11 1993

informatyka

WYDAWNICTWO SIGMA NOT

Cena 25 000 zł

informatyka

nr 11 1993
Listopad
Rok wydania XXVIII

Miesięcznik
ISSN 0542-9951
INDEKS 36124

WYDAWCA:
Wydawnictwo Czasopism i Książek Technicznych SIGMA NOT
Spółka z o.o.
ul. Ratuszowa 11
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
01-552 Warszawa,
Pl. Inwalidów 10, p. 104, 105
tel. 39-14-34

Materiałów nie zamówionych redakcja nie zwraca

W numerze:

Strona

W najbliższych numerach:

- **Kazimierz Subieta ocenia niektóre aspekty stanu informatyki w Polsce, prezentuje listę patologii związanych z zastosowaniem matematyki w informatyce.**
- **Stanisław Kędzierski charakteryzuje modelowanie obiektów dynamicznych.**
- **Piotr Woźniak opisuje metodę SuperMemo jako narzędzie wzrostu wydajności pracy programisty – artykuł po raz pierwszy w INFORMATYCE publikowany w języku angielskim.**
- **Janusz Gwiazda dzieli się z Czytelnikami wiedzą na temat zasad pracy ośrodka obliczeniowego firmy ubezpieczeniowej w Kanadzie.**

Warunki prenumeraty na 1993 r.

Zamówienia na prenumeratę czasopism wydawanych przez Wydawnictwo SIGMA-NOT można składać w dowolnym terminie. Mogą one obejmować dowolny okres czasu, tzn. dotyczyć dowolnej liczby kolejnych zeszytów każdego czasopisma.

Zamawiający może otrzymywać zaprenumerowany przez siebie tytuł począwszy od następnego miesiąca po dokonaniu wpłaty. Zamówienia na zeszyty sprzed daty otrzymania wpłaty będą realizowane w miarę możliwości – z posiadanych zapasów magazynowych.

Warunkiem przyjęcia i realizacji zamówienia jest otrzymanie z banku potwierdzenia dokonania wpłaty przez prenumeratora. Dokument wpłaty jest równoznaczny ze złożeniem zamówienia.

Wpłat na prenumeratę można dokonywać na ogólnie dostępnych blankietach w Urzędach Pocztowych (przekazy pieniężne) lub Bankach (polecenie przelewu), przekazując środki pod adres:
Wydawnictwo SIGMA-NOT Spółka z o.o.
Zakład Kolportażu
00-950 Warszawa, skr. poczt. 1004
konto:
PBK S.A. III 0/Warszawa nr 370015-1573

Wpłaty na prenumeratę od marca br. przyjmują także wszystkie urzędy pocztowe nadawczo-odbiorcze oraz doręczyciele na terenie całego kraju

Na blankiecie wpłaty należy czytelnie podać nazwę zamawianego czasopisma, liczbę zamawianych egzemplarzy, okres prenumeraty oraz własny adres.
Na życzenie prenumeratora, zgłoszone np. telefonicznie, Zakład Kolportażu ul. Bartycka 20, 00-950 Warszawa, (telefony: 40-30-86, 40-35-89 oraz 40-00-21 wew. 249, 293, 299) wysyła specjalne blankiety zamówień wraz z aktualną listą tytułów i cennikiem czasopism.

Odbiorcy zagraniczni mogą otrzymywać czasopisma poprzez prenumeratę dewizową (wpłata dokonywana poza granicami Polski w dewizach, wg cennika dewizowego z cenami podanymi w dolarach amerykańskich) lub poprzez zamówioną w kraju prenumeratę ze zleceniem wysyłki za granicę (zamawiający podaje dokładny adres odbiorcy za granicą, dokonując równocześnie wpłaty w wysokości dwukrotnie wyższej niż cena normalnej prenumeraty krajowej).

Egzemplarze archiwalne (sprzedaż przelewowa lub za zaliczeniem pocztowym) można zamawiać pisemnie, kierując zamówienia pod adresem: Wydawnictwo SIGMA NOT, Spółka z o.o. Zakład Kolportażu, 00-716 Warszawa, ul. Bartycka 20, paw. B, tel. 40-37-31, natomiast za gotówkę można je nabyć w Klubie Prasy Technicznej w Warszawie ul. Mazowieckiej 12, tel. 26-80-17.

Istnieje możliwość zaprenumerowania 1 egz. czasopisma po cenie ulgowej przez indywidualnych członków stowarzyszeń naukowo-technicznych zrzeszonych w FSNT oraz przez uczniów zawodowych i studentów szkół wyższych. Blankiet wpłaty na prenumeratę ulgową musi być opatrzony na wszystkich odcinkach pieczęcią koła SNT lub szkoły.

W przypadku zmiany cen w okresie objętym prenumeratą Wydawnictwo zastrzega sobie prawo do wystąpienia o dopłatę różnicy cen oraz prawo do realizowania prenumeraty tylko w pełni opłaconej.
Cena jednego egzemplarza: normalna 25 000 zł, ulgowa 18 750 zł
Wartość prenumeraty w zł:
Normalna: kwartalna 75 000, półroczna 150 000, roczna 300 000
Ulgowa: kwartalna 56 250, półroczna 112 500, roczna 225 000.

Skład i druk: Drukarnia SIGMA NOT Sp. z o.o. z. 394/93

Piąta Jubileuszowa Międzynarodowa
Górska Szkoła PTI Szczyrk
21–25.06.1993 r.

WOJCIECH M. JAWORSKI
Computer Science Department
Concordia University
Montreal, Kanada

ANDRZEJ ZALIWSKI
MARIAN KURAŚ
Katedra Informatyki
Akademia Ekonomiczna
Kraków

infoMapy
inna metoda gromadzenia wiedzy

Nie ma nic bardziej praktycznego
niż dobra teoria
(*Pan Edek powołujący się na W.I.L.*
a ostatnio na A. Ensteina)

Głównym problemem jaki napotyka każdy, kto zajmuje się projektowaniem lub konserwacją wielkich systemów oprogramowania, jest ich złożoność. Systemy oprogramowania znacznie różnią się od innych systemów. Projektant musi operować na wielu różnych poziomach abstrakcji oraz zajmować się wieloma szczegółami. System oprogramowania – o czym należy pamiętać – przyjmuje złożoność swoich aplikacji, a i sam proces wytwarzania oprogramowania jest szczególnie złożony. Typowymi produktami towarzyszącymi procesowi tworzenia oprogramowania są raporty, podręczniki, diagramy, teksty źródłowe itp. Do sporządzenia każdego z tych dokumentów są używane inne techniki notacyjne. Nierzadko dla utworzenia pojedynczego dokumentu używa się kilku metod. W takiej sytuacji występuje silna zależność i pokrywanie się informacji w poszczególnych dokumentach, będących produktami kolejnych faz procesu rozwoju oprogramowania. Jest to nie do uniknięcia, zwłaszcza, że wspomniane dokumenty są rezultatem stosowania różnych podejść, metod, technik reprezentacji wizualnej, konwencji opisu, języków, narzędzi wspomagania itp. W związku z tym dokumenty związane z procesem rozwoju oprogramowania są fragmentarycznymi opisami i częściowo wzajemnie nakładającymi się na siebie obrazami systemu oprogramowania.

Do dokumentowania systemów informatycznych (SI) stosowane są różne techniki diagramowania w celu prezentacji różnych aspektów funkcjonowania systemów (np. DFD – *Data Flow Diagrams*, ERD – *Entily-Relationship Diagrams*, STD – *State-Transition Diagrams, Structure Charts, Jackson Data Structure Diagrams, Nassi-Shneiderman Diagrams*). Każdy z diagramów może opisywać ten sam system i prezentuje jego komponenty. Żaden z nich nie reprezentuje całego systemu – daje tylko pewien specyficzny obraz, przekrój systemu. W konsekwencji modyfikacja jakiegoś komponentu musi spowodować odpowiednie zmiany w innych reprezentacjach. Takie pochodne zmiany nie zawsze są od razu jasne i czytelne. Utrzymanie spójności takiej dokumentacji wymaga znaczących nakładów pracy.

Tworzenie i rozwój oprogramowania możemy potraktować jako proces pozyskiwania wiedzy. Na określonym etapie tworzymy bazę wiedzy zawierającą informacje o tworzonym systemie. Rozwój następuje w wyniku pozyskiwania dalszych informacji i ich włączania do bazy wiedzy lub jako efekt przetwarzania informacji zgromadzonej wcześniej. Przechowywana wiedza jest zróżnicowana w zależności od fazy projektu. Występuje na różnych poziomach szczegółowości, począwszy od założeń programu a kończąc na kodzie. Rzadko zdarza się by powiązania między tymi poziomami były formalne lub kompletne. Przed techniką rozwoju oprogramowania stawia się następujące wymagania:

- formalnej notacji,
- technologii, która wspomaga i jest wspomagana przez notację,
- metod reprezentacji obiektów i wiedzy w obrębie danej dziedziny,
- reguł odwzorowania modeli między różnymi konwencjami notacji.

Celem niniejszej publikacji jest przedstawienie techniki notacyjnej spełniającej powyższe wymagania, zatem mogącej stanowić podstawę rozwoju oprogramowania. Technika ta nosi nazwę: **infoMapy**. Jej możliwości pozwalają na opisywanie systemu oprogramowania na wielu poziomach abstrakcji oraz na uniknięcie trudności związanych z translacją i konwersją informacji projektowych, gdyż ta sama konwencja notacji może być użyta we wszystkich fazach procesu produkcji oprogramowania. Ponadto, notacja infoMap umożliwia modelowanie aplikacji i ich systemu oprogramowania na wszystkich poziomach rozwoju, począwszy od jego założeń i specyfikacji a kończąc na kodzie źródłowym programu.

infoMapy jako technika notacji

W celu rozwinięcia formalizmu, który może być użyty dla szerokiego zakresu aplikacji na różnych poziomach abstrakcji, potrzebujemy podstawowych struktur i technik do manipulowania nimi. Notacyjnie technologia infoMap jest oparta na podstawowych pojęciach matematyki: zbiorach i relacjach. W sensie matematycznym zbiór jest kolekcją obiektów spełniających pewne warunki lub mających tę samą charakterystykę. Standardowymi strukturami matematyki dyskretnej są zbiory, funkcje, relacje i grafy. Relacje są szeroko używane w teorii baz danych, podczas gdy grafy są używane dla tworzenia narzędzi do rozwoju oprogramowania [15].

Relacja jest zdefiniowana jako odwzorowanie produktu kartezjańskiego pewnej liczby zbiorów na zbiór {0, 1}. Innymi słowy, relacja między zbiorami istnieje lub nie. Taka definicja relacji jest niewłaściwa dla wielu zastosowań deskryptywnych. W rzeczywistości bowiem relacja może także przenosić znaczenie. Przykładowo, relacja *matka, matki* do jej *dziecka* nie tylko istnieje, ale także zawiera rodzaj zależności. Definicja relacji

powinna więc zostać uogólniona jako odwzorowanie produktu kartezjańskiego pewnej liczby zbiorów na zbiór symboli, gdzie każdy symbol definiuje jednoznacznie rodzaj relacji. infoMapa jest reprezentacją tak uogólnionej relacji.

Notacja infoMap [6–12] łączy wizualną prostotę grafu z gęstością informacyjną tablicy. Z dziedziny baz danych została zapożyczona koncepcja słownika danych. Reprezentacja wizualna została zapożyczona z dziedziny tablic decyzyjnych. Powstaje w ten sposób nowa, macierzopodobna reprezentacja o dużej pojemności informacyjnej.

Przeanalizujmy prosty przykład. infoMapa z rysunku 1 przedstawia plan wykładu na temat infoMap. infoMapa ta zawiera następujące informacje dotyczące:

- zastosowania przekroju,
- struktury wykładu,
- rysunków.

W jMapie wydzielone są dwie zasadnicze części:

- obszar definicji zbiorów
- obszar definicji relacji.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1																	
2		A	A	A	A	A	A	A	A	A	A	A	A	A		{ZASTOSOWANIE PRZEKROJU}	
3		v														Instytucja	
4			v													Wykładowca	
5				v	v	v										Struktura wykładu	
6							v	v	v	v	v	v	v	v		Użycie rysunków	
7															v	Liczebność	
8		M	M	H	H	H	o	o	o	o	o	o	o	o	16	{CZĘŚCI WYKŁADU}	
9		v														Cracow School of Economics	
10		v														Canadian -Polish Management Centre	
11			v													Wojciech M. Jaworski, Concordia University, Montreal	
12				h			o									*infoMaps: a System for Knowledge Transfer Automation*	
13				1	h											Wprowadzenie	
14					1		o									Efektywność transferu wiedzy	
15					2			o								Dokładność transferu wiedzy	
16				2		h										Dlaczego "Automatyzacja transferu wiedzy"?	
17						1										Człowiek-, Komputer-, Wiedza "wykonawcza"	
18						2										Proces pozyskiwania wiedzy i kontrola jakości	
19						3										Zastosowanie transferu wiedzy	
20						4										Użycie wiedzy - Operator, Komputer	
21				3						o						Funkcjonalność "infoFarm"	
22				4							o					Produkcja oprogramowania - bardzo prosty przykład	
23				5								o				Składniki bazy wiedzy "infoFarm"	
24				6									o			Odzyskiwanie wiedzy	
25							M	M	M	M	M	M	M	M	8	{RYSUNKI}	
26							v									Model wykładu1	
27								v								Fig. A. Tryby transferu wiedzy	
28									v							Fig. B. "Captive Mind (wise joke)"	
29										v						Fig. 1. infoMap - przykład oceny studentów	
30										v						Fig. 2 .. 7.4 - "infoFarm"	
31											v					Program przeszukiwania binarnego: infoMapy 1 .. 12	
32												v				Przestrzeń Systemu: infoMapy 13 .. 17	
33													v			Przykład: MIS : infoMapa 18	
34																	

Rys. 1. infoMapa zawierająca plan wykładu na temat infoMap

	1	2	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
1																															
2		A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A		{VIEW's Purpose}
3		v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v			InfoSYNTAX for jMAPS (tm) Technology
4		v																													Domain/Methodology InfoModels
5			v	v	v	v	v	v	v	v	v	v	v																		Primary infoSCHEMATA
6														v	v	v	v	v	v	v	v	v	v	v	v	v	v	v			Mapping Set Roles to Set Members Roles
7																												v			Cardinality
8		I																											8		{infoMODEL}
17		I	A	A	A	A	A	A	A	A	A	A	A																8		{Common infoSCHEMA}
25			A	A	A	A	A	A	A	A	A	A	A																10		{NARRATIVE}
36			I	I	I	I	I	I	I	I	I	I	I	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	14		{SET_ROLE}
37			c	c	c	c	c	c	c	c	c	c	c	o																	A::= Associative structure (of structures)
38			u												o															H::= Hierarchical structure
39				u												o														I::= Generalization (Inheritance). "Is-a" structure
40					p	p	p	p	p	p	p	p	p				o														O::= Dominant set
41					u													o													X::=Dependant set in a One-to-One Relationship
42						c													o												M::= Dependant set
43							c													o											R::= Relationship
44								c													o										F::= Flow definition
45									c													o									L::= directed graph with cycles and forks
46										c													o								S::= Sequence
47											c													o							G::= Guard or post-Guard
48												c													o						E::= Event structure
49																										o					V::= Value or instance
50																											o				T::= NavigationTrace i.e. history of execution
51													c															o			P::= "Part-of" in Aggregation
52														M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	15		{SET_MEMBER_ROLE}
53														v																	v::= column marker.
54														v									v				v			1..n::= "place in sequence"
55															v			v	v								v			v::= cell marker.
56															v															h::= root of a hierarchical structure
57															v															1 .. n::= element identification
58																v												v		w,p::= "whole", "part-of"
59																	v													o::= dominant set member
60																			v											a,k::= attribute type: simple, key
61																				v										O,o,m,u,[m;n]::= One, one, many, unary, [at least m; at most n]
62																					v									i,o,b::= flow direction: "in","out", "in/out"
63																						v								s,d,l,a,e::= "source","destination","loop","assertion","exception"
64																								v						t,f,c,T,F::="true","false","complementary","asserted t","asserted f"
65																									v					a,f,e,d::="arrival","finish", "enabled", "disabled"
66																										v				integer, real, char
67																														c,u,a::= use flow: "child", "uses", "ancestors"
68		A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A		{AUTHOR}
69		v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		InfoSCHEMA Copyright © 1991 W.M. Jaworski
70		v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		InfoMAP Copyright © 1991 W.M.Jaworski

Rys. 2. infoSyntax

Kolumny 15 i 16 odpowiadają obszarowi definicji zbiorów, kolumny od 1 do 13 obszarowi definicji relacji. Nazwy zbiorów i ich elementy są zdefiniowane w obszarze definicji zbiorów. Relacje między zbiorami oraz relacje między elementami zbiorów są zdefiniowane w obszarze definicji relacji.

W obszarze definicji zbiorów, nazwy zbiorów są zestawione w tej samej kolumnie, ujęte w nawiasy klamrowe i napisane dużymi literami. Zdefiniowane w przykładzie z rys. 1 nazwy zbiorów to:

{*ZASTOSOWANIE PRZEKROJU*}, {*CZĘŚCI WYKŁADU*} oraz {*RYSUNKI*}.

Elementy wchodzące w skład tych zbiorów są wymienione odpowiednio poniżej nazw tych zbiorów. Każda kolumna jMapy reprezentuje relację. Relacje są niepowtarzalne, z czego wynika, że nie powinno być dwóch identycznych kolumn. Dzięki temu unikamy redundancji danych. Każda komórka w obszarze definicji relacji odpowiadająca wierszowi elementu zbioru uczestniczy w tej relacji.

Uczestniczenie zbioru lub jego elementu w danej relacji jest oznaczane przez umieszczenie odpowiedniego elementu notacji (symbolu specjalnego) na skrzyżowaniu kolumny relacji z wierszem, w którym jest nazwa zbioru (lub nazwa elementu zbioru). Pusta komórka w jakimś miejscu kolumny relacji oznacza, że zbiór lub jego element w odpowiadającym wierszu nie uczestniczy w tej relacji. Symbole specjalne są podzielone na dwie grupy *Rola zbiorów* oraz *Rola elementów zbioru*. Stosowane oznaczenia oraz zależności pomiędzy tymi grupami (infoSYNTAX) zostały zamieszczone na rysunku 2. Należy w tym miejscu zaznaczyć, że składnia infoMAP jest otwarta. Użytkownik ma całkowitą swobodę w definiowaniu lub redefiniowaniu przyjętych oznaczeń, znaczenia obiektów infoMapy a także dopuszczalnych operacji na tych obiektach.

W jMapie istnieją dwa różne poziomy relacji. Pierwszy poziom specyfikuje relacje między zbiorami. Drugi poziom, niższy – relacje między elementami zbiorów. Pierwszy ze wspomnianych poziomów nosi nazwę **infoSCHEMY** (*infoRAME*). infoSCHEMA jest modelową strukturą dla reprezentowania określonej dziedziny wiedzy i stanowi ramową strukturę, która jest stopniowo wypełniana i uszczegółowiana aż do otrzymania infoMapy. Stąd infoMapy mogą być traktowane jako implementacja infoSCHEMY. Generalnie rzecz biorąc z jednej infoSCHEMY można otrzymać wiele infoMap. Podobnie jak w przypadku programów: na podstawie jednej specyfikacji można zakodować wiele programów. Przejście od infoSCHEMY do infoMapy wymaga udziału człowieka, podczas gdy operacja odwrotna może być przeprowadzana automatycznie.

	1	2	3	4	7	14	15	16
1								
2		A	A	A	A	A		{ZASTOSOWANIE PRZEKROJU}
8		M	M	H	O	16		{CZĘŚCI WYKŁADU}
25					M	8		{RYSUNKI}
34								

Rys. 3. infoSCHEMA dla rysunku 1

Na rysunku 3 jest przedstawiona infoSCHEMA dla przykładu z poprzedniego rysunku. W przypadku interpretowania relacji w infoMapie należy najpierw analizować relacje między zbiorami (czyli relacje przedstawione na infoSCHEMIE) a dopiero potem relacje między elementami zbiorów. W naszym przykładzie relację zawartą w kolumnie 7 możemy interpretować w sposób następujący:

Jeden (One) element zbioru {*CZĘŚCI WYKŁADU*} *jest powiązany z wieloma (Many) elementami zbioru* {*RYSUNKI*}.

Oznacza to, że między tymi zbiorami istnieje relacja typu 1:*M*. Ponadto czytamy z rysunku, że relacja w kolumnie 7 jest fragmentem relacji zagregowanej (ang. *Aggregation*) wskazującej na autora tej kolumny-relacji wymienionego w zbiorze {*AUTHOR*}. Stopniowo, przez uzupełnianie infoSCHEMY o relacje między elementami zbiorów otrzymujemy infoMapę taką jak na rysunku 1.

Ważnym elementem każdej infoMapy jest specjalny zbiór o nazwie {*VIEW*} lub {*VIEW'S PURPOSE*}. (W wypadku rysunku 1 i 3 jest to {*ZASTOSOWANIE PRZEKROJU*}. Każda relacja, stanowiąca przecież określony przekrój dziedziny jest opisana przez komentarze zawarte w tym zbiorze. Przykładowo na rysunku 1, w wierszu *Struktura wykładu* znajdują się trzy litery *v* w kolumnach 4, 5, 6 co oznacza, że te kolumny opisują strukturę wykładu. Oprócz tego istnieje specjalny przekrój oznaczony jako *cardinality* (liczebność). Związana z nim kolumna zawiera informacje o liczebności zbiorów. Informacje te są przydatne przy weryfikacji mapy.

Kolejny przykład – infoMapa przedstawiona na rysunku 4 zawiera zestawienie ocen dotyczących sytuacji w Polsce w 1992 roku, uzyskanych od różnych osób. Czytelnikom proponujemy samodzielne odczytanie informacji zawartych w tej infoMapie.

