wew. 248, 249, 293, 297, 299 lub 40-30-86 i 40-35-89.

Egzemplarze archiwalne czasopism — można nabyć za gotówkę w Klubie Prasy Technicznej, Warszawa ul. Mazowiecka 12 (tel. 27-43-65), lub zamówić pisemnie. Zamówienia na egzemplarze archiwalne czasopism przyjmuje: Zakład Kolportażu, Dział Handlowy, 00-950 Warszawa, skr. poczt. 1004 (tel. 40-37-31), na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

CENA		Prenumerata normalna		Prenumerata ulgowa bez zmiany	
Normalna	ulgowa bez zmiany	kwartalna	półroczna	kwartalna	półroczna
250,—	50,—	750,—	1.500,—	150,—	300,—

Basic po polsku (1)

Dwa lata temu p. Maciej Szewczuk opracował polską wersję Basica dla komputerów ZX Spectrum. Interpreter o nazwie TACT POLBASIC jest zapisany w pamięci ROM rozpowszechnianej w postaci specjalnej przystawki dołączanej do komputera. Ważną cechą tej przystawki jest możliwość programowego przełączania pracy, na przemian, na angielską lub polską wersję Basica. Sądzę, że w rubryce terminologicznej warto co najmniej omówić polskie odpowiedniki angielskich słów kluczowych Basica i ustosunkować się do takiego wyboru, odpowiadając na pytanie, czy przypadkiem po raz kolejny nie usiłowano „spolszczyć Esperanta". Zaznaczam, że rozważania te dotyczą tylko kwestii językowych i terminologicznych, a nie samego interpretera lub jego realizacji.

Oddajmy najpierw głos Autorowi oprogramowania. Wyjaśniając w instrukcji sposób tłumaczenia słów kluczowych, podaje on następującą argumentację.

Przetłumaczyć rozkazy i funkcje można na wiele sposobów. Nie wszystkie polskie hasła pokrywają się swoim znaczeniem z hasłami angielskimi. Dobierając je kierowałem się nadrzędną dyrektywą, aby tłumaczenia układały się w programie w poprawne polskie zdania. Z tego samego powodu nie stosowałem skrótów haseł. W kilku wypadkach zdecydowałem się na użycie haseł dwuwyrazowych, jak np. TO WTEDY czy RYSUJ KOŁO, tak aby zasadzie poprawnej składni stało się zadość. ... Walor edukacyjny takiego rozwiązania jest bezsporny. Nie tłumaczyłem wszystkiego, gdyż uważam, że przesada jest szkodliwa. Dlatego pozostały w swojej poprzedniej postaci skróty matematyczne: EXP, FN, DEF, LN — mają one swoją tradycję w polskiej matematyce.

Po przeczytaniu takiego wyjaśnienia można powiedzieć tylko jedno: Brawo! Założenie zgodności programów w Basicu ze składnią języka polskiego jest koniecznym, choć nie jedynym argumentem, który może mnie przekonać o celowości programowania przy użyciu polskich słów kluczowych. Nie jestem tylko pewien, czy każdy program poprawny składniowo w Basicu będzie poprawny składniowo w języku polskim. Jest bardzo pożądane, aby tak było, choćby jedynie w dobrym przybliżeniu, ale należałoby to dokładniej zbadać.

Gdyby jednak nawet udało się tego dowieść, celowość kodowania programów w języku polskim będzie nadal wątpliwa. Co bowiem począć, jeśli nie można korzystać z książek uczących programowania w Basicu w wersji angielskiej lub — z programów napisanych w angielskiej wersji języka?

Z pewnym zdziwieniem stwierdziłem, że p. Szewczuk znalazł na to bardzo prostą radę! Oddajmy mu głos ponownie. W instrukcji użytkowania przystawki POLBASIC stwierdzono, że: *komputer posiada jeszcze jedną bardzo użyteczną właściwość — tłumaczy program napisany po polsku na język angielski lub odwrotnie. Pozwala to zanalizować w polskiej wersji językowej (co chyba łatwiejsze) dowolny, uprzednio napisany po angielsku na normalnym SPECTRUM, program w Basicu. Wystarczy go tylko wczytać i wybrać, ... polską wersję ROM-u. Dla tych, którzy chcą także nauczyć się angielskiej wersji Basica, zaleca się napisanie programu po polsku a następnie, po przełączeniu na wersję angielską, zanalizowanie programu przez porównywanie go z wersją polską.*

Sądzę, że te dwa założenia:

- zapewnienie zgodności składni Basica ze składnią języka polskiego;
- natychmiastowa przetłumaczalność wszystkich programów w Basicu angielskojęzycznym na Basic polskojęzyczny i odwrotnie;

są nieodzownymi warunkami celowości spolszczenia Basica i stanowią wystarczającą argumentację na rzecz podjęcia takich działań. Nawiasem mówiąc, warunki te mogą dotyczyć każdego języka (nietrudno też zauważyć, że żadnego z nich nie spełnia polskie Logo PTI).

