8-9 1979

informatyka

WYDAWNICTWO NOT SIGMA

W NUMERZE

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA

SIGMA

CZASOPISM I KSIĄŻEK TECHNICZNYCH

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

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pokój 331, tel. 27-71-40 lub centrala 26-82-61 w. 285, dyżury redakcji 10.00—13.00

Zakł. Graf. „Tamka". Zam. 308. Papier druk. sat. V kl. 70 g. A1. Obj. 7,5 ark. druk. Nakład 7350. C-112.

Cena egzemplarza zł 50.— INDEKS 36124 Prenumerata roczna zł 300.—

Informatyka

zastosowania w gospodarce, technice i nauce

Nr 8-9
MIESIĘCZNIK
1979
ROK XIV
sierpień-wrzesień

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

JAN GOLIŃSKI
Zjednoczenie Informatyki
Warszawa

Aktualne problemy oprogramowania

Od chwili stworzenia maszyn liczących dających się swobodnie programować, rozmyśla się nad sposobami ułatwienia programowania. Potrzeba wyższej wydajności programistów, większej pewności i niezawodności programów doprowadziła w okresie ostatniego dziesięciolecia do powstania rozlicznych pomocy dla programistów w postaci systemów i technik programowania. Programowanie normowane i strukturalne, tablice decyzyjne, systemy oparte na schematach blokowych i „optymizery", techniki generacyjne i programowanie konwersacyjne, wszystko to stanowi próbę udoskonalenia drogiego i pracochłonnego programowania metodami tradycyjnymi.

Oprogramowanie jest tą niematerialną częścią systemu elektronicznego przetwarzania danych, którą najchętniej przyrównuje się do przysłowiowej beczki bez dna; o ile bowiem w sferze sprzętu wzrost wydajności i obniżka kosztów idą zgodnie naprzód, o tyle produkcja oprogramowania z roku na rok regularnie drożeje.
Udział kosztów programowania w ogólnych kosztach działalności ośrodka obliczeniowego bardzo często przekracza już 50% i wykazuje w dalszym ciągu tendencję do dalszego wzrostu.

CZYM JEST OPROGRAMOWANIE?

Pod nazwą „oprogramowanie" *(ang. software)* rozumie się to, co stanowi niematerialną część systemu komputerowego, a więc również różne „środki pomocnicze" i „narzędzia" oraz gotowe programy (systemowe i użytkowe). Programowanie ma za sobą już kilka stadiów rozwojowych, począwszy od programowania „maszynowo-zorientowanego" tzn. programowania w kodzie wewnętrznym komputera, poprzez programowanie symboliczne, aż do dominującego obecnie programowania „problemowo-zorientowanego".

To ostatnie stadium charakteryzuje się opracowaniem i szybkim rozwojem języków programowania zorientowanych problemowo tzn. pozwalających formułować przebiegi komputerowe w sposób zwięzły i przejrzysty w zakresie pewnych określonych typów zagadnień i zadań. Językami takimi są m.in.: COBOL, PL/1, FORTRAN, BASIC. Trzeba powiedzieć, że przejście od programowania zorientowanego maszynowo do programowania zorientowanego problemowo przyniosło znaczny wzrost wydajności programowania i ogólnie — produktywności komputera.

ZNACZENIE POSZCZEGÓLNYCH JĘZYKÓW PROGRAMOWANIA

Nie ma, jak dotąd, powszechnie słusznych jednoznacznych ocen na temat znaczenia i rozprzestrzenienia się poszczególnych języków programowania. Niemniej jednak dysponujemy wynikami sondażu opinii w kwestiach fragmentarycznych — między innymi także w odniesieniu do stopnia rozpowszechniania określonych języków.

I tak w roku 1971 tygodnik „Wirtschaftswoche" ogłosił rezultaty badań, według których udział poszczególnych języków w ogólnej masie zastosowań przedstawiał się następująco: ASSEMBLER 44%, COBOL 27%, RPG 11%, FORTRAN 10%, PL/1 6% i ALGOL 2% (razem 100%). Nowsze badania tego zagadnienia, przeprowadzone w 1974 r. wśród użytkowników sprzętu komputerowego firmy SIEMENS dały podobny obraz: ASSEMBLER 45%, COBOL 21%, RPG 17,2%, FORTRAN i PL/1 po 6,5% i ALGOL 3,8% (razem 100%).