	1	2	3	4	5	6	7	8	9
1									
2									
3		A	A	A	A	A	A		{PUNKT WIDZENIA}
4		v	v		v				Polska widziana oczyma Polakow z Zachodu
5				v					Polska widziana oczyma Polakow w Polsce
6		v	v	v	v	v	v		Skala ocen; 1=wyjatkowo dobrze, 2=zupelnie dobrze, 3=troche dobrze, 4=neutralna, 5=troche zle, 6=zupelnie zle , 7=wyjatkowo zle
8		O	O	O	O	O	O		{OPINIODAWCA}
9		o							A. Targowski (na Zachodzie od 1980 r.)
10			o						J. Nowak Jezioranski (na Zachodzie od II Wojny Swiatowej)
11				o					S. Lem (w kraju)
12					o				W. Jaworski (na Zachodzie od 1968 r.)
13						o			Marian Kuras (roczne stypendium w Kanadzie 1991-92)
14							o		Andrzej Zaliwski (w kraju)
15		V	V	V	V	V	V		{DZIEDZINA}
16		6	3	6	6	5	5		Sytuacja polityczna
17		7	2	7	7	7	7		Strategia polityczna
18		6	2	6	6	4	5		Sytuacja gospodarcza
19		7	?	7	7	6	6		Strategia gospodarcza
20		?	1	?	5	4	4		Przywodcy/Dzialacze demokratyczni do 1989 r
21		6	5	6	6	6	5		Obecna kadra polityczna/politycy
22		6	?	5	6	6	7		Kultura polityczna
23		6	?	7	7	6	6		Wiedza polskiego spoleczenstwa o strategii gospodarczej
24		?	?	?	6	7	6		Wiedza polskiego spoleczenstwa o pozycji Polski w swiecie
25		?	?	?	6	3	4		Prognoza powaznych inwestycji/pozyczek zagranicznych
26									
27									
28		A	A	A	A	A	A		{ZRODLO}
29		v	v	v	v	v	v		K.M. Ford and all. ,"An Approach to Knowledge Acquisition Based on the Structure of Personal Construct Systems", IEEE Trans. on Knowledge and Data Eng., March 1991 pp 78-88
30		v	v	v	v	v	v		Z. Pawlak, "Rough Sets: Theoretical Aspects of Reasoning about Data", De Kuyper, Amsterdam, 1992
31		v							A. Targowski, "Rewolucja zjada wlasne dzieci", Studium Newsletter, January 1992
32			v						J. Nowak Jezioranski, "W odpowiedzi Andrzejowi Targowskiemu", Studium Newsletter, February/March 1992
33				v					S. Lem, "Swiat i Polska", Kultura, Paryz, Nr 5/536, 1992
34									

Rys. 4. Ocena sytuacji w Polsce w 1992 r. przedstawiona w postaci infoMapy

Jak już mieliśmy możliwość zauważyć notacja infoMap jest notacją bardzo pojemną i elastyczną. W celu pokazania walorów tej techniki notacyjnej oraz przekonania, iż jest rzeczywiście notacją uniwersalną, mogącą opisywać każdy aspekt SI, w dalszej części artykułu prezentujemy kilka przykładów. Większość z nich dotyczy samego programowania, a jeden dziedziny bardzo odległej od programowania.

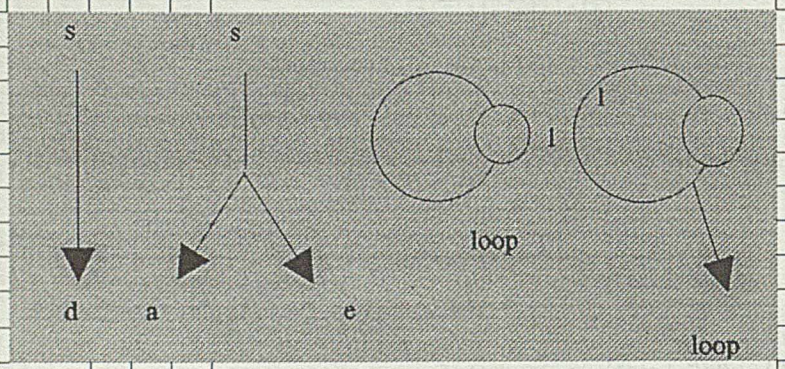
Zastosowania w programowaniu

W celu ilustracji możliwości opisu przy zastosowaniu konwencji infoMap wybraliśmy sposób reprezentacji najbardziej elementarnych struktur sterujących i przykładów programów.

Właściwości elementarnych struktur sterowania są dyskutowane w wielu publikacjach (np. [1, 2, 3, 16]). Graf przepływu sterowania tych struktur sterujących może być reprezentowany przez cztery elementarne grafy oraz ich złożenie. Te podstawowe konstrukcje są określane jako: *Sekwencja, Sekwencja z asercją* lub *wyjątkiem* (ang. *exception*), *Pętla, pętla z wyjściem*. Rys. 5a obrazuje te cztery elementarne konstrukcje a rysunek 5b sposób ich przedstawienia w postaci infoMap. Nazwy tych czterech podstawowych konstrukcji zostały wymienione jako elementy zbioru {*GENERIC GRAPH*}. Przecięcie wiersza, w którym jest wymieniona nazwa konstrukcji z kolumną opisującą tę konstrukcję jest oznaczone literą *o*. Litera *s* została użyta do określenia źródła (ang. *source*) a litera *d* jako oznaczenie stanu docelowego (ang. *destination*). Symbol *l* oznacza pętlę (ang. *loop*), *a* – asercję (ang. *assertion*), zaś *e* – wyjątek (ang. *exception*).

Na rysunku 6 z lewej strony pokazano w postaci kodu strukturalnego zbliżonego do języka Pascal niektóre konstrukcje stosowane w programach. Środkowa część rysunku przedstawia te same konstrukcje w postaci infoMap. Zwróćmy uwagę na zbiór {*PRECONDITION*} na rysunku 6. Jeśli wymieniony w nim warunek *V* jest spełniony (*true*) zachodzi sytuacja przedstawiona w pierwszej kolumnie. Jest wykonywana sekwencja akcji *S*1 i *S*2. Litera *S* sygnalizuje operacje sekwencyjne. Cyfry '1' i '2' pod literą *S* określają wówczas kolejność wykonywania operacji. Jeśli warunek *V* nie jest spełniony (ang. *false*), to mamy sytuację opisaną w drugiej kolumnie. Tak jak w poprzednim przypadku również następuje przejście z jednego stanu do drugiego, ale akcje *S*1 i *S*2 nie mają miejsca.

Na rysunku 7 pokazano kod strukturalny dla programu przeszukiwania binarnego (a) oraz odpowiadającą mu reprezentację w postaci infoMapy (b). Pozycje *O, L, G, S* i *G* określają role jakie zostały przydzielone odpowiednim zbiorom tj. {*Transition*}, {*State*}, {*preCondition*}, {*Action*} oraz {*postCondition*}. Znaczenie poszczególnych pozycji jest następujące:
Każde (*each*) przejście (*transition*) łączy (*Links*) stany (*States*). Każde przejście jest zaimplementowane jako sekwencja (*Sequence*) akcji kontrolowanych (*Guarded*) przez warunki wstępne (*preconditions*) oraz kontrolowana (*asserted*) przez warunki końcowe (*postconditions*).
Litera A pozwala użytkownikowi na zaznaczenie faktu, że kilka kolumn jest zgrupowane w relację poziomą. Litera *S* oznacza



branch — branch with fork

(a) Reprezentacja graficzna

O	O	O	O	{GENERIC GRAPH}
o				branch from s to d
	o			branch from s to either a or e
		o		loop
			o	loop with exit to e
L	L	L	L	{NODE/STATE}
s				start
d				destination
	s			start
	a			asserted destination
	e			exception destination
		l		loop node
			l	loop node
			e	exception destination
A	A	A	A	{AUTHOR}
v	v	v	v	InfoSCHEMA Copyright © 1991 W.M. Jaworski
v	v	v	v	InfoMAP Copyright © 1991 W.M.Jaworski

(b) Reprezentacja w postaci infoMAPY

Rys. 5. Cztery podstawowe grafy przepływu sterowania

relację pionową i sekwencyjne uporządkowanie elementów zbioru.

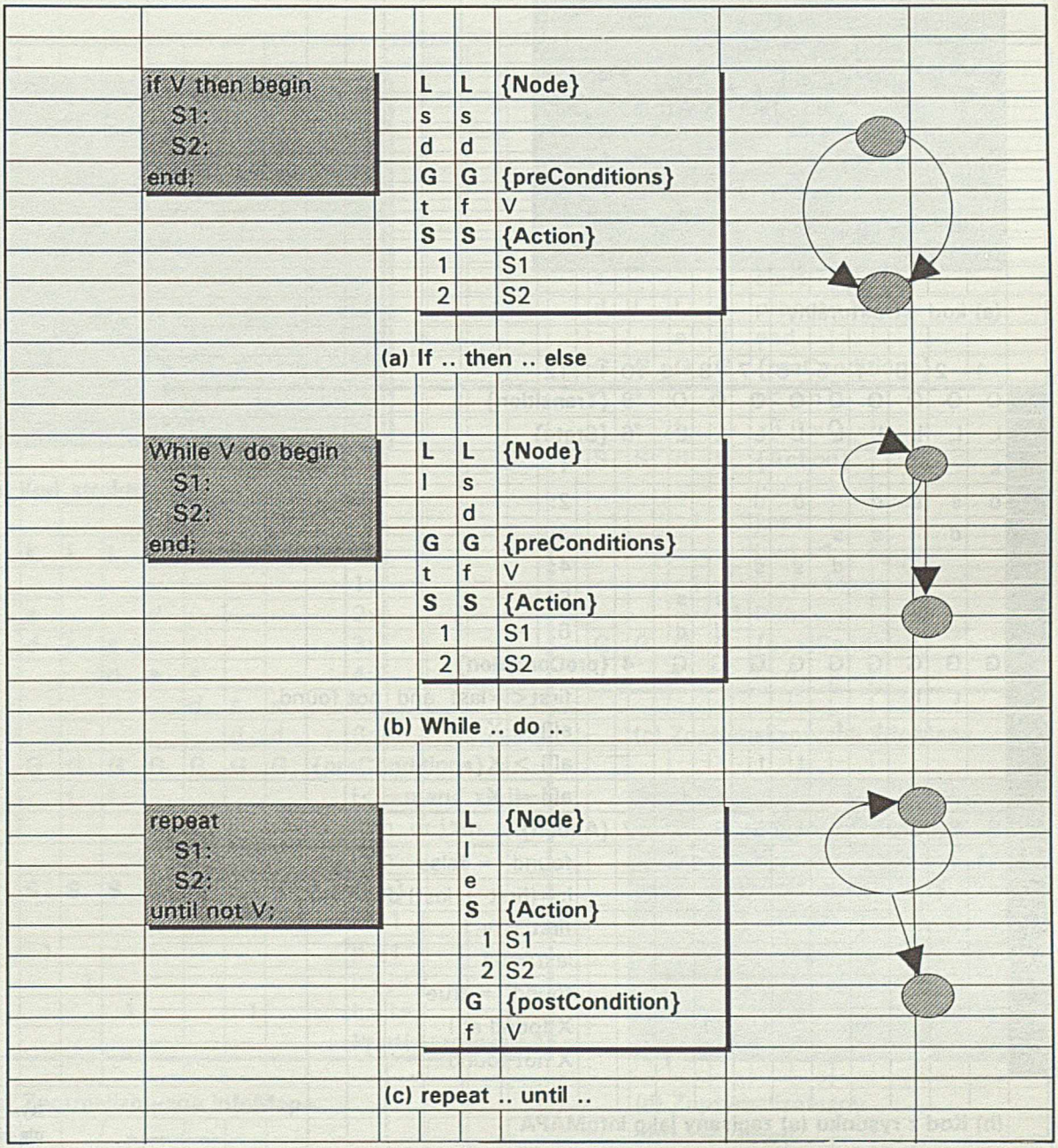
Poniżej zamieszczamy przykład oparty na następującym zadaniu, które sformułował Dijkstra w [4]:

„Niech X będzie macierzą N × N zawierającą liczby całkowite. Należy napisać program, który wyświetla numer pierwszego zerowego wiersza macierzy X, o ile taki istnieje."

Rozwiązanie tego problemu pochodzące z [15] i jego translację na infoMapę przedstawia rysunek 8 (a) i (b). Przez przetworzenie wierszy i kolumn infoMapy otrzymano zoptymalizowaną infoMapę (c) oraz zoptymalizowany kod źródłowy (d). Inne rozwiązanie tego samego problemu [13] oraz rozwiązania o nazwach *Pancode* i *EPN* zestawiono w postaci infoMapy na rys. 9. Rozwiązanie *Pancode* reprezentowane jako infoMapa (nie zamieszczona) ma 4 węzły-stany, 6 rozgałęzień-przejść, 3 warunki wstępne i 5 akcji. Oryginalne rozwiązanie *EPN* ma 5 węzłów-stanów, 6 rozgałęzień-przejść, 3 warunki wstępne, 5 akcji i 1 warunek końcowy. Rozwiązanie w języku Pascal zaczerpnięte z [15] reprezentowane w postaci infoMapy (rysunek (b)) ma 6 węzłów-stanów, 8 rozgałęzień-przejść, 3 warunki wstępne i 5 akcji. Rozwiązanie zoptymalizowane (c) ma 3 węzły-stany, 4 rozgałęzienia-przejścia, 2 warunki wstępne, 5 akcji i 1 warunek końcowy. Rozwiązanie zostało zoptymalizowane – otrzymane przez połączenie stanów, przejść, przez eliminację redudantnych warunków i akcji.

Kompletna infoMapa z rysunku 10 zawiera zestawienie dwóch algorytmów przeszukiwania binarnego. Zostały one oznaczone *Binary Search Program Version* 1 i *Binary Search Program Version* 2 jako elementy zbioru {*PROCESS*}. Kolumny oznaczone literami *v* w wierszu *Binary Search Program Version* 1 opisują ten algorytm. Analogiczna sytuacja zachodzi w przypadku drugiego algorytmu.

infoMapę możemy zacząć analizować praktycznie w dowolnym miejscu w zależności od tego co nas interesuje. Zacznijmy może od zbioru {*preCONDITION*}. Zawarty w nim jest warunek *first* < = *last* i jeśli jest on spełniony (tzn. w jego wierszu jest litera *t*), analizujemy kolumnę 3, jeśli (*f*) – kolumnę



Rys. 6. Grafy przepływu sterowania

4. Zakładamy, że warunek jest spełniony (*t*) i analizujmy kolumnę 3. Idąc w górę kolumny napotykamy litery *s* i *d*, które mówią, że następuje przejście ze stanu 1:*Start* do stanu 2:*Process element a[i]* (przetwarzanie elementu *a[i]*). Jeszcze wyżej w zbiorze {*TRANSITION*} badanej kolumnie odpowiada *try middle* (spróbuj element środkowy).

Jeśli warunek *first <= last nie jest spełniony (f)*, mamy sytuację opisaną w kolumnie 4. Następuje przejście od stanu 1:*Start* do stanu 3:*End*. Odpowiadająca pozycja dla kolumny 4 w zbiorze {*TRANSITION*} to *failure* i w zasadzie nie ma czemu się dziwić. Przeszukiwanie w takiej sytuacji nie ma sensu. Każda kolumna infoMapy przedstawionej na rysunku 10 opisuje inny stan programu. InfoMapa 10 stanowi zestawienie możliwych stanów dwóch porównywanych programów.

Z powyższych przykładów wynika, że infoMapy mogą być wykorzystywane do porównywania oraz optymalizowania algorytmów. Możliwe jest także sprawdzenie czy dany „nowy" nie jest w gruncie rzeczy tym samym, nieco zmienionym, już istniejącym wcześniej algorytmem oraz, czy rzekomo „nowa" metoda rzeczywiście wnosi coś nowego do inżynierii oprogramowania.

Jako kolejny przykład opiszemy rozwój analizatora synktatycznego oryginalnie dyskutowany w [14]. Załóżmy, że na pewnym poziomie tworzenia tego analizatora otrzymaliśmy już graf przepływu danych pokazany na rysunku 11 (będący adaptacją rysunku 18 z [18]. Yau daje zbiór relacji między tym grafem przepływu a diagramem struktury z rysunku 13. Graf przepływu i diagram struktury są wyrażone jako infoMapy na ry-

```
1: found:=false
2: while( first <= last)
     and  not  found do begin
                i := ( first + last ) div 2;
3:              if a[i]<X then first := i+1
4:              else if a[i]>X then last:=i-1
                     else found := true;
   end;
5: if a[i] = X then
              X found at i
   else X not found;
6:
```

(a) kod strukturalny

	1	2	3	4	5	6	7	8	9	10	11	12	
O	O	O	O	O	O	O	O	O	O	9			{Transition}
L	L	L	L	L	L	L	L	L	L	6			{State}
s												1:	
d	s	s	d		d	d						2:	
	d		s	s								3:	
				d	s	s						4:	
							s	s				5:	
							d	d				6:	
G	G	G	G	G	G	G	G	G	G	4			{preCondition}
	t	f											first <= last and not found
			t	f									a[i] < X
					t	f							a[i] > X
							t	f					a[i] = X
S	S	S	S	S	S	S	S	S	S	7			{Action}
1													found := false
	1												i:=(first + last) div 2
			1										first:=i+1
					1								last:=i-1
							1						found := true
								1					X found at i
									1				X not found

(b) Kod z rysunku (a) zapisany jako infoMAPA

Rys. 7. Algorytm wyszukiwania binarnego

sunkach 12 i 14. Przykładowo schemat rysunku 12(a) mówi nam o składnikach rysunku 12(b): Są to *OPERATION* i *dataOBIECT*. Górna część infoMapy z nagłówkiem złożonym z sześciu liter *O* na rysunku (12b) definiuje sześć operacji odpowiadających sześciu wierzchołkom grafu przepływu danych[1].

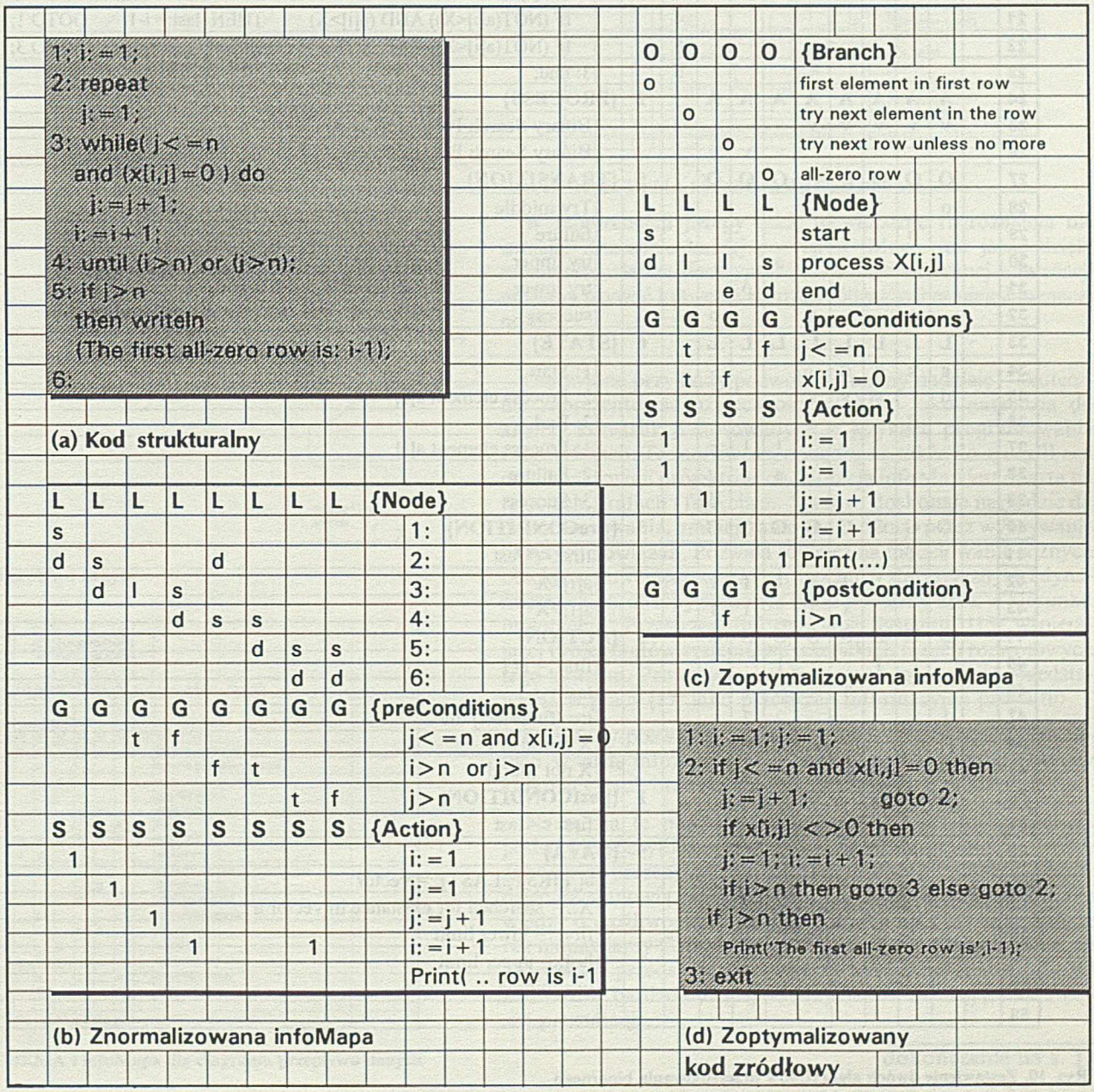
Dolna część infoMapy opisuje dwie binarne relacje od operacji do obiektów. Wiersz zawierający *g* odpowiada obiektowi wejściowemu (lub inaczej danemu: *given*). Wiersz odpowiadający obiektowi wyjściowemu jest oznaczony symbolem *r* (*result*). Schemat infoMapy na rysunku 14 jest bogatszy niż ten z rysunku 12. W części górnej po lewej stronie użyto symbolu *H* do oznaczenia formy drzewiastej diagramu struktury, symbole *g* i *r* mają takie samo znaczenie jak zdefiniowane wcześniej. Symbol *b* wskazuje, że dany obiekt jest zarówno wprowadzany jak i zwracany przez operację. Przykładowo *GetWords* otrzymuje nazwę pliku z *GetFileName* i wysyła ją do *split*. Użycie kilku różnych symboli w treści jMapy odpowiada użyciu kilku rodzajów krawędzi w grafie.