Jeżeli można więc uznać generalnie zasadność takiego podejścia do spolszczenia Basica, to tym bardziej warto przyjrzeć się bliżej polskim odpowiednikom angielskich słów kluczowych. Zasadniczo wszystkie słowa kluczowe w Basicu można podzielić na dwa rodzaje — oznaczające funkcje i instrukcje. Poniżej omówiono tylko nazwy funkcji, odkładając omówienie polskich nazw instrukcji do numeru 8/1988.

Jak przyjmuje Autor oprogramowania, nie wszystko należy bezwzględnie spolszczać. Nie należy tłumaczyć nazw funkcji matematycznych, które utrwaliły się w języku polskim. Jest to zasada bardzo ważna, choć zastosowana nie w pełni konsekwentnie. Utrzymano, na przykład, takie nazwy: EXP, LN, SIN, COS, FN, INT, dostosowując inne do skrótów używanych w języku polskim, np. TG, CTG, ASIN, ACOS (powinno być, zresztą, ARCSIN i ARCCOS), lecz niestety nie utrzymano oryginalnych nazw wielu innych funkcji, np. logicznych.

Moim zdaniem, tak jak kiedyś dobierano łacińskie nazwy funkcji matematycznych i nieskuteczne były próby ich spolszczenia (słynna jest nieudana próba spolszczenia nazw funkcji sinus i cosinus), obecnie w informatyce powszechnie przejmuje się nazwy funkcji z języka angielskiego i próby spolszczenia tych nazw będą także kończyły się niepowodzeniem. Dlatego uważam, że bardziej sensowne byłoby pozostawienie następujących nazw oryginalnych: NOT, AND, OR, a także ABS, SGN i RND. Szczególnie przetłumaczenie SGN (ang. sign, łac. signum) na ZNAK jest błędem, ponieważ w języku polskim (i w POLBASICU) wyraz „znak" znaczy „znak liczby" i „znak alfanumeryczny", co może prowadzić do nieporozumień. Funkcja RND (lub RAND, RANDOM) jest dostatecznie rozpowszechniona w językach programowania, aby można jej nazwę uznać za ustaloną, podobnie jak ABS. Nie jestem też przekonany, czy zmiana rozpowszechnionej już nazwy SQR na znak pierwiastkowania jest sensowna — raczej należałoby przy niej pozostać, ponieważ znakiem pierwiastkowania w matematyce obejmuje się graficznie całe wyrażenie, co trudno wykonać w tekście programu.

Do pozostałych polskich nazw funkcji w zasadzie nie można mieć zastrzeżeń. Nazwy funkcji udostępniających napisy utworzono poprawnie (ZNAK$ — CHR$, KLAWISZ$ — INKEY$, EKRAN$ — SCREEN$, WARTOŚĆ$ — VAL$) poza jednym wyjątkiem — wyraz string oznacza w Basicu napis, a nie łańcuch. Tak więc polskim odpowiednikiem funkcji STRING$ powinna być funkcja NAPIS$. Inne funkcje nazwano zgodnie ze znaczeniem pełnych wyrazów angielskich (ATRYBUT — ATTR, KOD — CODE, DŁUGOŚĆ — LEN, WARTOŚĆ — VAL) lub treścią operacji (DZIESIĘTNIE — BIN, ADRES — USR).

Dwie funkcje IN i PEEK, dość specyficzne, bo mające swoje odpowiedniki w instrukcjach (OUT i POKE), mają nazwy dobrane symetrycznie do nazw tych instrukcji.

JANUSZ ZALEWSKI

Griswold R. E., Griswold M. T.: Języki wysokiego poziomu do przetwarzania napisów — COMIT, SNOBOL4 i Icon (1)

INFORMATYKA 1988, nr 4, s. 1

Pierwsza część charakterystyki metod i narzędzi do przetwarzania napisów, zawierająca omówienie języka COMIT.

Griswold R. E., Griswold M. T.: High level languages for string processing — COMIT, SNOBOL4 and Icon (1)

INFORMATYKA 1988, No. 4, p. 1

First part of characteristics of methods and tools for string processing, which contains discussion of the COMIT language.