Niejednoznaczności notacji można uniknąć, jeśli wybierze się symbole z wielkiego alfabetu (małe i duże litery, cyfry, litery greckie itd.). Liczba rodzajów krawędzi jaka może być rozróżniana (jasne, ciemne, przerywane itp.) jest ograniczona. Prezentacja grafów wymaga ponadto właściwego oprogramowania,

{Wersja}	A	A	A	A	
Zoptymalizowana	v				
Pancode		v			
EPN-long			v		
EPN-short				v	
{Atrybuty}	V	V	V	V	
branch	2	5	5	5	
branch with fork	0	0	1	0	
loop	1	1	0	1	
loop with exit	1	0	0	0	
state	3	4	5	4	
transition	4	6	6	6	
preCondition	2	3	3	3	
action	5	5	5	5	
postCondition	1	0	1	0	

Rys. 9. Zestawienie różnych rozwiązań zadania

[1] Graf skierowany $G<\{O\},\{D\}>$, może być określony jako zbiór wierzchołków oznaczony literą O i zbiór krawędzi oznaczony literą D. Do identyfikacji krawędzi zbioru G, używa się liter *g*, *r* oraz *b* (*given*, *result*, *both*).



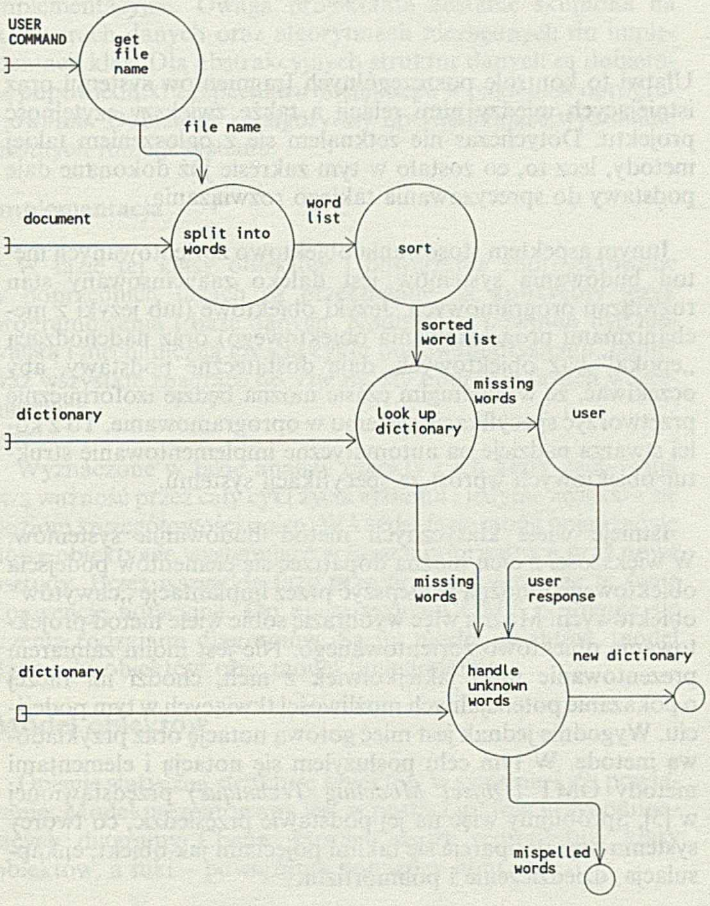
Rys. 8. Optymalizacja kodu źródłowego programu przy pomocy infoMapy

	2	3	4	5	6	7	8	9	10	11	12	13	14
1													
2	A	A	A	A	A	A	A	A	A	A	A		{AUTHOR}
3	v	v	v	v	v	v	v	v	v	v	v		InfoSCHEMA Copyright © W.M. Jaworski
4	v	v	v	v	v	v	v	v	v	v	v		InfoMAP Copyright © WMJ
5	A	A	A	A	A	A	A	A	A	A	A		{VIEW's Purpose}
6	v												Pascal-like Code for Binary Search
7		v	v	v	v	v	v	v	v				Pascal-like Code rewritten as infoMAP (Version 1 and 2)
8										v			infoMap rewritten as Pascal-like Code
9											v		Cardinality
10	S									S	23		{STATEMENT}
11	1												Start: IF (first <= last) THEN i:= (first+last) div 2;
12	2												ELSE X not found; GOTO Exit;
13	3												IF a[i]<X THEN first := i-1; GOTO Start;
14	4												ELSE
15	5												IF a[i]>X THEN last:= i-1 GOTO Start;
16	6												ELSE X found at i GOTO Exit;
17	7												Exit:
18										1			1: IF (first <= last) THEN i:= (first+last)div 2 GOTO 2;
19										2			IF NOT(first <= last) THEN X not found GOTO 3;
20										3			2: IF a[i]<X THEN first := i+1 GOTO 1;
21										4			IF (NOT(a[i]<X)) AND (a[i]>X) THEN last:= i-1 GOTO 1;
22										5			IF (NOT(a[i]<X)) AND NOT(a[i]>X) THEN X found at i GOTO 3;
23										6			3: End;
24		A	A	A	A	A	A	A	A		2		{PROCESS}
25		v	v	v	v	v							Binary Search Program Version 1
26							v	v	v				Binary Search Program Version 2
27		O	O	O	O	O	O	O	O		5		{TRANSITION}
28		o											Try middle
29			o										failure
30				o			o						try upper
31					o			o					try lower
32						o			o				success
33		L	L	L	L	L	L	L	L		6		{STATE}
34		s	s	d	d								1: Start
35		d		s	s	s							2: Process element a[i]
36			d			d							3: End
37							l	l	s				1: Process element a[i]
38							a	a					2: Failure
39									d				3: Success
40		G	G	G	G	G	G	G	G		3		{preCONDITION}
41		t	f										first <= last
42				t	f	f	t	f	f				a[i]<X
43					t	f		t	f				a[i]>X
44		S	S	S	S	S	S	S	S		5		{ACTION}
45				1			1						first := i+1
46					1			1					last:= i-1
47		1					2	2					i:= (first+last) div 2;
48						1			1				X found at i
49			1										X not found
50		G	G	G	G	G	G	G	G		1		{postCONDITION}
51							f	f					first <= last
52		F	F	F	F	F	F	F	F		5		{DATA}
53				i	i	i	i	i	i				a[FIRST..LAST]::= vector
54				i	i	i	i	i	i				X::= searched for existance in vector a
55		i	i	o			b	i					first::= lower limit
56		i	i		o		i	b					last::= upper limit
57		o		i	i	i	b	b	i				i::= middle
58													

Rys. 10. Zestawienie dwóch algorytmów przeszukiwania binarnego

gdyż trudno jest wyświetlić na ekranie graf mający wiele wierzchołków. Nieetykietowana krawędź grafu zawiera tylko jeden bit informacji; osobie oglądającej mówi tylko czy dwa węzły są w relacji. W celu zinterpretowania grafu trzeba dokładnie wiedzieć, co oznacza obecność krawędzi (kontekst).

W grafie przepływu danych na rysunku 11 każda krawędź odpowiada przepływowi danych pewnego rodzaju. Przykładowo, krawędź *sorted word list* łączy słowa z węzła *sort* z węzłem *lookup dictionary*. Na diagramie struktury (rys. 13) węzły są połączone, jeśli jest jakikolwiek przepływ danych między nimi. Strzałki wskazują zależność, a nie kierunek przepływu. Przykładowo: węzeł *lookup dictionary* otrzymuje posortowane słowa a zwraca słowa nieznane.

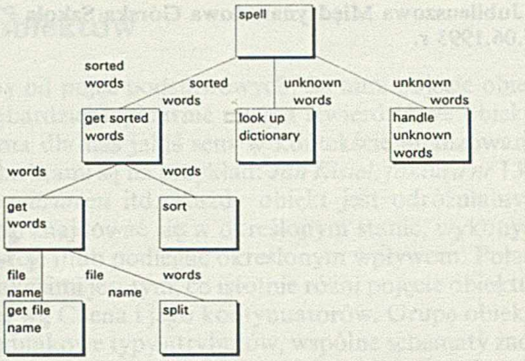


Rys. 11. Graf przepływu danych

	O	{OPERATION}
	N	{dataOBJECT}
(a) infoSCHEMA dla grafu przepływu danych		

O	O	O	O	O	O	{OPERATION}
o						lookup dictionary
	o					handle unknown words
		o				sort
			o			get file name
				o		split into words
					o	user
N	N	N	N	N	N	{dataOBJECT}
g	g					dictionary
	r					new dictionary
				g		document
			g			user command
	r					mispelled words
			r	g		file name
		g		r		word list
g		r				sorted word list
r	g				g	missing words
	g				r	user response
(b) infoMapa dla grafu przepływu danych						

Rys. 12. infoSCHEMA i infoMapa dla diagramu przepływu danych



Rys. 13. Diagram struktury dla grafu z rysunku 11

H	O	{OPERATION}
	N	{dataOBJECT}
(a) infoSCHEMA dla diagramu struktury		

H	H	H		O	O	O	O	O	O	O	O	{OPERATION}
p				o								Spell
1	p				o							Get sorted words
2						o						Lookup dictionary
3							o					Handle unknown words
	1	p						o				Get words
	2								o			Sort
		1								o		Get file name
		2									o	Split into words
				N	N	N	N	N	N	N	N	{dataOBJECT}
								b		r	g	file name
					b			r	g		r	words
				b	b	g			r			sorted words
				b		r	g					unknown words
												user response
(b) infoMapa dla diagramu struktury												

Rys. 14. infoSCHEMA i infoMapa dla diagramu struktury

W reprezentacji jMapy takie niewidoczne rozróżnienia nie istnieją. Znaczenie każdej pozycji w jMapie jest jasne dzięki zawartości wiersza deskryptorów. W prezentowanej konwencji możemy używać krawędzi różnych rodzajów bez niejednoznaczności.

* * *

Omówione przykłady pozwoliły – mamy nadzieję – skutecznie zaprezentować użyteczność infoMap jako narzędzia do analizy konstrukcji stosowanych w językach programowania i algorytmach. Te możliwości odnoszą się także do specyfikacji oraz konstrukcji projektowych. Notacja infoMap jest oparta na relacjach, grafach i funkcjach. Stanowi doskonałe narzędzie do dokumentowania, zmniejszania redundancji przez wykrywanie ukrytych powiązań. Pozwala również na modelowanie różnych obiektów z uwzględnieniem ich aspektów statystycznych i dynamicznych. infoMapy stanowią także poważny krok w kierunku utworzenia „Przestrzeni Informacyjnej Systemu" [15], wspierającej projektantów systemu we wszystkich fazach rozwojowych tego systemu. Propozycja implementacji tej idei jest przedstawiona w pracy [17] jako przestrzeń informacyjna projektu.

infoMapy pozwalają na bardzo łatwe przetwarzanie zawartych w nich informacji np. przez użycie typowych arkuszy kalkulacyjnych oraz narzędzi rozszerzających te arkusze [5].

Narzędzia te pozwalają na generację kodu programu na podstawie specyfikacji programu w postaci infoMapy. I żeby zakończyć prezentację możliwości infoMap możemy powiedzieć, iż stanowią one narzędzie pozyskiwania i gromadzenia wiedzy o możliwościach nie w pełni jeszcze odkrytych. Jeszcze raz namawiamy Czytelnika na zmierzenie się z rysunkiem 4, na którym jest przedstawiona w sposób ustrukturyzowany wiedza z dziedziny bardzo odległej od przykładów ilustrujących niniejszą prezentację.

dokończenie na s. 13

Piąta Jubileuszowa Międzynarodowa Górska Szkoła PTI Szczyrk
21–25.06.1993 r.

ZBIGNIEW BENEDYKT
Zakładowy Ośrodek Informatyki BEFAMA
Bielsko-Biała

Co daje obiektowe podejście do analizy i projektowania systemów informatycznych?

W poszukiwaniu technik i narzędzi ułatwiających tworzenie oprogramowania oraz podnoszących jego niezawodność dobrnęliśmy wreszcie do technik obiektowo zorientowanych. Tak, tak, minęło już ćwierć wieku od pojawienia się SIMULI 67 – niekwestionowanego prekursora tego podejścia. Język stworzony dla potrzeb modelowania, z definicji musi wywierać wpływ na sposób percepcji i opisu rzeczywistości. Można stąd wnioskować, że techniki obiektowe odziedziczyły tę cechę po swoim „wielkim przodku". I tak jest istotnie.

Zawsze istniała jakaś współzależność pomiędzy technikami stosowanymi przez programistów i projektantów oraz analityków systemów. W tym przypadku zależność ta jest znacznie bardziej wyrazista i nic w tym dziwnego, że oprócz programowania obiektowego istnieje obiektowe podejście do modelowania i projektowania systemów. Metody (techniki) rozwijane przez tak długi okres nie powodują rewolucyjnych zmian w metodach dotychczas stosowanych. Oczywiste, że to co było cenne i dało się zaadaptować wcześniej, weszło już do praktyki. Istnieje jednak określony próg, po przekroczeniu którego zaczynamy nazwom metod dodawać kolejny przymiotnik. **Obiektowo zorientowana metoda** (technika), to taka, w której rozpatrujemy świat i organizujemy system informatyczny jako kolekcję dyskretnych obiektów łączących ze sobą zarówno dane (stany) jak i zachowanie się bytów istniejących w postrzeganym świecie.

Większość dotychczas stosowanych metod rozdziela opis struktur danych od opisu procesów, aby w dalszych etapach analizy skojarzyć te dwa opisy dokonując weryfikacji ich zupełności i niesprzeczności. Dla każdego z etapów są stosowane różne techniki notacyjne (diagramy związków encji, przepływów danych, historii życia encji). Prowadzi to do swoistego dualizmu „danowo-akcyjnego", a przecież rzeczywiste byty są jednorodne i postrzegać je można tylko wtedy, gdy następuje interakcja między podmiotem i przedmiotem. Zatem obiekt jest konstrukcją bardziej adekwatną do opisu świata, niż encja i proces! Biorąc pod uwagę to, co zostało wyrażone wyżej, można oczekiwać, że metody obiektowe wyeliminują oddzielne diagramy dla poszczególnych etapów. wprowadzając jeden typ diagramu odpowiedni dla całego procesu tworzenia systemu. Ułatwi to kontrolę poszczególnych fragmentów systemu oraz istniejących między nimi relacji a także zwiększy czytelność projektu. Dotychczas nie zetknąłem się z ogłoszeniem takiej metody, lecz to, co zostało w tym zakresie już dokonane daje podstawy do sprecyzowania takiego rozwiązania.

Innym aspektem stosowania obiektowo zorientowanych metod budowania systemów jest daleko zaawansowany stan rozwiązań programowych. Języki obiektowe (lub języki z mechanizmami programowania obiektowego) oraz nadchodząca „epoka" baz obiektowych, dają dostateczne podstawy, aby oczekiwać, że w niedługim czasie można będzie izoformicznie przetworzyć specyfikację systemu w oprogramowanie. To z kolei stwarza nadzieje na automatyczne implementowanie struktur obiektowych wprost ze specyfikacji systemu.

Istnieje wiele klasycznych metod budowania systemów. W większości z nich można dopatrzeć się elementów podejścia obiektowego i można je ulepszyć przez implantację „chwytów" obiektowych. Można więc wyobrazić sobie wiele metod projektowania obiektowo zorientowanego. Nie jest moim zamiarem prezentowanie tutaj jakiejkolwiek z nich, chodzi mi raczej o pokazanie potencjalnych możliwości tkwiących w tym podejściu. Wygodnie jednak jest mieć gotową notację oraz przykładową metodę. W tym celu posłużyłem się notacją i elementami metody OMT (*Object Modeling Technique*) przedstawionej w [5]. Spróbujmy więc na jej podstawie prześledzić, co twórcy systemu daje podparcie się takimi pojęciami jak obiekt, enkapsulacja, dziedziczenie i polimorfizm.

Ogólna charakterystyka metody OMT

Istotą metody OMT jest stworzenie modelu rzeczywistości, który posłuży do zbudowania systemu informatycznego. Model ten powinien wiernie oddawać syntaktyczne i semantyczne własności tego fragmentu rzeczywistości, który pozwoli nam na rozwiązanie problemów stawianych do rozwiązania przez system. Model powinien być łatwo implementowany, tzn. powinna istnieć możliwość łatwego jego przetransponowania na system informatyczny. Oczywiście model taki nie powstanie w jednym akcie twórczym. Proces jego tworzenia rozciąga się w czasie i wymaga zaangażowania różnej wiedzy specjalistycznej. Metoda zakłada podział całego procesu na cztery fazy: analizy, projektowania systemu, projektowania obiektów oraz implementacji.

Analiza

Zaczynając od sformułowania problemu, analityk buduje model sytuacji w świecie rzeczywistym, pokazując w nim istotne jego właściwości. Analityk musi współpracować z zamawiają-

Mgr **ZBIGNIEW BENEDYKT** ukończył w 1976 r. studia matematyczne na Uniwersytecie Śląskim o specjalności Metody numeryczne. Pracuje w Zakładowym Ośrodku Informatyki BEFAMY jako analityk. Zajmuje się metodami budowania systemów informatycznych z zastosowaniem narzędzi wspomagających oraz bazami danych.

cym system, aby zrozumieć jego potrzeby oraz określić zasięg „terytorialny" systemu. W swojej pracy musi precyzyjnie określić, co system ma realizować, nie zajmując się tym, jak będzie to realizowane.

Projekt systemu

Projekt systemu obejmuje generalne decyzje dotyczące architektury systemu, podział docelowego systemu na podsystemy, decyzje dotyczące charakterystyk eksploatacyjnych, wyboru strategii rozwiązania problemu oraz próbny podział zasobów.

Projekt obiektów

Projekt obiektów obejmuje uzupełnienie modelu o szczegóły implementacyjne. Uwaga projektanta zostanie skupiona na strukturach danych oraz algorytmach niezbędnych do implementacji klas. Dla abstrakcyjnych struktur danych są dobierane odpowiedniki z repertuaru dostępnego w pakietach oprogramowania. Dobierane są algorytmy optymalizujące wykonanie istotnych funkcji systemu.

Implementacja

W fazie tej klasy obiektów i powiązania zaprojektowane w poprzednich fazach są przekładane na konkretny język programowania i bazę danych. Kodowanie powinno być niewielką i mechaniczną częścią cyklu budowania systemu, ponieważ wszystkie znaczące decyzje należy podjąć w fazach wcześniejszych.

Wyznaczone w fazie analizy obiekty i ich klasy zachowują swą ważność przez cały cykl życia systemu i jedynie zwiększa się poziom szczegółowości opisu. W każdej fazie mogą pojawiać się nowe obiekty nie występujące w fazach poprzednich oraz nowe metody. Przez wszystkie fazy przechodzi się stosując te same konwencje notacyjne. Do opisu systemu OMT posługuje się trzema rodzajami diagramów. Są to: model obiektów, model dynamiki obiektów oraz model funkcjonalny.

Model obiektów

Opisuje statyczną strukturę obiektów w systemie i ich powiązania. Model prezentowany jest w postaci grafów przypominających diagramy Chena, których węzły reprezentują klasy obiektów, a łuki – powiązania pomiędzy nimi.

Model dynamiki

Prezentuje schematy zmiany stanów obiektów systemu. Jest przedstawiany w postaci diagramów zmian stanów. Są to grafy, których węzłami są stany, a łukami – ich zmiany powodowane przez zdarzenia.

Model funkcjonalny

Prezentuje przepływy danych w systemie. Jest przedstawiany w postaci diagramów przepływu danych.

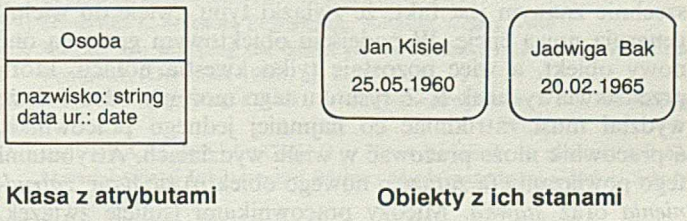
Te trzy modele stanowią ortogonalne części opisu systemu. Wprawdzie model obiektów jest podstawowym, ale nie przedstawia on dynamiki obiektów, i stąd konieczne są pozostałe dwa modele. Twierdzę, że metodę i diagramy można zmodyfikować tak, aby wszystkie informacje o systemie zamknąć w jednym tylko modelu, ale to wymaga szczegółowego uzasadnienia, wykraczającego poza ramy tego artykułu. Ponieważ najwięcej korzyści z podejścia obiektowego widać w modelu obiektów, tylko on będzie przedmiotem dalszych rozważań.

Model obiektów

Zacznijmy od pojęć podstawowych. Są nimi pojęcie obiektu i klasy. Najbardziej lapidarnie można stwierdzić, że obiektem jest to, co ma dla nas jakiś sens w kontekście analizowanego systemu. Obiektami są na przykład: *Jan Kisiel, faktura nr* 13/93, *wydział narzędziowni* itd. Każdy obiekt jest odróżnialny od innych, może znajdować się w określonym stanie, wykonywać określone akcje i/lub podlegać określonym wpływom. Połączenie stanu z akcjami jest tym, co istotnie różni pojęcie obiektu od pojęcia encji wg Chena i jego kontynuatorów. Grupę obiektów mających jednakowe typy atrybutów, wspólne schematy zachowań, identyczne związki z innymi obiektami oraz jednakową semantykę nazywamy klasą obiektów. Obiekt kojarzymy z inkarnacją bytu, zaś klasę – z jego typem. Ale uwaga: klasa jest również obiektem wyznaczonym przez wspólne (dla wszystkich obiektów klasy) stany i zachowania. Na przykład: liczba jest klasą obiektów, takich jak: liczba całkowita, wymierna itd., ale sama liczba jest obiektem; jeśli więc interesują nas stany magazynowe, to materiał o danym identyfikatorze jest obiektem, ale z drugiej strony to klasa obrotów magazynowych. Na diagramach klasę przedstawiamy w postaci prostokąta, a obiekt w postaci czworokąta o zaokrąglonych rogach (rys. 1).

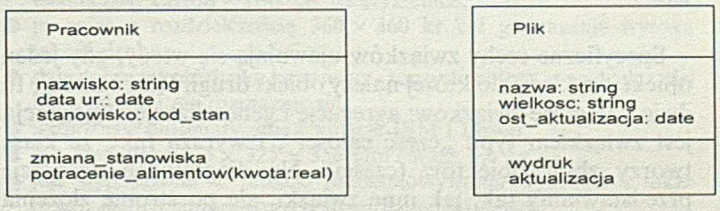
Rys. 1

Zazwyczaj obiekty mają atrybuty, których wartości określają stan obiektu. Przedstawiając na diagramie obiekt, wpisujemy przykładowe wartości atrybutów (stan obiektu). Przedstawiając klasę podajemy nazwę oraz typ atrybutu (rys. 2). Często ze względów implementacyjnych posługujemy się specjalnym atrybutem służącym do identyfikacji obiektu. Na diagramach obiektów takich atrybutów nie umieszcza się, ponieważ nie należą one do świata rzeczywistego.



Rys. 2

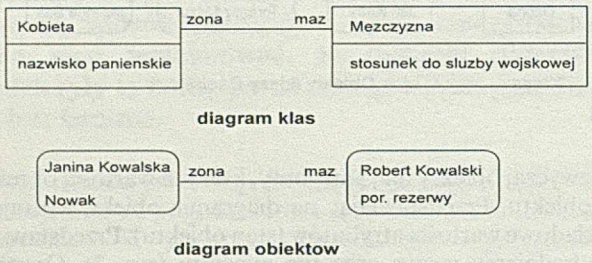
Oprócz stanów obiekty miewają metody (akcje), które mogą je wykonywać lub „znosić", np. pracownik może zmieniać stanowisko, być zwalniany, zmuszony do płacenia alimentów itd. Metody specyfikuje się wyłącznie na diagramach klas. Właściwie są to procedury lub funkcje, a zatem ich specyfikacja odpowiada tym kategoriom. Przykłady tego przedstawia rys. 3.



Rys. 3

Świat byłby mało interesujący, gdyby obiekty nie wchodziły we wzajemne związki. Praca analityka staje się frapująca dopiero wtedy, gdy te związki może odkrywać. Prowadzi to często również do odkrycia nie uświadamianych wcześniej obiektów. W „klasycznych" metodach brakuje mechanizmów pozwalających wystarczająco dokładnie zarejestrować semantykę związków encji. Powód wydaje się oczywisty. Znaczenie związków ujawnia się często przez działania (metody) obiektów, te zaś są rozpatrywane poza diagramami związków encji. Siłą podejścia obiektowego jest większa „wykrywalność" zagadnień, których mógł nam nie przekazać użytkownik systemu. Mamy również mechanizmy pozwalające te cechy zanotować i przekazać do dalszej obróbki przez projektantów i programistów.

Co w tym zakresie oferuje OMT? Zacznijmy od rozróżnienia powiązania obiektów od związków między klasami. Jeśli weźmiemy dwa obiekty, to mogą one być ze sobą powiązane np. *Maria Kowalska* może być żoną *Jana Nowaka*. Powiązanie to wyraża istniejący stan. Jeśli zaś spojrzymy na związek między klasami, to zobaczymy, że jest to tylko szablon możliwej sytuacji. Na diagramach zarówno powiązanie obiektów, jak i związki pomiędzy klasami wyraża się za pomocą linii łączących odpowiednie skrzynki (rys. 4).

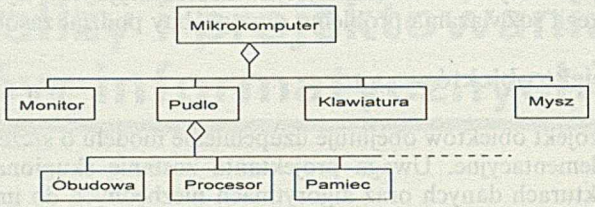


Rys. 4

Takie własności związków, jak: krotność, opcjonalność weszły już – tak mi się wydaje – do folkloru informatycznego, możemy więc tutaj o nich nie mówić. Po prostu cechy te przenoszą się bez większych zmian na teren obiektów. Powszechnie znanym jest fakt, że związki typu „wiele do wielu" generują nową encję. W podejściu obiektowym generują one nowy obiekt, a więc pozostaje tylko kwestia notacji, którą przedstawia rysunek 5. Z rysunku tego możemy odczytać, że wydział musi zatrudniać co najmniej jednego pracownika, a pracownik może pracować w wielu wydziałach. Atrybutami tego powiązania (a zarazem nowego obiektu) są: *data_zatrudnienia* oraz *stawka*. Między pracownikami istnieje związek. Pracownik może mieć co najwyżej jednego szefa, natomiast szefem można być wtedy, jeśli ma się przynajmniej jednego podwładnego.

Rys. 5

Specyficzne cechy związków ujawniają się wtedy, gdy jeden obiekt jest klasą, do której należy obiekt drugi. Rozróżnia się tu dwie kategorie związków: agregację i generalizację. **Agregacja** jest związkiem typu „część całości", i wyraża fakt, że klasę tworzy zbiór obiektów (części). Na diagramach agregację przedstawiamy tak, jak inne związki, ale po stronie złożenia rysujemy mały romb (rys. 6). Agregacja może być wielopozio-

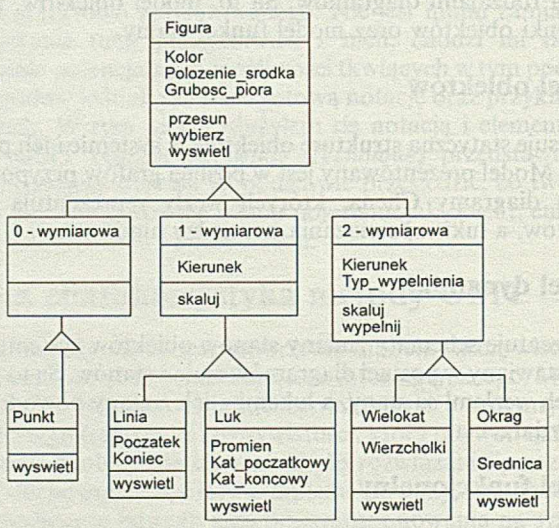


Rys. 6

mowa. Na rysunku 6 przedstawiono przykład agregacji o ustalonej liczbie składowych (linia przerywana oznacza dalsze pojedyncze elementy i pełni rolę wielokropka). Można jednak modelować sytuacje, w których liczba obiektów na niższym poziomie jest zmienną, np. książka składa się z wielu rozdziałów, rozdział z wielu paragrafów, a liczba paragrafów w rozdziale jest zmienna. Można również modelować agregację rekurencyjną, np. struktura konstrukcyjna wyrobów.

Generalizacja jest związkiem między klasą, a jej obiektami (lub jak kto woli – między klasą i jej uszczegółowieniem), ale w tym przypadku klasę wyznaczają cechy podobieństwa obiektów, a nie ich kolekcja.

Rysunek 7 przedstawia typową sytuację, gdzie generalizacja odgrywa pierwszorzędną rolę, pozwalając istotnie uprościć model. Jeśli się przyjrzeć uważnie, to można zauważyć, że między generalizacją a dziedziczeniem istnieje pełna zgodność. To właśnie powoduje, że generalizacja jest bardzo użyteczną konstrukcją, i to nie tylko na poziomie konceptualnym, ale i implementacyjnym. Diagram obiektów łączy klasy i związki między nimi w jedną logiczną całość. Mogą na nim współwystępować zarówno związki między klasami nie należącymi do wspólnej hierarchii, jak i wszelkie związki hierarchiczne. Niektórym z nich można nadać jeszcze inne znaczenia, np. definicja przez przypadki jest formą generalizacji.



Rys. 7

Podsumowując, model obiektów przedstawia wszystkie obiekty systemu (w kolejnych etapach budowy systemu ich przybywa) oraz związki między klasami obiektów. Wszystko to, co można zapisać (przedstawić) na diagramach związków encji jest rejestrowane na modelu obiektów. Ponadto, mając do dyspozycji opakowane wraz z danymi procedury, możemy dokładniej wyrazić semantykę tych związków. Brakuje tu jednak możliwości wyrażenia scenariuszy zdarzeń uruchamiających metody i powodujących zmiany stanów obiektów. Jest to realizowane na diagramach dynamiki i funkcjonalnym.

Środki implementacji

Modele budujemy nie po to, by się cieszyć ich urodą, lecz by przekształcić je w systemy „chodzące" i służące do rozwiązywania konkretnych problemów. Jeśli krok, jaki musimy wykonać by przekształcić model w system jest niewielki i dobrze zdefiniowany, to mamy szansę na szybkie i w pełni kontrolowane osiągnięcie celu. Nowoczesne systemy zarządzania bazami danych pozwalają na stosunkowo łatwą implementację struktur danych, odciążają projektanta od zajmowania się wieloma uciążliwymi problemami z zakresu ich ochrony oraz organizacji wielodostępu. Teoria struktur danych zapewnia nam narzędzia do odwzorowywania jednych struktur w inne. Potrzebna jest teraz możliwość uczynienia tego samego z oprogramowaniem, ale można to uczynić tylko wtedy, gdy już na poziomie modelu konceptualnego dobrze zdefiniujemy akcje wykonywane przez obiekty i ich porządek. Wiele z tych akcji wynika bezpośrednio ze związków między klasami, stąd tak ważnym jest, aby związki te były bardzo precyzyjnie zdefiniowane. Podejście obiektowe pozwala umieszczać je we właściwych miejscach. Wykorzystanie mechanizmu dziedziczenia pozwala na ograniczenie się tylko do jednokrotnej definicji procesu, a polimorfizm umożliwia określenie klasy procesów do wykonania takiej samej metody dla różnych klas obiektów. To bardzo pięknie porządkuje nam obraz świata, który modelujemy i daje możliwość łatwego ustalenia homomorfizmów między światem rzeczywistym i modelem oraz między modelem a światem komputera. Zapewne przyjdzie nam jeszcze trochę poczekać na wdrożenie do praktyki takich baz danych, które będą mogły pamiętać i obsługiwać obiekty, wykorzystywać mechanizmy dziedziczenia oraz dopasowywać do obiektów metody polimorficzne. Ale tak, jak można odwzorować encję w rekord, tak też można przekształcić standardowe klasy metod w gotowe szablony procedur i udostępnić je do użytku w środowisku baz danych. Takie rozwiązanie jest dostępne np. w bazie MAGIC (firmy MSE z Izraela). Wbudowano tam operacje połączeń typu „jeden do jednego" oraz typu „jeden do wielu" z możliwością aktualizacji powiązanych krotek. Interesujące byłyby również mechanizmy pozwalające obsługiwać agregację.

LITERATURA

[1] Dokumentacja pakietu MAGIC. MSE
[2] Materiały konferencji: CASE'91, Warszawa
[3] Materiały konferencji: INFORMATION SYSTEM DEVELOPERS WORKBENCH, Gdańsk 1990, 1992
[4] Pinson L.J., Wiener R.S.: An Introduction to Object-Oriented Programming and Smalltalk. Addison-Wesley Publishing Company, 1988
[5] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W.: Object-Oriented Modeling and Design. Prentice-Hall Inc., 1991
[6] Yourdon E.: Modern Structured Analysis. Prentice-Hall, 1989.

infoMapy inna metoda gromadzenia wiedzy

dokończenie ze s. 9

LITERATURA

[1] Aho A.V., Hopcroft J.E., Ullman J.D.: Projektowanie i analiza algorytmów komputerowych, PWN, Warszawa 1983
[2] Amstel J.J., Poirters J. A.A. M.: The design of Data Structures and Algorithms, Prentice Hall, Academic Service, 1989
[3] Berztiss A.T.: Data Structures. Theory and Practice, Academic Press. New York, London 1971
[4] Dijkstra E.W.: Guarded commands, nondeterminancy and the formal derivation of programs, CACM 18, August 1975, pp. 453–457
[5] InfoFARM: User Manual, GS General Strategies Inc., 1991
[6] Jaworski W.M., Ficocelli L., O'Mara K.S.: The ABL/W4 Methodology for System Modelling, System Research J., 1987, Vol. 4, No. 1, pp. 23–27
[7] Jaworski W.M., Virard M.: Converting a Software Company to a New Technology, Proceedings of the 1986 Canadian Conference on Industrial Computers, Montreal 1986, pp. 121–127
[8] Jaworski W.M., MacCuaig I., Marinelli T., Nysztor T.: Executable Specification for a Large Industrial Process, Proceedings of the 1986 Canadian Conference on Industrial Process, Proceedings of the 1986 Canadian Conference on Industrial Computers, Montreal 1986, pp. 60–65
[9] Jaworski W.M., Cummings T.: Program Normalization and Optimization: Using infoMaps as a inspection and program processing tool, Canadian Conference on Electrical and Computer Engineering, Quebec City, September 1991
[10] Jaworski W.M.: System Analysis and Design in a Classroom: infoMAPs Teaching Factory, Modelling Conference, Pittsburg, Pa., May 2–3, 1990
[11] Jaworski W.M., Deslauries B.: Software Process, Maintenance, Products: Software Development Environment based on infoMAPs, Control and Electrical Eng. Conference, Ottawa, September 1990
[12] Jaworski W.M., Grogono P.D.: infoMAPs: a Pragmatic Environment for Seamless and Nondeterministic Software Development, Concordia University, Computer Science Dept., January 1991
[13] Kovats T.A.: Comments on innovative control constructs in Pancode and EPN, SIGPLAN Notices 25, 11 (November 1990)
[14] Somerville I.: Software Engineering, Addison-Wesley, 1986
[15] Webster D.: Mapping the design representation terrain: a survey, IEEE Computer, (December 1988), pp. 8–23
[16] Wirth N.: Algorytmy+Struktury danych=Programy, WNT, Warszawa 1989
[17] Zaliwski A., Kuraś M.: CASAD – Nowe podejście do modernizacji SI, Materiały Piątej Jubileuszowej Wyższej Międzynarodowej Górskiej Szkoły PTI, Szczyrk 1992
[18] Yau S., Nicholl R., Tsai J. and Liu S.S.: An integrated life cycle for software maintenance, IEEE Trans. Soft. Eng., 1988, 14, 8, 1128–1144.

Drukarka Star SJ-144

„Lepsza jakość druku i większa elastyczność w projektowaniu" – to slogan reklamowy, za pomocą którego firma ABC Data na konferencji prasowej w dniu 5 października br. lansowała nową drukarkę termotransferową Star SJ-144.

Technologia termicznego odwzorowania obrazu jest znana na rynku od dziesięciu lat, jest jedną z najbardziej efektywnych technik drukowania. Dotychczas cena drukarki termotransferowej przekraczała 60 mln zł, co stawiało tę technologię poza zasięgiem masowego nabywcy. Jedną ze specjalizacji Stara jest doskonalenie sprawdzonych technologii w celu udostępnienia ich masowemu odbiorcy. Tak postąpiono z technologią termicznego odwzorowania obrazu, w efekcie czego powstała drukarka Star SJ-144 w cenie ok. 15 mln złotych.

Japoński producent drukarek komputerowych STAR Micronics wprowadzając na rynek nowy produkt odniósł duży sukces na Targach COMDEX w USA, gdzie drukarka ta została uznana najlepszym produktem wystawy, m.in. za wyjątkową jakość kolorów i najlepsze parametry w swojej klasie cenowej.

Drukarka Star SJ-144 spełnia podstawowe wymagania użytkownika:

- drukuje na wszystkim, poczynając od papieru, przez folię, kalkę techniczną, karton – kończąc na etykietach,
- pracując z rozdzielczością 360 × 360 kr./cal gwarantuje wysoką jakość druku,
- daje dobrą jakość druku barwnego – soczyste kolory, a wydruk z niej nie płowieje i nie rozmazuje się,
- rozpoznaje automatycznie emulacje IBM i Epson,
- jest niewielka: 175 × 325 × 356 mm i waży 6,6 kg,
- jest wyposażona w funkcję przeskalowywania wydruku a także druku dwóch zmniejszonych stron na jednej kartce.
- gwarantuje niski poziom hałasu.

JAN RYŻKO
Instytut Maszyn Matematycznych
Warszawa

Projekt nowych standardów

Zestaw znaków w przetwarzaniu i przesyłaniu informacji

Do roku 1987 normy z zakresu przetwarzania danych opracowywane były w ramach ISO (*International Organization for Standardization* – Międzynarodowa Organizacja Normalizacyjna), ściślej – komitetu technicznego TC 97 tej organizacji. W 1987 r. do działalności w tej dziedzinie włączyła się IEC (*International Electrotechnical Commission* – Międzynarodowa Komisja Elektrotechniczna) i powołany został połączony komitet ISO i IEC pod nazwą ISO/IEC *Joint Technical Committee* o symbolu JTC1 o zakresie tematycznym Information Technology (Technika Informatyczna). W ramach tego komitetu działa kilka podkomitetów, wśród których podkomitet SC2 o nazwie *Character Sets & Information Coding* (Zestawy znaków i kodowanie informacji) zajmuje się interesującymi nas zagadnieniami. Są tu utworzone dwie grupy robocze; pierwsza z nich o symbolu WG2 o nazwie *Universal Multiple-Octed Coded Charakter Set* (Uniwersalny, wielobajtowy zestaw kodowanych znaków) zwany krótko USC (*Universal Character Set*), natomiast druga, oznaczona jako WG3, nazywa się *7-Bit and 8*-Bit Codes (Kody 7- i 8-bitowe) i zajmuje się dotychczas stosowanymi kodami.

Pierwszą normą znaków komputerowych był kod znany powszechnie jako ASCII (*American Standard Code for Information Interchange*) przyjęty jako norma ISO-646. Był to kod 7-bitowy określający 128 znaków, z których 32 to znaki funkcyjne, a 94 – graficzne (alfanumeryczne, interpunkcji, nawiasy itp.). W kodzie 8-bitowym, pierwsze 128 znaków odpowiada zestawowi kodu 7-bitowego, a pozostałe 128 zajmują litery narodowe oraz graficzne znaki specjalne.

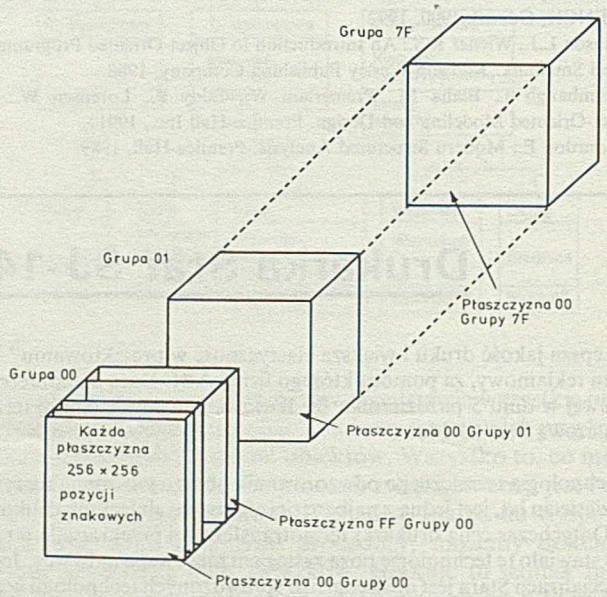
Poszczególne duże firmy komputerowe wprowadzają własne zestawy znaków zwane *code pages* (strony znaków). Najbardziej znanymi takimi stronami są IBM *Latin*-1, oznaczona numerem 850 oraz *Latin*-2 (z literami alfabetów Europy Centralno-Wschodniej) oznaczona jako 852, a także *Microsoft Windows* 1250, z których każda zapewnia kilkadziesiąt użytecznych znaków.

Jednakże jest to wciąż niewystarczające dla kodów „uniwersalnych", w których przewiduje się następujące grupy znaków:
- zestaw około 220 symboli, optymalizowany dla tekstów z dziedziny handlu, organizacji przedsiębiorstw i techniki,
- dodatkowe symbole, stosowane w specjalnych aplikacjach wydawniczych,
- bardzo szeroki zestaw do specjalizowanych zastosowań matematycznych,
- inne znaki specyficzne dla poszczególnych alfabetów.

Poniżej zostaną przedstawione założenia projektu opracowanego przez grupę roboczą WG2, noszącego taką samą nazwę jak nazwa tej grupy, a któremu przyporządkowano numer 10646. Korzystano tu z dokumentu o symbolu ISO/IEC DIS 10646-1.2 z 26 grudnia 1991 roku [1].

Część pierwsza tego projektu jest zatytułowana „Architektura i podstawowa płaszczyzna wielojęzykowa". Zawarte w niej określenia podstawowych pojęć wykorzystywanych w tej normie, określenie ogólnej struktury zestawu kodowanych znaków i sprecyzowanie wspomnianej w tytule podstawowej płaszczyzny wielojęzykowej (*Basic Multilingual Plane* – BMP). Ponadto jest tu podane określenie znaków graficznych używanych w alfabetach światowych, wyszczególnienie reprezentacji kodowych i nazw znaków graficznych BMP, sprecyzowanie 4-bajtowej formy kanonicznej UCS, oznaczonej UCS-4, oraz formy dwubajtowej (UCS-2), a także określenie reprezentacji kodowych dla funkcji sterowania i zarządzania przyszłymi uzupełnieniami do tego zestawu.