Griswold R. E., Griswold M. T.: Höhere Sprachen zur Zeichenfolgenverarbeitung — COMIT, SNOBOL4 und Icon (1)

INFORMATYKA 1988, Nr. 4, S. 1

Erster Teil einer Charakteristik von Methoden und Hilfsmittel zur Zeichenfolgenverarbeitung, der eine Besprechung der COMIT-Sprache umfasst.

Stawicki J.: Narzędzia tworzenia systemów ekspertowych na mikrokomputerach

INFORMATYKA 1988, nr 4, s. 3

Charakterystyka pakietów programowych wspomagających tworzenie systemów ekspertowych na mikrokomputerach klasy IBM PC XT/AT.

Stawicki J.: Tools for expert system building on microcomputers

INFORMATYKA 1988, No. 4, p. 3

Characteristics of program packages for assisted expert system building on microcomputers of the IBM PC XT/AT class.

Stawicki J.: Hilfsmittel zum Expertensystemebau auf Mikrorechnern

INFORMATYKA 1988, Nr. 4, S. 3

Eine Charakteristik von Programmpaketen zur Unterstützung des Expertensystemebaues auf Mikrorechnern der IBM PC XT/AT-Klasse.

Królikowski Z.: Mikrokomputerowe systemy zarządzania bazami danych

INFORMATYKA 1988, nr 4, s. 6

Przegląd najczęściej stosowanych mikrokomputerowych systemów zarządzania bazami danych oraz omówienie perspektyw dalszego ich rozwoju.

Królikowski Z.: Microcomputer data base management systems

INFORMATYKA 1988, No. 4, p. 6

Survey of most applied microcomputer data base management systems and discussion of their future development.

Królikowski Z.: Mikrorechner-Datenbankverwaltungssysteme

INFORMATYKA 1988, Nr. 4, S. 6

Übersicht der am häufigsten angewendeten Mikrorechner-Datenbankverwaltungssysteme und eine Besprechung von Aussichten ihrer weiteren Entwicklung.

Zieliński K., Indulska J., Magura-Witkowski P., Walasek T.: Architektura oprogramowania lokalnej sieci komputerowej UMMLAN-2

INFORMATYKA 1988, nr 4, s. 9

Charakterystyka oprogramowania lokalnej sieci komputerowej UMMLAN-2, zaprojektowanej i rzealizowanej w Instytucie Informatyki Akademii Górniczo-Hutniczej w Krakowie.

Zieliński K., Indulska J., Magura-Witkowski P., Walasek T.: Software architecture of the UMMLAN-2 local area network

INFORMATYKA 1988, No. 4, p. 9

Software architecture characteristics of the UMMLAN-2 local area network, which was designed and realized in the Informatics Institute of the Mining and Metallurgy University in Cracov.

Zieliński K., Indulska J., Magura-Witkowski P., Walasek T.: Software-Architektur von UMMLAN-2-Lokalnetz

INFORMATYKA 1988, Nr. 4, S. 9

Eine Charakteristik von Software des UMMLAN-2-Lokalnetzes, die von Institut für Informatik der Bergbau- und Hüttenakademie in Krakau projektiert und realisiert wurde.

Sysło M. M.: Algorytmy kombinatoryczne i ich efektywność (2). Problem plecakowy

INFORMATYKA 1988, nr 4, s. 13

Druga część charakterystyki wybranych metod rozwiązywania problemów kombinatorycznych, zawierająca omówienie sposobu rozwiązania problemu plecakowego.

Sysło M. M.: Combinatorial algorithms and their effectiveness (2). Knapsack problem

INFORMATYKA 1988, No. 4, p. 13

Second part of selected methods for combinatorial problem solving characteristics, which contains discussion of the method for knapsack problem solution.

Sysło M. M.: Kombinatorische Algorithmen und ihre Effektivität (2). Rucksackproblem

INFORMATYKA 1988, Nr. 4, S. 13

Zweiter Teil einer Charakteristik von ausgewählten Lösungsmethoden für kombinatorische Algorithmen, der eine Besprechung von Lösungsmethode des Rücksackproblems umfasst.

Krępski A.: Język Smalltalk-80 (2)

INFORMATYKA 1988, nr 4, s. 16

Druga część charakterystyki języka programowania Smalltalk-80.

Krępski A.: Smalltalk-80 language (2)

INFORMATYKA 1988, No. 4, p. 16

Second part of the Smalltalk-80 programming language characteristics.

Krępski A.: Smalltalk-80-Sprache (2)

INFORMATYKA 1988, Nr. 4, S. 16

Zweiter Teil einer Charakteristik von der Smalltalk-80-Sprache.