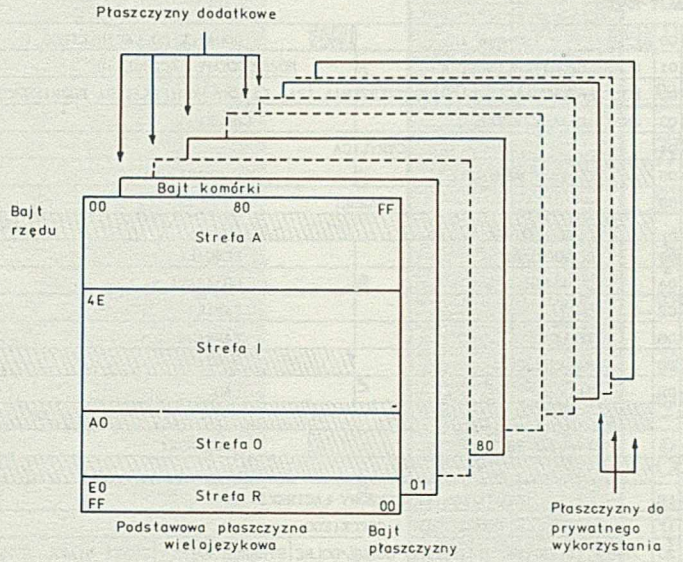
Wymagania zgodności, jakie tu są stawiane, odnoszą się do wymiany informacji oraz urządzeń, jakie tu mogą wystąpić. Pierwszy rodzaj wymagań jest spełniony wówczas, gdy elementy danych w postaci znaków kodowanych spełniają określenia definicji znaku i jego granic, funkcji sterowania oraz położenia w tablicy kodowej. Natomiast drugi rodzaj wymaga opisu urządzenia, które służy do wprowadzania, przesyłania lub odbierania znaków.



Rys. 1. Całe przestrzeń kodowania Uniwersalnego wielobajtowego zestawu kodowanych znaków

Dokumentami związanymi z tym projektem są: ISO 2022, mówiący o 7- i 8-bitowych zestawach znaków kodowanych, oraz ISO/IEC 6429, mówiący o funkcjach sterujących zestawów znaków kodowanych.

Ogólną strukturę omawianego zestawu kodowanych znaków pokazano na rysunkach 1 i 2. Wartość każdego bajtu jest wyrażona w zapisie szesnastkowym (heksadecymalnym) od 00 do *FF*. Forma kanoniczna tego zestawu kodowanych znaków wykorzystuje czterowymiarową przestrzeń kodowania, rozpatrywaną jako całość, składającą się z 128 grup trójwymiarowych. W ten sposób najbardziej znaczący (ósmy) bit bajtu w formie kanonicznej kodowanego znaku może być wykorzystany do celów wewnętrznego przetwarzania w urządzeniu do chwili, gdy jest on ustawiony na zero w ramach odpowiedniego elementu danych kodowanego znaku.



Rys. 2. Grupa 00 Uniwersalnego wielobajtowego zestawu kodowanych znaków

Każda grupa składa się z 256 płaszczyzn dwuwymiarowych, każda płaszczyzna składa się z 256 jednowymiarowych rzędów (wierszy), a każdy rząd zawiera 256 komórek. Znak jest umieszczony i zakodowany w komórce w ramach tej przestrzeni kodującej, lub komórka jest opisana jako niewykorzystana.

W formie kanonicznej do reprezentowania każdego znaku są wykorzystane cztery bajty i one określają odpowiednio: grupę, płaszczyznę, rząd i komórkę. Dzieje się tak dlatego, że dwa bajty nie wystarczałyby do objęcia wszystkich alfabetów światowych, a ponadto 32-bitowa reprezentacja odpowiada architekturze współczesnych procesorów.

Pierwsza płaszczyzna (płaszczyzna 00 grupy 00) jest nazwana podstawową płaszczyzną wielojęzykową (BMP). Zawiera ona znaki ogólnie używane w pismach (ang. *scripts*) alfabetycznych, sylabowych i ideograficznych, łącznie z różnymi symbolami i cyframi. BMP ma strefę ograniczonego wykorzystania, w której znaki mają specjalne właściwości. Kolejne płaszczyzny są traktowane jako dodatkowe i będą zawierać dodatkowe znaki graficzne.

32 płaszczyzny o wartościach bajtu płaszczyzny od *E0* do *FF* grupy 00 są przeznaczone do prywatnego wykorzystania. 32 grupy o wartościach bajtu grupy 60 do *7F* tego zestawu kodowanych znaków są również przeznaczone do prywatnego wykorzystania. Zawartości komórek w strefach prywatnego wykorzystania nie są wyszczególniane w tym projekcie. Każdy znak jest określony w zestawie kodowanych znaków przez bajt grupy, bajt płaszczyzny, bajt rzędu oraz bajt komórki.

Oprócz postaci kanonicznej jest określona również postać dwubajtowa. W ten sposób *Podstawowa płaszczyzna wielojęzykowa* może być traktowana jako dwubajtowy zestaw kodowanych znaków określany jako *USC*-2. Podzespoły przestrzeni kodowania mogą być wykorzystane do określenia podzestawów znaków graficznych.

Jak już wspomniano, *Uniwersalny wielobajtowy zestaw znaków kodowanych* jest traktowany jako całość zawierająca 128 grup, z których każda jest złożona z 256 rzędów znaków, a każdy rząd zawiera 256 komórek.

Na tablicy kodowej, przedstawiającej zawartość płaszczyzny (rys. 2), oś pozioma będzie reprezentować najmniej znaczący bajt o mniejszych wartościach z lewej strony, a oś pionowa – bardziej znaczący bajt o mniejszych wartościach od góry.

Każda oś tej przestrzeni będzie kodowana za pomocą jednego bajtu. W ramach każdego bajtu najbardziej znaczącym będzie bit ósmy, a najmniej znaczącym – pierwszy.

Kodowanie znaków odbywa się w ten sposób, że każdy znak w kanonicznej formie zestawu kodowanych znaków jest reprezentowany przez ciąg czterech bajtów. Najbardziej znaczący bajt, to bajt grupy *G*, a najmniej znaczący – to bajt komórki *K*. Pomiędzy tymi bajtami znajdują się kolejno: bajt płaszczyzny *P* oraz bajt rzędu *R*.

Wartość dowolnego bajtu może być reprezentowana przez dwie cyfry szesnastkowe, np. 31 lub *FE*. Jeśli pojedynczy znak ma być określony przez wartość jego grupy, płaszczyzny, rzędu i komórki, to przyjmuje postać np.

0000 0030 – dla cyfry zero lub
0000 0041 – dla dużej łacińskiej litery *A*.

Kiedy odnosimy się do znaku w płaszczyźnie, to można opuścić pierwsze cztery zera (bajt *C* i bajt *P*) oi wówczas 0030 oznacza cyfrę zero.

Ciąg bajtów reprezentujących znak powinien być przedstawiony w podanej kolejności. Jeśli są one przesyłane szeregowo, bardziej znaczący bajt powinien poprzedzać bajty mniej znaczące. Jeśli nie są one przesyłane szeregowo, to kolejność bajtów powinna być określona odpowiednim porozumieniem między przesyłającym a odbierającym.

Przestrzeń kodująca ma następujące właściwości:

1. Wartości bajtów *P*, *R* i *K*, używanych do reprezentacji znaków graficznych, powinny przyjmować wartości w zakresie od 00 do *FF*. Wartości bajtów *G*, używanych do reprezentowania znaków graficznych, powinny przyjmować wartości w zakresie od 00 do *7F*. Na dowolnej płaszczyźnie wartości kodu *FFFE* i *FFFF* nie powinny być stosowane. Wartość kodu *FFFE* jest zarezerwowana dla „sygnatur", a wartość *FFFF* może być użyta do wewnętrznego przetwarzania wymagającego wartości numerycznej, która nie jest kodem znaku.

2. Wartości kodu, którym nie są przyporządkowane znaki wg omawianej normy (z wyjątkiem znaków przeznaczonych do wykorzystywania prywatnego) powinny być zarezerwowane do przyszłej normalizacji i nie powinny być wykorzystywane do żadnych innych celów. Pozycje kodu dla znaków do wykorzystywania prywatnego nie mogą być przyporządkowane do znaków graficznych w przyszłych wydaniach tej normy.

3. Ten sam znak graficzny nie może być przyporządkowany więcej niż jednej pozycji kodu. Istnieją w zestawie znaków kodowanych znaki graficzne o podobnych kształtach; stosowane są one do różnych celów i mają różne nazwy. Jednakże może się zdarzyć, że znak końcowy może być kombinacją znaku podstawowego i np. określonego akcentu. W tej sytuacji ma on inną reprezentację kodową niż znak podstawowy.

Podstawowa płaszczyzna wielojęzykowa (rys. 2) jest podzielona na cztery strefy:

strefa *A* – pozycje kodowe od 0000 do 4*DFF* (19 903 pozycje),
strefa *I* – pozycje kodowe od 4*E*00 do 9*FFF* (20 992 pozycje),
strefa *O* – pozycje kodowe od *A*000 do *DFFF* (16 384 pozycje),
strefa *R* – pozycje kodowe od *E*000 do *FFFD* (8190 pozycji).

Pozycje kodowe od 0000 do 001*F* oraz od 007*F* do 009*F* zarezerwowane są w BMP dla znaków sterowania. Pozostałe pozycje strefy *A* są wykorzystane do pism alfabetycznych i sylabowych, a także różnych symboli. Strefa I jest wykorzystana do zunifikowania ideogramów wschodnioazjatyckich (chińsko-japońsko-koreańskich – CJK), a strefa *O* jest zarezerwowana dla przyszłych standardów. Natomiast strefa *R* to wspomniana strefa *Ograniczonego wykorzystania*, zawierająca znaki używane prywatnie, postaci podstawiania oraz znaki zgodności. Płaszczyzny od 01 do *DF* w grupie 00 oraz płaszczyzny od 00 do *FF* w grupach od 01 do 5*F* są zarezerwowane dla przyszłej normalizacji i dlatego nie powinny być wykorzystywane do innych celów.

Znaki używane prywatnie w strefie *Ograniczonego wykorzystania* nie podlegają żadnemu ograniczeniu przez tę normę i mogą być użyte do znaków określonych przez użytkownika, co stosuje się powszechnie w pismach ideograficznych. Przy wymianie takich znaków jest konieczne porozumienie się w tej sprawie między nadającym, a odbierającym dane. Te używane prywatnie znaki mogą być stosowane do symboli, które można ponownie określić dynamicznie, ale wymaga to odrębnego porozumienia, gdyż norma ta nie określa technik dla tych znaków.

Postać przedstawienia znaku jest odmiennym kształtem i nie tylko odmianą – nominalnej postaci znaku lub ciągu znaków, jakie są przedstawione w innych strefach graficznych. Przekształcenie od formy nominalnej może obejmować podstawienie, nałożenie lub kombinację. Zasady nakładania, wyboru różnie ukształtowanych znaków lub kombinacja do ligatur albo połączeń – co często jest sprawą złożoną – nie są przedmiotem tej normy. Na ogół postaci przedstawienia nie zastępują nominalnych postaci znaków graficznych, określonych w innym miejscu tego zestawu znaków. Jednakże specjalne zastosowania mogą zakodować te postaci zamiast postaci nominalnych, np. ze względu na zgodność z istniejącymi urządzeniami.

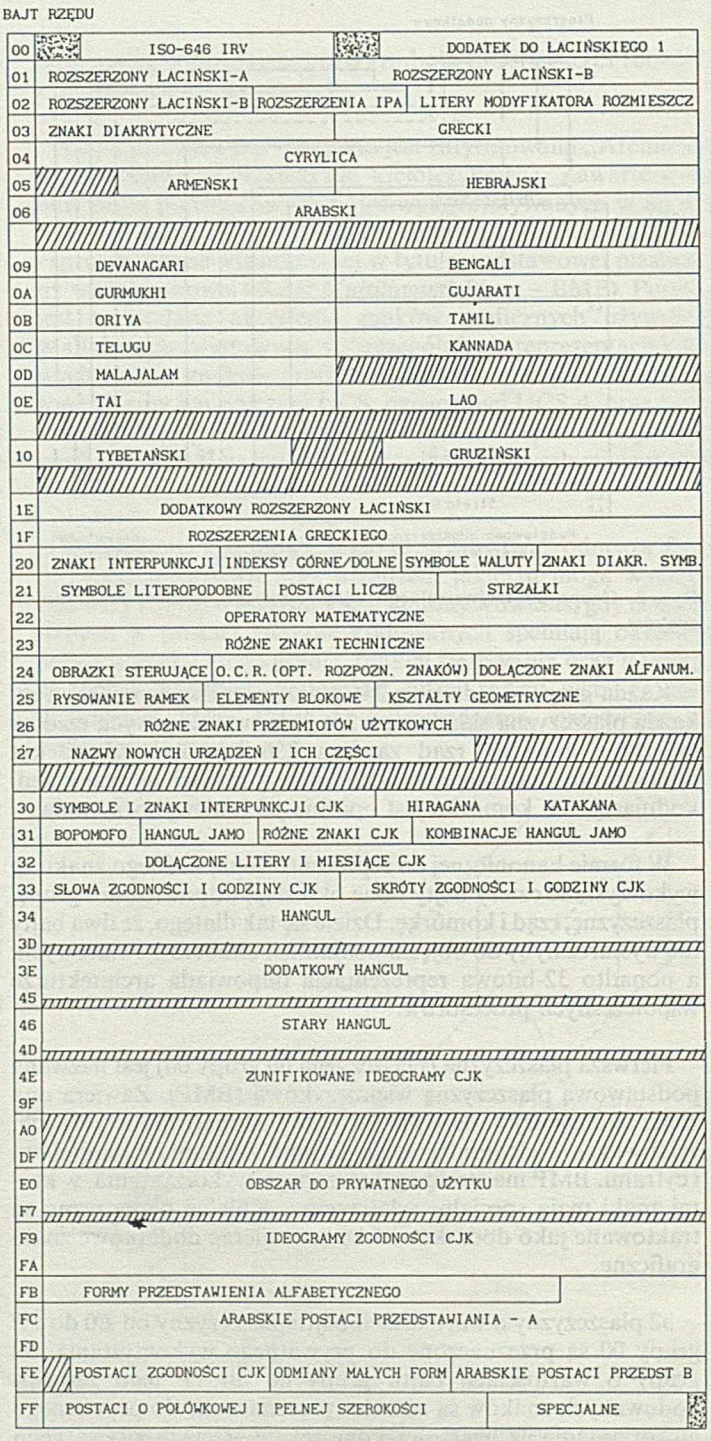
Znaki zgodności są włączone do tej normy głównie ze względu na to, aby pozwolić na dwukierunkową konwersję kodu bez straty informacji.

Rewizji i uaktualniania tego zestawu kodowanych znaków będzie dokonywać podkomitet OSI/IEC JTC1/SC2, przy czym zakłada się, że przyjęte tu nazwy i przyporządkowania nie ulegną zmianie.

Norma przewiduje określenie podzespołów kodowanych znaków graficznych, jakie są wykorzystywane przy wymianie między urządzeniem inicjującym a odbierającym. Rozróżnia się podzespoły: ograniczony oraz wybrany. Przyjęty podzespół może być jednym z nich lub ich kombinacją. Podzespół ograniczony składa się z listy znaków graficznych w określonym podzespole. Pozwala to na stosowanie aplikacji i urządzeń wykorzystujących inne kody do współpracy z tym zestawem kodowanych znaków. Żądanie dostosowania się w przypadku zespołu ograniczonego sprowadza się do podania nazw znaków graficznych w tym podzespole lub ich pozycji kodowych, tak jak to zostało określone w tej normie.

Wybrany podzespół natomiast składa się z listy zbiorów znaków graficznych określonych w normie. Powinien on zawsze zawierać komórki od 20 do 7*E* rzędu 00 płaszczyzny 00 grupy 00. Żądanie dostosowania się do wybranego podzespołu jest spełnione wtedy, gdy są podawane zbiory zgodnie z tą normą.

Norma zapewnia dwie alternatywne formy kodowanej reprezentacji znaków: dwubajtową formę BMP oraz czterobajtową formę kanoniczną. Ponadto norma określa dwa poziomy



Rys. 3. Podstawowa płaszczyzna wielojęzyczna

implementacji. Przy pierwszym z nich element informacji w postaci kodowanego znaku nie powinien zawierać kodowanych reprezentacji znaków kombinowanych. Natomiast na drugim poziomie implementacji taka reprezentacja znaków kombinowanych może być stosowana. Na tym poziomie ten sam znak może mieć dwie lub więcej reprezentacji kodowych.

Zespół znaków sterujących jest taki sam, jak w podanych odnośnych normach i znaki te mają być przedstawione jako ciąg jednego lub więcej bajtów. Natomiast rozszerzenia znaków sterujących wprowadzone w normie ISO-2022 nie są tu wykorzystane.

	000	001	002	003	004	005	006	007
0	000	015	032	0 048	@ 064	P 080	096	p 112
1	001	017	! 033	1 049	A 065	Q 081	a 097	q 113
2	002	018	" 034	2 050	B 066	R 082	b 098	r 114
3	003	019	# 035	3 051	C 067	S 083	c 099	s 115
4	004	020	$ 036	4 052	D 068	T 084	d 100	t 116
5	005	021	% 037	5 053	E 069	U 085	e 101	u 117
6	006	022	& 038	6 054	F 070	V 086	f 102	v 118
7	007	023	' 039	7 055	G 071	W 087	g 103	w 119
8	008	024	(040	8 056	H 072	X 088	h 104	x 120
9	009	025	) 041	9 057	I 073	Y 089	i 105	y 121
A	010	026	* 042	: 058	J 074	Z 090	j 106	z 122
B	011	027	+ 043	; 059	K 075	[091	k 107	{ 123
C	012	028	, 044	< 060	L 076	\ 092	l 108	| 124
D	013	029	- 045	= 061	M 077	] 093	m 109	} 125
E	014	030	. 046	> 062	N 078	^ 094	n 110	~ 126
F	015	031	/ 047	? 063	O 079	_ 095	o 111	127

G = 00
P = 00

Tablica 1.
Rząd 00: ISO-646 IRV

Jeśli wysyłający posługuje się tą normą, to powinna być ona znana również odbierającemu. Droga, którą sposób identyfikacji jest przekazywany odbiorcy, nie jest określona w tej normie. Jednakże niektóre normy dotyczące wymiany kodowanej informacji mogą pozwalać lub wymagać, aby kodowa reprezentacja identyfikacji, odnosząca się do elementu informacji w postaci znaków kodowych, tworzyła część wymienianej informacji.

Rysunek 3 przedstawia *Podstawową płaszczyznę wielojęzykową* wraz z nazwami poszczególnych zespołów znaków i ich rozmieszczeniem w poszczególnych rzędach. Zespoły te identyfikowane według nazw użytych na tym rysunku, są następnie przedstawiane w szczegółowych tablicach, z których trzy pierwsze są przytoczone jako Tablice 1, 2 i 3. Pierwsza z nich, to powtórzenie normy ISO-646 zawierającej nie wymienione tu

	008	009	00A	00B	00C	00D	00E	00F
0	128	144	160	176	À 192	Đ 208	à 224	ð 240
1	129	145	¡ 161	± 177	Á 193	Ñ 209	á 225	ñ 241
2	130	146	¢ 162	178	Â 194	Ò 210	â 226	ò 242
3	131	147	£ 163	179	Ã 195	Ó 211	ã 227	ó 243
4	132	148	¤ 164	180	Ä 196	Ô 212	ä 228	ô 244
5	133	149	¥ 165	µ 181	Å 197	Õ 213	å 229	õ 245
6	134	150	¦ 166	¶ 182	Æ 198	Ö 214	æ 230	ö 246
7	135	151	§ 167	183	Ç 199	× 215	ç 231	÷ 247
8	136	152	¨ 168	¸ 184	È 200	Ø 216	è 232	ø 248
9	137	153	© 169	185	É 201	Ù 217	é 233	ù 249
A	138	154	ª 170	º 186	Ê 202	Ú 218	ê 234	ú 250
B	139	155	« 171	» 187	Ë 203	Û 219	ë 235	û 251
C	140	156	¬ 172	¼ 188	Ì 204	Ü 220	ì 236	ü 252
D	141	157	173	½ 189	Í 205	Ý 221	í 237	ý 253
E	142	158	® 174	¾ 190	Î 206	Þ 222	î 238	þ 254
F	143	159	¯ 175	¿ 191	Ï 207	ß 223	ï 239	ÿ 255

G = 00
P = 00

Tablica 2.
Rząd 00: Dodatek łaciński-1

znaki sterowania (w pierwszych dwóch kolumnach), a także znaki interpunkcji, niektóre symbole matematyczne, cyfry oraz duże i małe litery łacińskie. Druga, nosząca nazwę *Dodatek łaciński* 1, zawiera dwa dalsze rzędy znaków sterujących oraz dwa rzędy kolejnych symboli walut i innych znaków. Natomiast prawa połowa tablicy, to różne odmiany liter alfabetu łacińskiego, w których z polskich znaków narodowych występuje tylko *Ó* na miejscu 211 (0*D*3 w kodzie szesnastkowym oraz *ó* na miejscu 243 (O*F*3 w kodzie szesnastkowym). Trzecia tablica o nazwie *Rozszerzony łaciński – A* zawiera pierwsze 128 znaków rzędu 1 i poza ostatnim znakiem są to rozszerzenia alfabetu łacińskiego. Występują tu wszystkie pozostałe narodowe znaki polskie, które umieszczono w tablicy 4. Przedstawia ona przyporządkowanie poszczególnych liter pozycjom tablicy 3 oraz kodu szesnastkowego.

	010	011	012	013	014	015	016	017
0	Ā 000	Đ 016	G 032	İ 048	l· 064	Õ 080	Š 096	Ũ 112
1	ā 001	đ 017	ġ 033	ı 049	Ł 065	õ 081	š 097	ũ 113
2	Ă 002	Ē 018	Ģ 034	IJ 050	ł 066	Œ 082	Ţ 098	Ų 114
3	ă 003	ē 019	ģ 035	ij 051	Ń 067	œ 083	ţ 099	ų 115
4	Ą 004	Ĕ 020	Ĥ 036	Ĵ 052	ń 068	Ŕ 084	Ť 100	Ŵ 116
5	ą 005	ĕ 021	ĥ 037	ĵ 053	Ņ 069	ŕ 085	ť 101	ŵ 117
6	Ć 006	Ė 022	Ħ 038	Ķ 054	ņ 070	Ŗ 086	Ŧ 102	Ŷ 118
7	ć 007	ė 023	ħ 039	ķ 055	Ň 071	ŗ 087	ŧ 103	ŷ 119
8	Ĉ 008	Ę 024	Ĩ 040	ĸ 056	ň 072	Ř 088	Ũ 104	Ÿ 120
9	ĉ 009	ę 025	ĩ 041	Ĺ 057	ŉ 073	ř 089	ũ 105	Ź 121
A	Ċ 010	Ě 026	Ī 042	í 058	Ŋ 074	Ś 090	Ū 106	ź 122
B	ċ 011	ě 027	ī 043	Ļ 059	ŋ 075	ś 091	ū 107	Ż 123
C	Č 012	Ĝ 028	Ĭ 044	ļ 060	Ō 076	Ŝ 092	Ŭ 108	ż 124
D	č 013	ĝ 029	ĭ 045	Ľ 061	ō 077	ŝ 093	ŭ 109	Ž 125
E	Ď 014	Ğ 030	Į 046	ľ 062	Ő 078	Ş 094	Ů 110	ž 126
F	ď 015	ğ 031	į 047	Ŀ 063	ő 079	ş 095	ů 111	127

G = 00
P = 00

Tablica 3.
Rząd 01:
Rozszerzony łaciński-A

Tablica 4. Polskie znaki narodowe w tablicy „Rozszerzony łaciński – A"

Litera	A	ą	Ć	ć	Ę	ę	Ł	ł	Ń	ń	Ś	ś	Ź	ź	Ż	ż
Pozycja w tablicy 3	004	005	006	007	024	025	065	066	066	067	090	091	121	122	123	124
Kod szesnastkowy	104	105	106	107	118	119	141	142	143	144	15A	15B	179	17A	17B	17C

Poszczególne znaki są określone w normie przez ich graficzny obraz, reprezentację kodową oraz nazwę. Symbole graficzne należy traktować jako typowe obrazowe reprezentacje graficzne tych znaków. Niektóre z tych znaków są nieco powiększone w celu poprawienia ich czytelności. Norma nie opisuje dokładnie kształtu każdego znaku. Na kształt wpływa projekt wykorzystywanej czcionki, co nie wchodzi w zakres tej normy. W większości przypadków znaki graficzne są jednoznacznie identyfikowane przez swe nazwy. Mimo różnych nazw niektóre znaki mają taki sam kształt, np. duża łacińska litera *A*, grecka duża litera alfa oraz duża litera *A* w cyrylicy.

Ciekawostką może być tu fakt, że do opisu polskich znaków narodowych, nawet w angielskim oryginale normy wykorzystuje się słowo *ogonek* przy dużych i małych literach *ą* i *ę*, natomiast dla ukośnych kresek nad *ć, ń, ó, ś* i *ż* – angielskie słowo *acute*, a dla kropki nad *ż dot*.

Zaproponowany projekt normy pokrywa się w znacznej mierze z rozwiązaniem *Unicode Standard*, wersja 1.0 [2], opracowanym przez *Unicode Consortium*, które obecnie włączone jest w niedochodową organizację *Unicode Inc.* utworzoną dla promocji tego rozwiązania. Ostatnio podjęto prace nad ujednoliceniem obu projektów (DIS 10646 oraz *Unicode*).

Problem uniwersalnych znaków stał się na tyle popularny, że od początku 1993 r. zaczął wychodzić w Wielkiej Brytanii dwumiesięcznik pod nazwą *Universe of Character*. Jego redaktorami są: B. McGibbon, który przez wiele lat opracowywał oprogramowanie, zwłaszcza dla systemów wydawniczych, oraz H. McG Ross, znający alfabety wielu krajów świata. Dotychczas (lipiec 1993 r.) ukazały się pierwsze trzy numery tego pisma, których treść może zainteresować polskich Czytelników. W numerze pierwszym redaktorzy wyjaśniają sytuację w związku z istnieniem wspomnianych projektów znaków komputerowych oraz opisują różne ich kategorie podając konkretne przykłady. Numer ten przynosi również sprawozdanie z odbytego na początku grudnia 1992 r. w Niemczech sympozjum, zorganizowanego przez firmę *Unisys* z ramienia konsorcjum *Unicode*. Mówiono tam m.in. o globalnym przetwarzaniu tekstów, architekturze *Unicode*, różnych egzotycznych alfabetach. M. Suignard z *Microsoft France* opowiadał o zawiłościach systemu operacyjnego *Windows NT*, który będzie pierwszym wyrobem wykorzystującym *Unicode* jako podstawową strukturę kodowania. Pismo zawiera też stałe rubryki poświęcone nowym wyrobom (omawiana jest m.in. sprawa opóźnienia pojawienia się na rynku *Windows NT*) oraz komentarzom terminologicznym.

Drugi numer tego pisma, który ukazał się w marcu 1993 r. przynosi propozycje dołączenia nowych pism do proponowanych standardów obejmujących pismo burmańskie, wykorzystujące również te same konwencje kodowania co pisma indyjskie, kmerskie, wykorzystujące również te konwencje, ale o innym rozkładzie w tablicy kodowania, oraz etiopskie, gdzie stosuje się konwencje o podziale sylab na znaki pseudospółgłosek i pseudosamogłosek, które są kodowane. Druga grupa propozycji, określonych jako wstępne, dotyczy pism Sinhala, gdzie wykorzystuje się konwencje kodowania z uprzednich propozycji brytyjskich, z tym, że kodowanie jest inne, pism tybetańskich, wykorzystujące również starsze propozycje brytyjskie, ale nie uwzględniające ostatnich podobnych propozycji, oraz pism mongolskich, gdzie zastosowano nowe, eksperymentalne podejście. Ostatnia grupa propozycji dotyczy dwudziestu pięciu pism obejmujących m.in. języki etruski, mołdawski i staroirlandzki, gdzie występuje więcej wątpliwości niż w poprzednich grupach.

W numerze tym jest też kontynuowane przedstawianie różnych symboli, takich jak spacje, linie poziome, znaki interpunkcji i inne, których liczbę ocenia się na 210 niezbędnych w użyciu. Inny artykuł omawia rolę firmy IBM w ustalaniu standardów znaków komputerowych, m.in. jej udział w konsorcjum *Unicode* i w pracach nad projektem normy ISO 10646.

W trzecim numerze wspomnianego czasopisma przytaczane są doniesienia z Brukseli, gdzie dział tłumaczeń Komisji Wspólnoty Europejskiej przyjął ogólne zasady kodowania znaków komputerowych uwzględniające projekt ISO 10646. Mowa jest też o pierwszej implementacji tego projektu dokonanej w Japonii przez Masami Hasagawę z firmy *DEC*, a stanowiącej dokument o objętości 760 stron formatu A4, w którym 78% zajmują znaki ideograficzne CKJ. Krytycznie omówiona jest wersja beta *Windows NT*, gdzie mimo istnienia 1690 dostępnych znaków brak jest niektórych, często używanych w Europie. Najobszerniejszy materiał tego numeru jest poświęcony akcentom i różnego rodzaju dodatkom stosowanym do liter łacińskich.

Autor pragnie podziękować mgr. inż. Hannie Kuźnickiej z Ośrodka Normalizacji IMM za udostępnienie materiałów i cenne uwagi.

LITERATURA

[1] Information technology, Universal MultipeOcted Coded Character Set (USC) – Part 1. Architecture and Basic Multilingual Pane. Working document for ISO/IEC Draft International Standard 10646-1,2, 26 December 1991

[2] The Unicode Standard – Wordwide Character Encoding. Version 1.0, Volume 1, The Unicode Consortium, Addison-Wesley Publ. Comp., 1991.

WITOLD KOMOROWSKI
Centrum Informatyczne
Politechnika Wrocławska

KRZYSZTOF GODULA
Opole

Analiza stosowania rozkazów mikroprocesora 8086

Przy ocenie listy rozkazów komputera pojawia się czasem kryterium **ortogonalności** oraz **symetrii** [1]. Ortogonalność jest rozumiana jako możliwość stosowania w jednym rozkazie różnych sensownych sposobów adresowania argumentów oraz możliwość stosowania danego rozkazu do różnych sensownych typów danych. Jest to więc postulat niezależności kodu operacyjnego od sposobu adresowania i typu danych; np. w maszynach VAX w większości rozkazów każdy argument może być adresowany na jeden z kilkunastu sposobów – niezależnie od pozostałych argumentów.

Z kolei symetria listy rozkazów wymaga, aby dla każdego rozkazu istniał rozkaz działający „symetrycznie": gdy jest możliwa zmiana typu danych x na typ y, to powinna być też możliwa zmiana y na x; gdy występuje operacja $M \leftarrow M$ **op** R, to powinna też $M \leftarrow R$ **op** M itp. Na przykład, w repertuarze VAX są rozkazy umożliwiające konwersję każdego z siedmiu typów liczb (trzy formaty stałopozycyjne i cztery formaty zmiennoopozycyjne) na dowolny inny – łącznie 40 różnych rozkazów.

Niekiedy do wymienionych kryteriów dodaje się jeszcze ogólność, która zakłada możliwość stosowania w każdym rozkazie każdego typu danych i każdego sposobu adresowania.

Powyższe kryteria są charakterystyczne dla oceny komputerów o architekturze typu CISC (*Complex Instruction Set Computer*), w rozwoju których dominowała tendencja, aby rozkazy języka wewnętrznego pozwalały w możliwie bezpośredni sposób realizować konstrukcje występujące w językach wyższego poziomu. Kryteria te oceniają **potencjalne** możliwości architektury mając na uwadze ewentualną wygodę programisty systemowego, piszącego np. kompilator. Aby ocenić rzeczywistą przydatność poszczególnych rozkazów należy zbadać, jak często są one stosowane w praktyce; wtedy można wnioskować o celowości ich umieszczenia w repertuarze rozkazów danej maszyny. Badania takie są prowadzone na etapie projektowania architektury, a wyniki bywają publikowane np. dla uzasadnienia celowości ograniczenia listy rozkazów w maszynach typu RISC (*Reduced Instruction Set Computer*), czy wyboru między mikroprogramową a emulacyjną implementacją niektórych rozkazów w mikroprocesorach.

Analizator

Do przeprowadzenia tak pojętej **pragmatycznej** oceny listy rozkazów jest potrzebny analizator działających na danej maszynie programów – układowy lub programowy. Analizator taki zbiera statystyki wykonywanych rozkazów, stosowanych typów danych, rodzajów adresowania, wielkości stosu itp. w czasie rzeczywistego wykonywania analizowanego programu.

Poniżej przedstawiono wyniki uzyskane za pomocą analizatora programowego opracowanego na Politechnice Wrocławskiej [2, 7], działającego na komputerach typu IBM PC/AT, dotyczącego listy rozkazów procesora Intel 80286 pracującego w trybie rzeczywistym.

Jako reprezentatywne wybrano do badań kompilatory – są to programy duże, przeznaczone do powszechnego stosowania i pisane przez profesjonalistów, dzięki czemu można przyjąć, że optymalnie wykorzystują możliwości oferowane przez architekturę komputera.

Testowano trzy kompilatory: Turbo Pascal 6.0, Top Speed Modula-2 oraz asembler TASM. Wyniki zestawiono łącznie dla tych trzech programów, podając udział procentowy poszczególnych rozkazów obliczony w stosunku do ogólnej liczby ok. ćwierci miliarda (dokładnie: 240 145 824) wykonanych rozkazów.

Najczęściej wykonywane rozkazy

W tabeli 1 przedstawiono, uszeregowane w kolejności malejącego udziału, 20 najczęściej wykonywanych rozkazów. Pełne zestawienie pokazuje, że spośród dostępnych w procesorze 99 rozkazów różniących się mnemonikami tylko 22 rozkazy mają w badanej próbce udział większy niż 1%, a aż 28 rozkazów ma udział mniejszy niż 0,01%. W grupie 13 rozkazów, które nie występują ani razu są wszystkie rozkazy arytmetyki dziesiętnej (*AAA, AAD, AAS, AAM, DAS* i *DAA*). Pięć pierwszych rozkazów z tab. 1 realizuje połowę wszystkich operacji w badanych programach.

Podobną nierównomierność częstości występowania rozkazów pokazują dane publikowane dla innych komputerów. Np. w IBM 360/85 przy repertuarze ponad 140 rozkazów, zaledwie pięć pierwszych w rankingu daje ponad 52% wykonań [3]. Badania dla kompilatora Fortranu na maszynach VAX [5] wskazują, że próg 50% wykonań jest osiągany przez skumulowany udział dziewięciu rozkazów (przy liście zawierającej około 300 rozkazów).

W tab. 1. znamienna jest różnica w występowaniu rozkazów *PUSH* i *POP*, co wskazuje na częste używanie stosu do przekazywania argumentów w podprogramach.

Wysoka pozycja rozkazu *CMP* (*compare*) wynika z potrzeby przygotowania warunków do ewentualnego rozgałęzienia programu (znaczniki są ustawiane według wyniku uzyskanego z odejmowania argumentów). Podobnie można wytłumaczyć stosunkowo duży udział rozkazu *TEST* (1,3%).

Tabela 1. Dwadzieścia najczęściej wykonywanych rozkazów (określonych przez mnemonik)

Rozkaz (mnemonik)	Udział [%]	Udział skumulowany [%]
MOV	22,0	20
JZ	9,6	30
CMP	8,4	40
MOVS	6,5	
JNZ	5,5	50
INT	5,1	
PUSH	4,4	60
OR	4,4	
XOR	3,7	
POP	3,2	70
STOS	3,0	
JMP	2,8	
ADD	2,1	80
CALL	1,7	
RET	1,7	
DEC	1,7	
LODS	1,5	
SUB	1,4	
INC	1,4	90
TEST	1,3	

Zakłócenia potokowości

Ważną grupą są rozkazy zmieniające sekwencyjność programu (tab. 2). Rozkazy te mają istotny wpływ na zmniejszenie szybkości działania procesora, gdyż wymagają opróżnienia kolejki rozkazów pobranych „na zapas", istniejącej między jednostką sprzęgu z magistralą (*Bus Interface Unit*), a jednostką wykonawczą (*Execution Unit*) [6]. Dla rozkazów skokowych procesor zachowuje się więc tak, jakby mechanizm przetwarzania potokowego (ang. *pipelinning*) był wyłączony.

Tabela 2. Rozkazy sterujące przebieg programu

Rozkaz	Udział [%]
Skoki (wg tab. 3)	20,5
INT	5,1
CALL	1,7
RET	1,7
LOOP	1,1
LOOPNZ	0,1
LOOPZ	0,0
Łączny udział	**30,2**

Duży udział rozkazów skokowych typu *jump, call* oraz *loop* wynika z samej natury programowania komputerów, wymagającej podejmowania decyzji warunkowych (np. w [4] podano, że średni udział rozkazów skokowych wynosi ok. 30%). Wysoka pozycja rozkazu *INT* (*interrupt*) – ponad 5% udziału – wynika ze specyfiki architektury procesora 8086 oraz systemu DOS, gdzie przerwanie programowe jest stosowane do wywołania funkcji systemu operacyjnego.

W obrębie rozkazów skokowych daje się zauważyć wyraźną preferencję stosowania stosunkowo prostych rozkazów warunkowych (tab. 3), co potwierdza ogólną obserwację, że skomplikowane rozkazy są używane rzadko. W analizowanych kompilatorach w ogóle nie występowały skoki badające nadmiar lub parzystość wyniku.

Tabela 3. Rozkład liczby wykonywanych rozkazów skokowych

Rozkaz	Warunek efektywności skoku	Udział [%]	Udział skumulowany [%]
JZ/JE	$ZF = 1$	46,6	40
JNZ/JNE	$ZF = 0$	26,6	70
JMP	–	13,6	80
JC/JNAE/JB	$CF = 1$	3,6	90
JNC/JAE/JNB	$CF = 0$	2,2	
JCXZ	$CX = 0$	2,1	
JNBE/JA	$ZF = 0$ and $CF = 0$	1,4	95
JBE/JNA	$ZF = 1$ or $CF = 1$	1,2	
JGE/JNL	$SF = VF$	1,2	
JNGE/JL	$SF \neq VF$	1,0	
JNG/JLE	$ZF = 1$ or $SF \neq VF$	0,5	100
JG/JNLE	$ZF = 0$ and $SF = V$	0,0	
JS	$SF = 1$	0,0	
JNS	$SF = 0$	0,0	
JO	$VF = 1$	0,0	
JNO	$VF = 0$	0,0	
JP/JPE	$PF = 1$	0,0	
JNP/JPO	$PF = 0$	0,0	

Skoki bliskie i dalekie

Złożony sposób obliczania adresu w procesorze 8086 (segment i przesunięcie) powoduje występowanie kilku form rozkazów *JMP* oraz *CALL*, różniących się m.in. szybkością wykonania. Dotyczy to skoków bezwarunkowych, gdyż skoki warunkowe są zawsze „bliskie" – wewnątrz segmentu i dzięki temu są kodowane w dwóch bajtach.

Tabela 4. Rozkład liczby wykonywanych rozkazów *JMP*

Rozkaz *JMP*	Udział [%]	Udział skumulowany [%]
Wewnątrzsegmentowy krótki	74,1	70
Wewnątrzsegmentowy bezpośredni	16,3	90
Wewnątrzsegmentowy pośredni	4,9	
Międzysegmentowy bezpośredni	4,7	100
Międzysegmentowy pośredni	0,0	

Tabela 5. Rozkład liczby wykonywanych rozkazów *CALL*

Rozkaz *CALL*	Udział [%]	Udział skumulowany [%]
Wewnątrzsegmentowy bezpośredni	75,3	70
Międzysegmentowy bezpośredni	19,5	90
Wewnątrzsegmentowy pośredni	3,1	
Międzysegmentowy pośredni	2,1	100

W tabelach 4 oraz 5 pokazano odpowiednie rozkłady dla skoków różniących się sposobem wskazywania adresu docelowego. I w jednym, i w drugim przypadku dominującą pozycję zajmują formy najkrótsze. „*CALL* międzysegmentowy" ma stosunkowo duży udział ze względu na mocułowość programów, choć i tak 3/4 wywołań procedur odbywa się w obrębie segmentu.

Komunikacja rejestry – pamięć

Interesujące jest zbadanie największej grupy rozkazów oznaczanych tym samym mnemonikiem – *MOV* (*move*) – mającej w ogólnym zestawieniu udział 22%. Analiza tych rozkazów według kodu operacyjnego pozwala uzyskać statystykę przesłań w zależności od źródła i miejsca przeznaczenia.

Tabela 6. Rozkład liczby wykonywanych rozkazów *MOV*

Rozkaz *MOV*	Udział [%]	Udział skumulowany [%]
$r/m \rightarrow r$	34,5	30
$imm \rightarrow r$	29,7	60
$r \rightarrow m$	10,7	70
$m \rightarrow ac$	9,3	80
$r/m \rightarrow sr$	8,7	90
$sr \rightarrow r/m$	3,3	
$ac \rightarrow m$	2,5	
$imm \rightarrow r/m$	1,3	100

W tabeli 6 przyjęto następujące oznaczenia argumentów rozkazu *MOV*: *m* – pamięć, *r* – rejestr, *ac* – akumulator, *sr* – rejestr segmentowy, *imm* – argument bezpośredni, *r/m* – rejestr lub pamięć.

Ponieważ stosowany analizator nie pozwala w przesłaniu $r/m \rightarrow r$ wydzielić przesłań międzyrejestrowych, trudno podać proporcje między rozkazami ładowania rejestru (*load*), a rozkazami zapamiętywania (*store*). Przyjmując za reprezentatywne przesłania do oraz z akumulatora, uzyskuje się stosunek *load* do *store* jak 3,7:1, pozwalający wnioskować o stosowanym modelu obliczeń.

Działania na łańcuchach bajtów

Ostatnią analizowaną tu grupą są rozkazy działające na łańcuchach bajtów (tab. 7).

Tabela 7. Rozkazy działające na łańcuchach bajtów

Rozkaz	Udział w grupie kompilatorów [%]	Udział w grupie programów przykładowych [%]
MOVS	6,5	0,35
STOS	3,0	2,49
LODS	1,5	0,17
SCAS	0,3	0,02
CMPS	0,2	0,01
Łączny udział	**11,5**	**3,04**

Duży udział tych rozkazów w grupie kompilatorów wydaje się wynikać ze specyfiki tego rodzaju programów. Wniosek taki potwierdzają wyniki uzyskane dla kilku przykładowych programów bibliotecznych (ponad 100 mln wykonań w czasie analizy), przytoczone dla porównania w tab. 7.

W grupie kompilatorów większość rozkazów łańcuchowych (ok. 75%) była realizowana w konstrukcji z przedrostkiem repetycyjnym *REP* lub *REPZ*.

Adresowanie argumentów

Niezależnie od rodzaju rozkazów badano sposób adresowania ich argumentów (tab. 8). Wyniki tej analizy wskazują, że niemal połowa obliczeń dotyczy argumentów umieszczonych w szybkich rejestrach procesora, a pozostała część obejmująca ponad 1/3 argumentów jest podawana bezpośrednio w programie. Oznacza to, że argumenty te są pobierane łącznie z rozkazem do kolejki rozkazów i w fazie wykonania nie ma dodatkowej straty czasu na ich przygotowanie. Jeżeli argumenty są pobierane lub zapisywane w pamięci, to stosuje się możliwie najprostsze sposoby adresowania.