Bielecki J.: Język C. Coraz bliżej normy (2)

INFORMATYKA 1988, nr 4, s. 19

Kontynuacja omówienia projektu normy języka C, opracowanego przez amerykański instytut normalizacji ANSI.

Bielecki J.: C language. Nearer and nearer to standard (2)

INFORMATYKA 1988, No. 4, p. 19

Discussion continuation of the C language standard project, which is elaborated by ANSI.

Bielecki J.: C-Sprache. Immer näher zur Norme (2)

INFORMATYKA 1988, Nr. 4, S. 19

Fortsetzung einer Besprechung von dem C-Sprache-Normentwurf, der von ANSI erarbeitet wurde.

Warda A.: Pamięci dyskowe Mazovii 1016 (2)

INFORMATYKA 1988, nr 4, s. 21

Druga część charakterystyki pamięci dyskowych mikrokomputera Mazovia 1016, zawierająca omówienie napędu i sterownika pamięci z dyskiem stałym.

Warda A.: Disk storages of Mazovia 1016 (2)

INFORMATYKA 1988, No. 4, p. 21

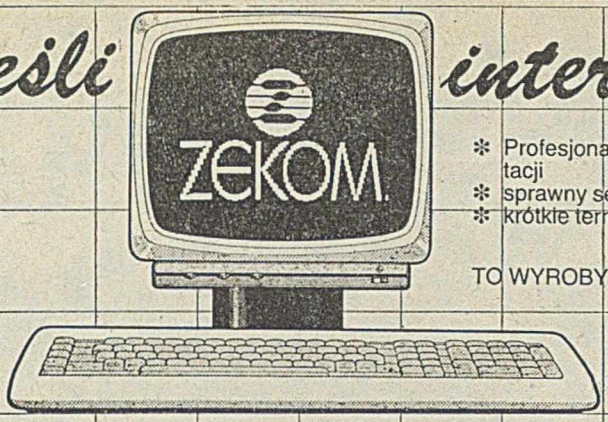
Second part of the Mazovia 1016 microcomputer disk storages characteristics, which contains discussion of the hard disk drive and controller.

Warda A.: Plattenspeicher von Mazovia 1016 (2)

INFORMATYKA 1988, Nr. 4, S. 21

Zweiter Teil einer Charakteristik von Plattenspeichern des Mazovia 1016-Mikrorechners, der eine Besprechung von Festplattenlaufwerk und -Steuerung umfasst.

Jeśli interesuje Państwa



* Profesjonalny sprzęt o wysokiej jakości, niezawodny w eksploatacji
* sprawny serwis
* krótkie terminy dostaw, lub dostawy natychmiastowe

TO WYROBY ZEKOMU SĄ DO PAŃSTWA DYSPOZYCJI

Zakład Elektroniki Komputerowej

oferuje terminale ekranowe:

MV 2580 Standard VT 52 firmy DEC / odpowiednik MERA 7953. Przeznaczony do pracy w systemach komputerowych wyposażonych w kanał transmisji V 24 lub pętli prądowej 20/60 mA – jako końcówka zdalnego dostępu.

MV 2581 Odpowiednik MERA 7911 N.
Przeznaczony do pracy w systemach komputerowych ODRA 1300 wyposażonych w jednostkę sterującą MERA 7802.

MV 2582E Odpowiednik terminala typu 7181/2 firmy ICL.
Przeznaczony do pracy w systemach komputerowych ODRA 1300, ICL 1900, ICL 2900, ICL system 4.

MR 1240 Odpowiednik MERA 7951.
Przeznaczony do wprowadzania danych do systemu MERA 9150 lub systemu REDIFON.

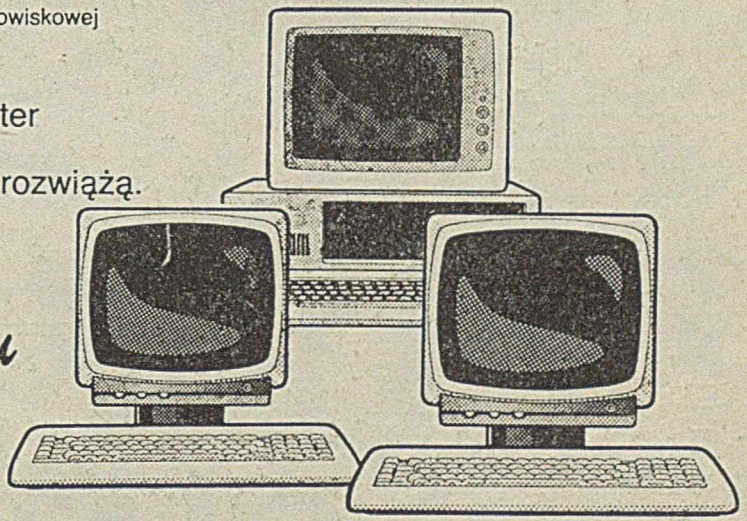
ZAKŁAD ELEKTRONIKI KOMPUTEROWEJ ul. Makowa 8, 91-480 Łódź tel. 34 30 49