Tabela 8. Rozkład sposobów adresowania argumentu

Sposób adresowania	Udział [%]	Udział skumulowany [%]
Rejestrowy	45,3	40
Natychmiastowy	39,2	80
Bezpośredni	8,4	90
Bazowy	3,9	
Indeksowy	2,9	
Bazowo-indeksowy	0,3	100

* * *

Analiza kodu badanych kompilatorów wskazuje, że stosowany jest „rejestrowy" model operacji, tzn. są preferowane działania na rejestrach, w których poprzednio przygotowuje się argumenty, a dopiero końcowe wyniki są odsyłane do pamięci. Świadczy o tym zarówno sposób wskazywania argumentów, jak i duży udział przesłań z wyraźną przewagą rozkazów typu *load* nad *store*.

Tam gdzie to możliwe, są stosowane te formy rozkazów, które wykonują się najszybciej, a argumenty są adresowane w możliwie nieskomplikowany sposób.

Nasuwający się wniosek o nadmiarowości listy rozkazów (ok. 80% rozkazów jest tak rzadko realizowanych, że ich skumulowany udział jest mniejszy niż 10%) wydaje się słuszny, ale w odniesieniu do analizowanej klasy zastosowań. W zastosowaniach specjalizowanych, przydatność rzadko stosowanych rozkazów może być większa. Na przykład, w programach obliczeniowych zapewne wzrośnie udział rozkazów arytmetycznych; dla kompilatorów pierwszy z tych rozkazów – dodawanie (*ADD*) zajmuje 13 miejsce, odejmowanie – 18, mnożenie – 37 a dzielenie dopiero 44 z udziałem 0,06%.

Duży, bo ponad 30% udział rozkazów zakłócających sekwencyjność realizacji programu wskazuje na istotne zmniejszenie efektów potokowości w cyklu rozkazowym mikroprocesora.

Ilościowa analiza czasowa wymagałaby zestawienia odpowiednich rozkładów według czasu zajmowanego na realizację danego typu rozkazu, a nie tylko ze względu na częstość jego występowania.

LITERATURA

[1] Barbacci M.R.: Structural and behavioural description of digital systems. W: New computer architectures (ed. Tiberghien J.). Academic Press Inc., London 1984

[2] Godula K.: Analiza efektywności listy rozkazów procesora i8086. Praca dyplomowa. Politechnika Wrocławska, Wrocław 1992

[3] Hopkins M.E.: A perspective on the 801/Reduced Instruction Set Computer. IBM System Journal, Vol. 26, No 1, 1987

[4] HP Precision Architekture. A New Perspective. Hewlett-Packard Co., 1986

[5] Levy H.M., Eckhouse R.H.: Computer Programming and Architecture: The VAX. Digital Press, 1989

[6] Liu Y.: Microcomputer systems: The 8086/8088 family. Prentice-Hall, 1986

[7] Nikonowicz P.: Analizator programowy mikroprocesora i8086. Praca dyplomowa. Politechnika Wrocławska, Wrocław 1991.

Piąta Jubileuszowa Międzynarodowa Górska Szkoła PTI Szczyrk
21–25.06.1993 r.

WOJCIECH GŁAZEK
Ośrodek Informatyki Śląskiej DOKP

Założenia, projekt i realizacja sieci transmisji danych KOLPAK

Od roku 1989 nastąpiły w Polsce zasadnicze zmiany w zakresie inwestycji telekomunikacyjnych, w tym także dla potrzeb transmisji danych. Dla wszystkich stało się oczywiste, że dalszy rozwój zastosowań informatyki jest związany z rozpowszechnieniem technik transmisji danych. Obecnie wymienić można następujące bardziej znane inwestycje w zakresie transmisji danych będące w fazie realizacji lub projektowania:

- publiczna sieć POLPAK,
- sieć bankowa TELBANK,
- sieć akademicka NASK i sieć EARN oraz sieci poszczególnych środowisk akademickich,
- sieć prywatna banku PKO S.A. – PKONET
- sieć prywatna Polskich Kolei Państwowych KOLPAK.

Polskie Koleje Państwowe (PKP) jako część infrastruktury państwa są modernizowane z kredytów Banku Światowego. Część tych kredytów przeznaczono na realizację inwestycji w dziedzinie informatyki, w tym na budowę prywatnej sieci transmisji danych PKP. W tym zakresie PKP idzie w ślady innych zarządów kolei europejskich, których większość zbudowała własne sieci transmisji danych. Także w krajach byłego RWPG, m.in. na Węgrzech i w byłej Czechosłowacji są budowane podobne sieci. Podstawą budowy takich sieci są własne sieci kablowe zarządów kolei.

Decyzja o budowie sieci PKP została poprzedzona pracami studialnymi firm konsultingowych. Prace te powierzono trzem firmom – po jednej z Kanady, Polski i Wielkiej Brytanii. Przeanalizowały one potrzeby PKP w dziedzinie informatyki, a w konsekwencji również wymagania w dziedzinie transmisji danych. W rezultacie uzasadniono potrzebę budowy własnej sieci transmisji oraz wybrano standard X.25 styku użytkownika z siecią. Sieć tę nazwano KOLPAK.

PKP powołało komisję ds. przygotowania wymagań dla sieci. Pracowała ona ok. 6 miesięcy i przygotowała wymagania techniczne. Po zatwierdzeniu tych wymagań (poszerzonych o część finansowo-prawną) przez Bank Światowy, ogłoszono przetarg na realizację sieci.

Mgr inż. WOJCIECH GŁAZEK w 1970 r. ukończył Wydział Automatyki Politechniki Śląskiej w Gliwicach. W latach 1970–1976 pracował w Zakładzie Systemów Automatyki Kompleksowej PAN w Gliwicach, 1976–1983 w Biurze Projektów Kolejowych w Katowicach, od 1983 na różnych stanowiskach w PKP, obecnie jako główny specjalista Ośrodka Informatyki Śląskiej DOKP Katowice. Kieruje realizacją sieci KOLPAK oraz wraz z zespołem zajmuje się problemami transmisji danych (sieci modemowe, X.25 oraz sprzęt dla budowy sieci X.25, poczta elektroniczna).

Przebieg przetargu i kryteria techniczne wyboru ofert

W przetargu wzięło udział jedenaście firm:

- Alcatel (Francja),
- Ascom (Szwajcaria),
- AT & T (USA),
- Bull (Francja)
- Canac Telecom (Kanada),
- I.T.S. (Niemcy),
- Olivetti (Włochy),
- Schrack Datacom (Austria),
- Siemens AG (Niemcy),
- Siemens Nixdorf (Niemcy),
- US Sprint (USA).

Przedmiotem przetargu były:

- węzły dla sieci,
- centrum zarządzania,
- poczta elektroniczna z oprogramowaniem EDIFACT,
- modermy dla połączeń międzywęzłowych wraz z centrum zarządzania siecią modermową,
- oferta na wykonawstwo sieci, sprzęt pomiarowy i szkolenia.

Komisja przetargowa niestety nie dysponowała konkretnym projektem sieci transmisji. W okresie poprzedzającym przetarg prowadzono w PKP jedynie prace studialne w zakresie zapotrzebowania na terminale i hosty dla różnych, przewidywanych informatycznych systemów aplikacyjnych. Uzyskane dane, a także rozpoznanie podobnych systemów funkcjonujących w Europie Zachodniej pozwoliły określić rząd wielkości zapotrzebowania na poszczególne rodzaje usług w sieci transmisji danych. W PKP funkcjonowały już pewne systemy, np. rezerwacji miejsc w ruchu krajowym i międzynarodowym a niektóre stacje rozrządowe zostały wyposażone w tzw. System Kierowania Pracą Stacji (SKPS), usprawniający tam obieg dokumentów technologicznych. W rezultacie prac studialnych ustalono, że w okresie 2–5 lat należy przewidywać osiągnięcie następującego rozwoju transmisji danych:

- liczba terminali typu IBM PC włączonych do sieci KOLPAK będzie wynosiła ok. 6 tysięcy;
- sieć KOLPAK powinna stać się podstawą do połączenia około czterdziestu instalacji komputerowych (klasy np. podwojonego RISC 6000 firmy IBM) zlokalizowanych na terenie całej sieci PKP, z centralną instalacją komputerową zlokalizowaną w Warszawie;
- PKP będzie modernizować własną sieć central telefonicznych w kierunku budowy central przystosowanych do ISDN (ogłoszono już przetarg na instalację siedmiu takich central);
- sieć analogowych połączeń telefonii nośnej będzie zastępowana siecią kabli światłowodowych; w roku 1992 zakończono budowę kabla światłowodowego relacji Koszalin–Gdańsk–War-

szawa–Kraków–Katowice–Cieszyn; przewiduje się także rozpowszechnienie połączeń w technice PCM;
- poczta elektroniczna powinna zastąpić istniejącą sieć łączności telegraficznej, a na etapie przejściowym powinna istnieć możliwość współpracy obu sieci;
- sieć transmisji powinna być użyta także do przekazywania telefaksów;
- powstająca sieć będzie eksploatowana bez większych zmian (w zakresie stosowanego wyposażenia) co najmniej do końca lat 90.

Na podstawie tych założeń sformułowano wymagania dotyczące aktualnego stanu oraz najbliższej przyszłości, które muszą być spełnione przez wszystkich uczestników przetargu. Wymagania te są następujące:
- Podstawowym rodzajem łączności w połączeniach międzywęzłowych będzie łączność przez kanały telefonii nośnej, a praktycznie osiągalna prędkość transmisji w tych kanałach nie przekroczy 14,4 kb/s dla transmisji asynchronicznej. Nie wyklucza to stopniowego przechodzenia na transmisję w tzw. *frame relay* (2,048 Mb/s), przy czym węzły sieci powinny być przystosowane do bezpośredniej współpracy z takimi kanałami, a producent powinien przedstawić referencje w tym zakresie.
- W węzłach sieci wszystkich rodzajów, w tym także najmniejszych, należy stosować technologię 32-bitową.
- Z punktu widzenia systemu zarządzania siecią niezbędne jest zrealizowanie następujących postulatów:
– stworzenie jednego centrum zarządzania dla wszystkich rodzajów węzłów,
– węzły sieci powinny być widziane z centrum zarządzania jako logicznie pojedyńcze urządzenia, a nie np. jako zespół odrębnych jednostek,
– parametry węzłów powinny być ustawiane zdalnie z centrum.
- Poczta elektroniczna musi mieć możliwość współpracy z siecią telegraficzną, w tym także musi rozpoznawać telegraficzne adresy odbiorców.
- Powinna istnieć możliwość wymiany informacji nadawanych – odbieranych telefaksem z abonentami poczty elektronicznej.
- Konstrukcja węzłów sieci musi przewidywać sprzętową refundację pakietów. Nie określa się minimalnego MTBF dla węzłów i centrum zarządzania siecią.
- Dostawca musi zapewnić możliwość współpracy z innymi sieciami X.25 przez dostarczenie oprogramowania realizującego wymagania standardów X.75 oraz X.121, a także oprogramowania dla taryfikacji.
- Dostawca musi wyposażyć węzły w opcję X.32, tj. dostęp terminala X.25 do węzła przez sieć telefoniczną komutowaną.

Dla porównania warunków finansowych proponowanych przez różnych dostawców, do wymagań technicznych dołączono dane na temat wielkości oraz liczby węzłów przewidywanych w sieci, z zastrzeżeniem, że dane rzeczywiste mogą się różnić od określonych w materiale przetargowym o 25%.

Tabela 1: Grupy węzłów

RODZAJ WĘZŁA (min. liczba portów)	LICZBA WĘZŁÓW	UWAGI
200	8	Istnieje opcja frame-relay Realizuje X.25, X.3/X.28/X.29,X.32,X.75
100	40	Nie przewiduje się frame-relay ani X.75
16	190	j.w.

Wyodrębnione trzy podstawowe grupy węzłów, dla których oferenci mieli określić koszt konfiguracji sieci – przedstawia to tabela 1.

Określono również liczbę modemów oraz urządzeń dla pasma podstawowego (np. base-band), przewidzianych dla realizacji połączeń międzywęzłowych. Założono, że realizując zalecenia V.22/V.22bis, V.32/V.32bis, V.54, V.42/V.42bis MNP5 będą one pracować na łączach trwałych komutowanych oraz w trybie synchronicznym-asynchronicznym. Liczbę modemów przyjętą dla oceny oferty ustalono na 1200 szt., a urządzeń typu base-band – na 100 sztuk.

Tabela 2: Urządzenia zaproponowane przez US Sprint jako węzły sieci

TYP	RODZAJ WĘZŁA (max liczba portów)	LICZBA ZAKUPIONYCH WĘZŁÓW
TP4954	432 porty synchroniczne lub 736 asynchronicznych i 32 synchroniczne	10
TP4944	96 synchron./asynchron.	75
TP8010-16	16 synchron./asynchron.	212
	RAZEM	297

W wyniku pracy komisji przetargowej została wybrana oferta firmy US Sprint, która zaproponowała dla sieci KOLPAK następujące rozwiązania techniczne:
- centrum zarządzania oparte na mikrokomputerze firmy Data General z systemem operacyjnym UNIX;
- jako węzły sieci – urządzenia firmy US Sprint (patrz tabela 2);
- własne rozwiązanie poczty elektronicznej o nazwie Sprintmail (wraz z oprogramowaniem EDIFACT) zrealizowaną na komputerze TANDEM CLX 820;
- jako modermy – urządzenia Penril Datalink 14,4 (DLX V.32 M), a jako urządzenia base-band – urządzenia Racal Comlink 7;
- do kontaktu z siecią telegraficzną – urządzenia firmy Hasler typu Hasler Telex Unit (16 sztuk) przyłączone do węzłów serii 4900;
- wymiana telefaksów między abonentami sieci KOLPAK a siecią telefoniczną będzie możliwa dzięki przyłączeniu do maszyny TANDEM CLX 820 ośmiu zestawów PC/Eicon;
- przygotowanie aplikacji oraz testowanie sprzętu i oprogramowania zapewni odrębna sieć testowa złożona z dwóch węzłów.

Realizacja sieci KOLPAK

Z chwilą wyboru oferenta przez komisję przetargową rozpoczęto pracę nad szczegółową topologią sieci. Prace te trwają nadal w tym sensie, że w niektórych sytuacjach szczegółowa lokalizacja węzłów jest określana przez funkcjonujący w każdej DOKP Zespół Wykonawczy ds. Sieci KOLPAK. Centralnie w Dyrekcji Generalnej PKP działa komisja uzgadniająca te zmiany i koordynująca realizację sieci. Pewne trudności spowodował fakt, że nie określono do końca, jakie systemy aplikacyjne będą w przyszłości wykorzystywały sieć. Z tego względu KOLPAK ma wszystkie cechy sieci publicznej, a nie jest siecią zorientowaną na określony rodzaj użytkownika. Mimo to zasadniczy proces inwestycyjny budowy sieci KOLPAK już rozpoczął się i obecnie został zrealizowany jego pierwszy etap. Cała budowa sieci została podzielona na siedem etapów, a jej koniec przewidziano na grudzień 1994 r.

Należy podkreślić, że przedstawiony tu zestaw urządzeń dostarczonych przez firmę US Sprint nie wystarcza dla uruchomienia abonenta sieci. Aby uzyskać w pełni funkcjonującą sieć transmisji danych, potrzebne jest wszystko to, co będzie służyć dla połączenia procesu aplikacyjnego z najbliższym węzłem sieci. W najprostszej sytuacji będą to dwa modemy oraz dopasowanie oprogramowania aplikacyjnego do współpracy ze zdalnym partnerem przez sieć. Wymaga to dopiero przygotowania.

Wybór dostawcy sieci i takiej usługi, jak np. poczta elektroniczna, w znacznej mierze determinuje zakres wyposażenia

dodatkowego, zapełniającego lukę między procesem aplikacyjnym, a węzłem sieci.

Porównanie sposobu przeprowadzenia budowy sieci pakietowej przez PKP i Deutsche Bundesbahn

Deutsche Bundesbahn (DB) rozpoczęło realizację swojej sieci X.25 (sieci IN) w latach 1986–1988, w zupełnie innych warunkach niż PKP. DB miało wcześniej sieć transmisji opartą na sprzęcie firmy SIEMENS (system TRANSDATA) i działającą na zasadzie hierarchicznej sieci modemowej. Kłopoty, jakie wiązały się z jej stosowaniem spowodowały, że DB zdecydowało się przejść na standard X.25. Finansowanie sieci IN odbywało się w pełni ze środków własnych DB lub budżetu państwa. DB znacznie wydłużyło (w stosunku do PKP) proces przygotowawczy, trwa on ponad rok. Prace przygotowawcze obejmowały szczegółową inwentaryzację potrzeb DB w każdej Dyrekcji Okręgowej (DB ma organizację podobną do PKP).

Podobnie, wcześniej dokonano analizy sieci kablowej w każdej DOKP. Przeprowadzono osobno przetargi na węzły małe i duże oraz modemy. Nadrzędna sieć (ang. *backbone*) dla publicznej sieci pocztowej i kolejowej jest zrealizowana na tym samym sprzęcie firmy SIEMENS AG. Ogólnie przyjęto założenie, że sień IN realizuje usługi stanowiące podzbiór sieci pocztowej DATEX-P i w każdej chwili można zastąpić sieć kolejową przez sieć pocztową. Obecnie sytuacja ta zmieniła się, bo zmieniły się przepisy prawne funkcjonowania poczty w dziedzinie transmisji danych (został zlikwidowany monopol poczty). Sieć kolejowa IN może w przypadku prywatyzacji stać się konkurentem dla DATEX-P. W okresie rozpoczęcia budowy sieci IN podkreślano, że obie sieci są własnością instytucji rządowych i jako takie należą do państwa.

* * *

Realizacja sieci X.25, jak w każdej dużej inwestycji, jest procesem wieloetapowym i skomplikowanym organizacyjnie. Wobec braku w Polsce doświadczenia w stosowaniu i eksploatacji tego rodzaju sieci, zadania stojące przed władzami PKP, a później przed jednostkami wykonawczymi, były i są szczególnie trudne.

Za szczególnie ważne należy uznać następujące wnioski wynikające z dotychczasowego przebiegu zdarzeń:

- Akcja zbierania ofert musi być poprzedzona uruchomieniem własnej, nawet małej, sieci zbudowanej w tym samym standardzie, jak i sieć główna. PKP miało taką sieć, zakupioną dla systemu rezerwacji miejsc w ruchu międzynarodowym.

- Wybór poszczególnych elementów sieci (centrum zarządzania, węzły, modemy, poczta elektroniczna) stwarza ogromne trudności w przypadku braku praktycznego doświadczenia, polegającego na wielomiesięcznym zapoznawaniu się z opisami technicznymi, próbami wdrożenia itd. Opis oferty danej firmy nie może być z oczywistych względów pełny, a z drugiej strony każdy producent ciągle modyfikuje swoje wyroby. W tej sytuacji przeprowadzenie właściwego wyboru jest szczególnie trudne. Przepisy Banku Światowego nie pozwalają na przetarg wieloetapowy, co jeszcze bardziej komplikuje sprawę.

- Administrator sieci transmisji danych musi, dla własnego bezpieczeństwa oraz wygody, testować wszystkie wyroby dopuszczone do stosowania na własnej sieci (sprzęt i oprogramowanie). Do tego celu powinien dysponować wyodrębnioną siecią testową.

Wskazówki dla Autorów

Nadsyłane artykuły nie mogą być publikowane lub przeznaczone do opublikowania w innych czasopismach.

Materiał oprócz tekstu zasadniczego powinien zawierać:

- ★ *krótki życiorys zawodowy Autora i jego zdjęcie,*
- ★ *wykaz literatury,*
- ★ *tabele,*
- ★ *materiał ilustracyjny (rysunki, zdjęcia czarno-białe, wydruki) dołączony do artykułu (nie wklejać materiału w tekst),*
- ★ *podpisy pod ilustracje.*

Na osobnej stronie prosimy podać tytuł naukowy, imię i nazwisko, nazwę zakładu pracy, adres domowy (koniecznie!), numer telefonu oraz informację, jaką drogą przekazać honorarium – kasa Wydawnictwa, poczta, bank.

Tekst artykułu prosimy dostarczyć w formie maszynopisu lub wydruku komputerowego (pisany jednostronnie, z podwójnym odstępem – bardzo ważne! – czyli 30 wierszy na stronie i 60 znaków w wierszu).

Wykaz literatury w porządku alfabetycznym.

Tabele – każda na osobnej stronie – powinny być ponumerowane, opatrzone tytułem oraz ściśle związane z tekstem (zaznaczone miejsce tabeli w tekście).

Rysunki – winny być czytelne (zaznaczyć ich miejsce w tekście), mogą być wykonane ołówkiem.

Zdjęcia – czarno-białe, kontrastowe, na błyszczącym papierze.

Wydruki – czytelne, kontrastowe, wykonane na białym papierze. Format – maksymalnie 18 cm w podstawie.

Każde pierwsze słowo opisu rysunków powinno być pisane dużą literą.

Podpisy pod ilustracje powinny być napisane na oddzielnej stronie, oprócz kolejnego numeru powinny zawierać tytuł (rysunku, zdjęcia, wydruku) i ew. legendę dotyczącą poszczególnych elementów.

Autor opublikowanego w INFORMATYCE artykułu otrzymuje honorarium i bezpłatny egzemplarz okazowy.

Materiałów nie zamówionych redakcja nie zwraca.

WYDAWNICTWA NAUKOWO-TECHNICZNE

Robin Holland: Testowanie i diagnostyka systemów mikrokomputerowych. WNT, Warszawa 1993 r., wyd. 1., s. 196. ISBN 83-204-1554-3

W książce omówiono podstawowe uszkodzenia występujące w mikrokomputerze i urządzeniach z nim współpracujących. Przedstawiono różne urządzenia do testowania mikrokomputera i podano sposoby lokalizacji eliminacji uszkodzeń.

Praca zawiera wiele praktycznych wskazówek przydatnych zarówno przy naprawach komputera, jak i przy pracy z nim.

Książka jest przeznaczona dla inżynierów i techników zajmujących się serwisem, a także dla każdego użytkownika komputera.

Zbigniew Weiss, Tadeusz Gruźlewski: Programowanie współbieżne i rozproszone w przykładach i zadaniach. WNT, Warszawa 1993 r., wyd. 1, s. 390. ISBN 83-204-1637-X

Programowanie współbieżne jest jednym z podstawowych działów informatyki. Jest szeroko stosowane w praktyce informatycznej takich dziedzin, jak systemy operacyjne, systemy zarządzania baz danych, wbudowane systemy czasu rzeczywistego.

W książce podano podstawowe pojęcia i problemy związane z programowaniem współbieżnym i rozproszonym oraz obszernie omówiono podstawowe mechanizmy do synchronizacji i komunikacji, takie jak semafory, monitory, spotkania i przestrzeń krotek. Szczególną uwagę zwrócono na mechanizmy programowania współbieżnego dostępne w systemie Unix, takie jak semafory, komunikaty i kanały oraz zdalne wywołanie procedury. Zasadniczą część stanowi zbiór przykładów i zadań z rozwiązaniami.

Niniejsza książka może być wykorzystana jako podręcznik do przedmiotu „Programowanie współbieżne" na kierunkach informatycznych. Stanowi źródło zadań ćwiczeniowych do przedmiotu „Systemy operacyjne". Książka może być również przydatna dla informatyków-praktyków.

Christopher Van Buren: Zrób to w DOS-ie. WNT, Warszawa 1993 r., wyd. 1, s. 224. ISBN 83-204-1609-4

Christopher Van Buren jest doświadczonym autorem wielu książek o tematyce komputerowej. Specjalizuje się w arkuszach kalkulacyjnych, lecz pisał także o komputerowym składaniu tekstu, grafice, systemach operacyjnych i programach zintegrowanych. W przemyśle komputerowym zaczął działalność jako autor-redaktor książek informatycznych, a następnie został kierownikiem ds. produkcji w firmie z San Diego wytwarzającej oprogramowanie. „Zrób to w DOS-ie" to książka dla każdego, kto bez łamania głowy chce się nauczyć podstaw systemu DOS. W książce tej można znaleźć wyjaśnienie pojęć: katalog, ścieżka dostępu czy zarządzanie plikami. Zawiera ona także proste opisy ważnych poleceń DOS-u i zasady ich stosowania, a także opisy takich operacji, jak: formatowanie i kopiowanie dyskietek, przemianowywanie plików, kopiowanie ich, przenoszenie z jednego miejsca w drugie, archiwowanie itp. Stanowi przegląd wszystkich podstawowych poleceń użytych w tekście, a na początku każdego rozdziału znajduje się krótki słownik podstawowych pojęć.

Christopher Van Buren: Zrób to w Windows wersja 3.1. WNT, Warszawa 1993 r., wyd. 1, s. 294, ISBN 83-204-1610-8

Jest to książka dla każdego, kto bez łamania głowy chce nauczyć się podstaw Windows. W podręczniku tym można znaleźć: opis ważnych funkcji systemu Windows z wyjaśnieniami, jak najlepiej z nich korzystać, informacje o tym jak najlepiej zorganizować pracę zarządcy programów i jak go uruchamiać, pełny opis sposobu korzystania z plików i katalogów za pomocą zarządcy plików, wskazówki na temat dopasowywania i optymalizowania środowiska Windows w sposób gwarantujący efektywność i wydajność. Książka zawiera również opis zarządzania pamięcią komputera i oszczędnego gospodarowania przestrzenią dyskową, opis narzędzi systemu Windows oraz wiele wskazówek i trików, których znajomość spotęguje wiedzę i umiejętność posługiwania się Windows. Na początku każdego rozdziału są zamieszczone krótkie słowniki pojęć.

Dorothy Elizabeth Robling Denning: Kryptografia i ochrona danych. WNT, Warszawa 1993 r., wyd. 2, s. 432. ISBN 83-204-1658-2

W książce omówiono zasady ochrony procesów przetwarzania danych, zagrożenia i bezpieczeństwa przetwarzania danych i możliwości przeciwdziałania im. Podano podstawowe pojęcia z dziedziny kryptografii, ochrony danych, teorii informacji, teorii złożoności i teorii liczb, algorytmy szyfrowania, techniki szyfrowania; omówiono sterowanie dostępu podmiotów do obiektów, sterowanie przepływu informacji i mechanizmy sterowania zapobiegające ujawnianiu poufnych danych.

Książka jest przeznaczona dla osób zajmujących się informatyką oraz dla studentów kierunków informatycznych.

Wojciech Pachelski: Programowanie strukturalne: Fortran 77 dla IBM PC. WNT, Warszawa 1993 r., wyd. 1, s. 288. ISBN 83-204-1494-6

Jest to nowoczesny podręcznik i poradnik programowania w Fortranie 77 w jednej z wersji tego języka – traktowanej jako przykładowa – spośród wielu zrealizowanych dla komputerów IBM PC. Książka zawiera pełny opis języka, zilustrowany przykładami umożliwiającymi nie tylko opanowanie formalnych reguł programowania, lecz również wskazujący, które z tych reguł i jak należy stosować w celu zapewnienia dobrego stylu programowania. Wiele miejsca poświęca się metodyce programowania – książka uczy pisania dobrych programów, a więc programów poprawnych, strukturalnych i efektywnych.

Autor postawił sobie zadanie dużo ambitniejsze niż tylko napisanie podręcznika Fortranu 77. Jego zamiarem było napisanie podręcznika programowania w ogóle i sposobu wyrażania programów w Fortranie 77. W książce można znaleźć wszystkie informacje praktyczne: o samym języku, o jego konkretnej realizacji przez firmę microsoft (MSFOR), łącznie z wiadomościami o instalowaniu translatora, i najniezbędniejsze informacje o systemie operacyjnym DOS. – tak o tym podręczniku pisze prof. dr hab. Władysław M. Turski.

Książka jest przeznaczona dla szerokiego grona czytelników używających maszyn cyfrowych i programowania do rozwiązywania problemów obliczeniowych – naukowych i inżynierskich.

INFORMACJI O PRENUMERACIE udziela Zakład Kolportażu Wydawnictwa Sigma NOT (ul. Bartycka 20, 00-716 Warszawa), skr. poczt. 1004, 00-950 Warszawa, tel. 40-00-21 w. 248, 249, 293, 297, 299 lub 40-30-86 i 40-35-89.

EGZEMPLARZE ARCHIWALNE CZASOPISM można nabyć za gotówkę w Klubie Prasy Technicznej w Warszawie, ul. Mazowiecka 12 (tel. 26-80-16) 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.

Jaworski W.M., Zaliwski A., Kuraś M.: infoMapy – inna metoda gromadzenia wiedzy INFORMATYKA 1993, nr 11, s. 1 Charakterystyka nowego narzędzia do analizy rozwiązań oraz dokumentowania oprogramowania systemu informatycznego.	**Jaworski W.M., Zaliwski A., Kuraś M.: infoMaps – eine andere Methode für Wissenaufspeicherung** INFORMATYKA 1993, Nr 11, S. 1 Eine Charakteristik des neuen Hilfsmittels für Lösungenanalyse und Softwaredokumentieren eines EDV-Systems.	**Jaworski W.M., Zaliwski A., Kuraś M.: infoMaps – an another method for science accumulation** INFORMATYKA 1993, No 11, p. 1 Characteristics of a new tool for solution analysis and for documenting of data processing system software.
Benedykt Z.: Co daje obiektowe podejście do analizy i projektowania systemów informatycznych? INFORMATYKA 1993, nr 11, s. 10 Omówienie możliwości oraz korzyści wynikających z obiektowego podejścia do analizy i projektowania systemów informatycznych na przykładzie metody OMT (Object Modelling Technique).	**Benedykt Z.: Was bietet eine objektorientierte Einstellung zur Analyse und Projektierung von EDV-Systemen?** INFORMATYKA 1993, Nr 11, S. 10 Eine Besprechung von Möglichkeiten und Vorteilen, die aus objektorientierter Einstellung zur Analyse und Projektierung von EDV-Systemen erfolgen, illustriert mit einem Beispiel der OMT (Object Modelling Technique)-Methode.	**Benedykt Z.: What offers the object-oriented approach to analysis and design of data processing systems?** INFORMATYKA 1993, No 11, p. 10 Discussion of possibilities and advantages which results from object-oriented approach to analysis and design of data processing systems on example of the OMT (Object Modelling Technique) method.
Ryżko J.: Projekt nowych standardów. Zestaw znaków w przetwarzaniu i przesyłaniu informacji INFORMATYKA 1993, nr 11, s. 14 Szczegółowe omówienie treści projektu normy ISO 10646 oraz nowego specjalistycznego czasopisma „Universe of Character".	**Ryżko J.: Entwurf von neuen Normen. Zeichenvorrat in Informationsverarbeitung und -übermittlung** INFORMATYKA 1993, Nr 11, S. 14 Eine ausführiche Besprechung von Inhalt des ISO 10646-Normenentwurfes sowie des neuen engspezialisierten Fachzeitschrift „Universe of Character".	**Ryżko J.: New standards proposal. Character set for information processing and transmitting** INFORMATYKA 1993, No 11, p. 14 Detailed discussion of the ISO 10646 standard proposal as well as of the new specialized magazine „Universe of Character".
Komorowski W., Godula K.: Analiza stosowania rozkazów mikroprocesora 8086 INFORMATYKA 1993, nr 11, s. 21 Omówienie wyników testowania trzech kompilatorów pod kątem częstotliwości wykonywania rozkazów mikroprocesora 8086 za pomocą analizatora programowego na komputerze typu IBM PC/AT.	**Komorowski W., Godula K.: Analyse von Anwendung der 8086-Mikroprozessorbefehle** INFORMATYKA 1993, Nr 11, S. 21 Eine Besprechung von Testungsergebnissen der drei Kompilatoren unter Gesichtspunkt der Ausführungshäufigkeit von 8086-Mikroprozessorbefehlen mit Anwendung eines Softwareanalysators auf einem Computer der IBM PC/AT-Klasse.	**Komorowski W., Godula K.: Analysis of the 8086 microprocessor instructions using** INFORMATYKA 1993, No 11, p. 21 Discussion of results of three compilers testing directed at execution frequency of 8086 instructions by means of software analyser on computer of the IBM PC/AT class.
Głazek W.: Założenia, projekt i realizacja sieci transmisji danych KOLPAK INFORMATYKA 1993, nr 11, s. 24 Charakterystyka rozwiązań oraz stan realizacji sieci transmisji danych dla Polskich Kolei Państwowych.	**Głazek W.: Voraussetzungen, Projekt und Realisierung des KOLPAK-Datenübertragungsnetzes** INFORMATYKA 1993, Nr 11, s. 24 Eine Charakteristik von Lösungen und Realisierungsstand des Datenübertragungsnetzes für die Polnischen Staatseisenbahnen.	**Głazek W.: Assumptions, design and realization of the KOLPAK data transmission network** INFORMATYKA 1993, No 11, p. 24 Characteristics of solutions and of realization stage of the data transmission network for Polish State Railways.

POLSKIE LITERY + NOWE MOŻLIWOŚCI

AKREM (90-950 Łódź 1, box 308, tel. 36-48-74) oferuje pakiety programowe:

POLY do CLIPPERA 87/5

- indeksowanie wg polskiego alfabetu w dowolnym standardzie,
- wprowadzanie, kontrola i manipulacje na danych o polskojęzycznym obrazie (GET-y, funkcje UPPER, ISALPHA itd.),
- praktycznie wszystkie znane metody programowego definiowania i symulacji polskich liter:
  - symulacja na Herkulesie w trybie znakowym i graficznym,
  - nie znikające polskie litery na EGA, VGA, SVGA we wszystkich trybach (w tym tryby tekstowe 25/28/43/50 wierszy, tryby graficzne, 512 fontów w trybie tekstowym),
  - polskie klawiatury (Alt i/lub dowolnie wybrany "klawisz polskiej litery", możliwość zmiany sposobu wprowadzania "w locie", liczne dodatkowe udogodnienia, jak np. automatyczna zmiana wielkości już wprowadzonych liter, "inteligentny "CapsLock, itd.),
  - na drukarce trzy metody , w tym nowość: nie znikający "download" (automatycznie samodefiniujący się po każdym włączeniu drukarki, działa bez żadnych przeróbek na każdej drukarce emulującej standard Epson lub IBM) oraz druk w trybie graficznym i najprostsza metoda "nakładania",
- dynamiczne dostosowanie programu do dowolnego (nawet własnego) standardu polskich liter,
- komunikacja programów w CLIPPERZE z rezydentami pakietu,
- bezbłędna współpraca z dowolnymi wersjami DOS-u,
- a ponadto system tablica/dysk, tablice wielowymiarowe o zwiększonej pojemności, szybkie operacje na tablicach, system pakowania danych (40-50% oszczędności pamięci dyskowej bez straty szybkości i miejsca w pamięci operacyjnej), opcja szyfrowania danych (przydatna dla użytkowników pracujących w sieci), menu wielopoziomowe, dwuwymiarowe i logiczne (wybór wielu opcji jednocześnie), manipulowanie atrybutami i kolorami ekranu, wyświetlanie dużych tekstów w okienkach, uniwersalne podejście okienkowe, polski kalendarz i liczby słownie, możliwość indeksowania i wyszukiwania danych wg polskiej wymowy, opcja definiowania własnych szablonów formatu, "trójwymiarowe" rozszerzenie funkcji DBEDIT, możliwość pracy z ekranami o zwiększonej liczbie wierszy i kolumn, klawisze F11 i F12 itd.

W skład pakietu wchodzą: biblioteka funkcji, cztery programy rezydentne (nakładki na DOS), edytor własnych fontów dla Herkulesa, VGA i drukarki (Draft i NLQ), programy pomocnicze, pliki rozmaitych fontów, programy przykładowe oraz doskonała dokumentacja (podręcznik oczywiście po polsku), a w przypadku CLIPPERA 5.x dodatkowo plik QUICKPOL umożliwiający polonizację bez zmian w programie źródłowym oraz plik NEWORDER definiujący strukturalno-obiektowy język NewOrder (włączający do CLIPPERA m.in. obiekty klasy MENU).

POLY-C

Zawiera wymienione wyżej programy rezydentne i narzędziowe, podręcznik napisany specjalnie dla C oraz bibliotekę blisko 300 funkcji dostarczanych w postaci źródłowej, przystosowanych do użycia wraz z dowolnym kompilatorem języka C na IBM PC (standard ANSI lub C++, szablony dla Borlanda 3.1), a także możliwych do wykorzystania przez znających C programistów w CLIPPERZE, FOX PRO czy też w PARADOX-ie.

Biblioteka funkcji POLY-C zawiera większość funkcji z biblioteki POLY dla CLIPPERA, a ponadto szereg funkcji napisanych specjalnie dla C. W bibliotece została uwzględniona kompleksowa (ekran, klawiatura, drukarka, manipulacje znakowe, konwersja) wzorowo zorganizowana polonizacja programu, a ponadto takie zagadnienia, jak: dowolne operacje na datach i łańcuchach znaków, pakowanie i szyfrowanie danych, pełnoekranowe wprowadzanie danych (obiekty GET), wyświetlanie dużych tekstów w okienkach, uniwersalne podejście okienkowe, funkcja INTPOLY symulująca operacje BIOS-u w pamięci ekranu, manipulacje ekranowe, drukowanie danych, organizacja menu wielopoziomowych, dwuwymiarowych i logicznych (obiekty MENU i PROMPT, menu strumieniowe), przetwarzanie list, help kontekstowy, język NewOrder, komunikacja programu w C z rezydentami pakietu, itd.

NEW INDEX

Nowy system obsługi baz danych w C. Zawiera obsługę obiektów FIELD, zbiorów DBF i DBT z możliwością pakowania pól bez straty szybkości pracy oraz zbiorów NDX, NTX i NZX (pliki indeksowe najnowszej generacji zaprojektowane specjalnie dla C, dowolnie wiele indeksów różnych rodzajów w jednym zbiorze, możliwość programowego ustawienia wielkości strony indeksowej). Udogodnienia do pracy w sieci. Generator Pól. Przezroczysta baza danych. Podręcznik.

POLY-DOS

Pakiet POLY-DOS umożliwia wszechstronną polonizację (tryb tekstowy i graficzny) oraz m.in. przedefiniowanie klawiatury (makrodefinicje dowolnej długości), definiowanie własnych fontów na ekranie i drukarce, komunikacja rezydentów z programami napisanymi w dowolnym języku, itd. Podręcznik.

EURO-SOFTWARE

Do POLY5 i POLY87 można dołączyć również moduły EXPORT umożliwiające przystosowanie programu do dowolnego alfabetu europejskiego (indeksowanie, wprowadzanie danych, duże i małe litery). Zaleca się zakup EXPORT-u łącznie z pełnym POLY-DOS-em, ponieważ można wtedy całkowicie przedefiniować klawiaturę, ekran i drukarkę.

BALL STAR - MECZ

Piłkarska baza danych na IBM PC zawierająca dowolne tabele i prognozy. Podręcznik.

* * *

A oto ceny pakietów (z podatkiem, ale bez opłaty pocztowej).

POLY 87 (do Clippera 87)	- 1.200 tys. zł
POLY 5 (do Clippera 5.0, 5.01 i 5.2)	- 1.400 tys. zł
POLY 5 + 87	- 1.800 tys. zł
EXPORT 5 lub 87	- 200 tys. zł
EXPORT 5 + 87	- 300 tys. zł
POLY-C (źródła)	- 2.200 tys. zł
NEW INDEX (źródła)	- 2.000 tys. zł
NEW INDEX + POLY C (źródła)	- 3.200 tys. zł
POLY-DOS	- 700 tys. zł
MECZ	- 550 tys. zł

Przy jednoczesnym zakupie POLY-C i do CLIPPERA dodatkowe 500 tys. zł zniżki. Zamówienia przyjmowane są listownie lub telefonicznie. Wysyłka za pobraniem.

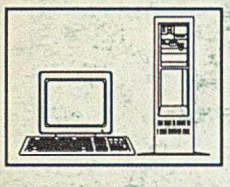
Oferujemy również CA-CLIPPER 5.2, CA-dBFast 2.0 (baza danych z kompilatorem pod Windows) i inne oprogramowanie firmy Computer Associates. W tym przypadku obowiązuje zamówienie pisemne wraz z kserokopią dowodu wpłaty na konto

AKREM, PBK SA V/O ŁÓDŹ, nr 374606-11729-136.

0/1/93

JUNISOFTEX®

Przedsiębiorstwo Produkcyjno-Usługowe Innowacji Technicznych Biuro Techniczno-Handlowe 44-100 Gliwice ul. Konstytucj 11.
tel. 31-57-00, 31-51-31, 31-75-10, fax 31-75-10, centr. 31-90-81 do 88, telex 036233
Zespół wdrożeń i analiz systemów komputerowych wewn. 15 ,25, 282, Zespół inżynierii oprogramowania wewn. 14, 20, 245,
Zespół d/s serwisu systemów Novell i sprzętu komputerowego wewn. 16, 245, Dział Handlowy wewn. 18, 28, 201,
Szkoła Komputerowa wewn. 282, Sekretariat wewn. 250, Prezes wewn. 13, 272, Dyrektor wewn. 12, 272, Główny Księgowy wewn. 256,
* Konto: PKO I/O Gliwice 27661-225298-136 *



OFERUJEMY

Kompleksową komputeryzację przedsiębiorstw obejmująca dostawę i wdrożenie zintegrowanych systemów komputerowych takich jak system:

finansowo-kosztowy (automatyczne rozliczanie kosztów)
+
gospodarka materiałowa
+
środki trwałe
+
wyposażenie
+
obrót towarowy fakturowanie sprzedaży
+
kadry-płace
+
techniczne przygotowanie produkcji

Sprzęt komputerowy w tym:

Komputery
Monitory
Dyski twarde
Drukarki (mozaikowe i laserowe)
osprzęt sieciowy (karty sieciowe, kable zwykłe i światłowodowe)

Systemy i oprogramowanie licencjonowane

DOS
Novell
Btrieve
SQL Base
Windows

Szkolenia komputerowe

podstawy obsługi komputerów
arkusze kalkulacyjne (Qpro, Lotus, EXCEL)
edytory tekstów (WordPerfect, TAG, ChiWriter)
bazy danych (dbase IV, Clipper, Paradox)
systemy operacyjne (DOS, Novell)

SOFTWARE SUPPORT FOR DOS, NOVELL 2.2, 3.11, 4.0, SQL BASE firmy GUPTA

*

OPROGRAMOWANIE DLA PRZEDSIĘBIORSTW

oferujemy własnej produkcji zakładowy system informatyczny PHU-Perfect(N) obejmujący swoim zakresem:
system finansowo-kosztowy umożliwiający automatyczne rozliczanie kosztów, atomatyczną dekretację, analizę rozrachunków
system obrotu materiałami, wyrobami, towarami (kody paskowe) w tym: stany magazynowe, ewidencje zamówień, fakturowanie sprzedaży (VAT) z automatyczną dekretacją rozdzielników kosztów i sprzedaży do systemu finansowo-kosztowego,
system ewidencji majątku trwałego i wyposażenia
system kadrowo-płacowy (SQL) z automatyczną dekretacją rozdzielników płacowych do systemu finansowo-kosztowego
system technicznego przygotowania produkcji (uk. 1993 r)
(oprogramowanie w systemie SQL BASE w przygotowaniu)

*

OPROGRAMOWANIE DLA URZĘDÓW MIAST I GMIN

oferujemy własnej produkcji system finansowo-podatkowy UMG-Perfect(N) obejmujący swoim zakresem:
system finansowo-księgowy,
systemy podatkowe obejmujące: ustalanie wymiarów, naliczanie i kontrolę płatności należności podatkowych dla dowolnych podatników i/lub płatników podatków na ich kontach i kontach dochodów ze swobodną korektą dowolnych informacji w dowolnym okresie a w tym:
ewidencja zapłat z wykorzystaniem kodów paskowych, emisja wezwań do zapłaty, naliczanie odsetek.
system ewidencji majątku

*

Zapraszamy do współpracy w zakresie wdrożeń naszych systemów partnerów, - firmy informatyczne z całego kraju !!!

0/7/92