multiTe™

Zestawy wieloterminalowe do pracy wielostanowiskowej
z komputerem klasy IBM PC/XT/AT

Jeśli chcecie Państwo
lepiej wykorzystać swój komputer
to zestawy multiTe
sprawnie i szybko ten problem rozwiążą.

Liczba stanowisk stosownie do potrzeb.
Możliwe nawet zestawy 8-terminalowe.



Nasze terminale gwarancją sukcesu

ZEKOM ZAKŁAD ELEKTRONIKI KOMPUTEROWEJ
Skrytka pocztowa nr 35, 90-955 Łódź 8 tel. 34-30-49

EO/307/87

6-letnie doświadczenie software'owe sprawdzone w ponad 1000 zakładach pracy

TYLKO SYSTEM CSK/32

jeśli zarządzanie Twoim przedsiębiorstwem wymaga minikomputera o mocy obliczeniowej Odry oraz kilkunastu skomputeryzowanych stanowisk pracy.

System CSK/32 to:

MIKROKOMPUTERY QUATRO CSK/32
— przystosowane do pracy z kilkunastoma terminalami, pracujące w dowolnym trybie graficznym

OPROGRAMOWANIE
— systemowe WDOS, MIKROLAN
— narzędziowe — MEGA BLOK, TABPLAN, PL-TEKST, BGRAF, TRYS
— aplikacyjne — FK K&K, EM K&K, KADRY, PŁACE

WDROŻENIA, SZKOLENIA
— zespoły ds. wdrożeń i szkoleń pracują w:
Gdyni tel. 248 018, tlx 054792 • Krakowie tel. 373 573 • Poznaniu tel. 676 271
Wrocławiu tel. 481 679 • Warszawie tel. 258 528, tlx 816711
Sopocie — w wyspecjalizowanym Salonie Sprzedaży, ul. Kombatantów 1

CSK™

computer studio kajkowscy

EO/74/88

MIĘDZYWOJEWÓDZKA SPÓŁDZIELNIA PRACY „SIÓDEMKA"

ŁODŹ, AL. KOŚCIUSZKI 93,

ZAKŁAD INFORMATYKI I SYSTEMÓW KOMPUTEROWYCH

poleca po **najniższych cenach** w kraju:

— mikrokomputery 8-, 16-, 32-bitowe najwyższej jakości renomowanych firm z całego świata
— urządzenia peryferyjne:
- drukarki — również 24-igłowe i laserowe
- streamery
- napędy dyskowe 3", 5.25"
- monitory monochromatyczne, kolorowe, EGA, HEGA, VGA
- karty rozszerzenia pamięci
- kontrolery
- dyski twarde typu Winchester 20 MB, 40 MB, 60 MB, 80 MB

— systemy wielodostępne i lokalne sieci mikrokomputerowe:
- MULTI-LINK
- D-LINK
- XENIX

— materiały eksploatacyjne:
- dyskietki 5.25" MD2-D
- dyskietki 5.25" MD2-HD
- dyskietki 3" CF2
- dyskietki 3.5" MF 2DD
- taśmy barwiące do wszystkich typów drukarek STAR i NEC

Termin realizacji zamówień **natychmiast** po złożeniu zamówienia. **Bezpłatnie:** szkolenia, kursy, zestawy oprogramowania narzędziowego i użytkowego — przy dostarczeniu kompletnych systemów.

Proponujemy również na wszelkiego rodzaju mikrokomputery programy wspomagające zarządzanie przedsiębiorstwem:
— system finansowo-księgowo-kosztowy
— system zbytu i zaopatrzenia
— system technicznego przygotowania produkcji
— system materiałowy
— system kadrowy i kadrowo-płacowy (również dla pracowników akordowych)
Wymienione systemy pracują w wersjach sieciowych i wielodostępnych.

Wszelkich informacji udzielamy codziennie (oprócz sobót) w siedzibie Zakładu w Łodzi przy Al. Kościuszki 101, tel. 36-51-00, w godzinach 8—16.

EO/1005/87