Również z 1974 roku pochodzą wyniki badań przeprowadzonych przez IBM u użytkowników systemów operacyjnych OS i DOS. Według nich 47% użytkowników wykorzystujących DOS posługuje się jeszcze językiem ASSEMBLER, a 33,6% chce się nadal nim posługiwać w nowych opracowaniach rozwojowych, podczas gdy tylko 25,8% ukierunkowuje się na COBOL. Jeżeli zaś chodzi o użytkowników korzystających z systemu OS, to w 40% stosują oni COBOL a tylko w 22% preferują ASSEMBLER. Oprócz tych języków wśród stosujących OS tylko RPG i PL/1 mają większe znaczenie.

Tak pojmowane i mierzone „znaczenia" języków programowania mogły od tamtych czasów do chwili obecnej ulec istotnym zmianom. COBOL na przykład odgrywa z pewnością dziś dużo większą rolę niż 5 lat temu, zważywszy, że problemy dysponowania odpowiednio dużą pojemnością pamięci operacyjnej mają obecnie już znaczenie drugorzędne. Poza tym we wspomnianych badaniach nie występuje jeszcze konwersacyjny język BASIC, o którym wiadomo, że w ostatnich latach bardzo poważnie zyskał na znaczeniu.

WYDAJNOŚĆ PRACY PROGRAMISTÓW

Projekty oprogramowania mają to do siebie, że nie dają się łatwo oceniać pod względem faktycznej pracochłonności i stopnia trudności opracowania. Z tego powodu nierzadko dochodzi do fałszywej — ilościowo i jakościowo — obsady kadrowej procesu opracowania, co z kolei pociąga za sobą konieczność przesuwania terminów, a nawet prowadzi do rezygnacji z realizacji zadania. Również proces planowania w przypadku realizacji większego systemu oprogramowania problemowego jest nadzwyczaj uciążliwy ze względu na fakt występowania olbrzymiej liczby wzajemnie uwarunkowanych i przeplatających się czynności, którym z reguły towarzyszą wielkości niepewne, a często nawet całkowicie nieznane.

A taką właśnie wielkością jest wydajność pracy programistów. Wyraża się ją najczęściej liczbą opracowanych przez 1 programistę instrukcji w jednostce czasu. W zamieszczonej tabeli 1 zawarte są średnie wartości wydajności programistów, zestawione na podstawie badań przeprowadzonych przez wspomnianą już firmę SIEMENS. Wykorzystano w nich dane statystyczne dotyczące różnych zespołów programistów realizujących ważniejsze projekty oprogramowania firmowego. Przytoczone w tabeli wydajności programistów uwzględniają wyłącznie czynności z zakresu „czystego" programowania, do których zaliczono: rozpoznanie problemu, zdefiniowanie stanu istniejącego, analizę problemu, zaprojektowanie i wykonanie rozwiązania, sporządzenie dokumentacji, a także testów dla weryfikacji wykonanych programów.

Średnia wydajność programistów (wg badań firmy SIEMENS)

Stopień trudności projektu	Średni czas realizacji projektu w miesiącach		
	6—12	12—24	więcej niż 24
	liczba opracowanych instrukcji na:		
	osobodzień	osobo-miesiąc	osoborok
Niski (mało powiązań programowych, prosta logika)	20	500	10 000
Średni	10	250	5 000
Wysoki (dużo powiązań i uwarunkowań programowych, skomplikowana logika)	5	125	1 500

Wydajność programisty jest oczywiście bardzo ściśle uzależniona od stopnia trudności zadania. Równocześnie jednak, przy tym samym stopniu trudności, występują niekiedy istotne wahania w wydajności pracy, uzależnione od samego czasu trwania projektowania. W miarę bowiem postępu prac zachodzi również proces uczenia się wykonawców, co ma wyraźny wpływ na osiąganie przez nich większych wydajności. Daje się to zauważyć zwłaszcza w przypadku opracowywania programów łatwych i średnio trudnych, przy realizacji których zatrudnia się najczęściej osoby jeszcze bez większego doświadczenia, w stosunku do których praktyka daje znacznie szybsze efekty. Natomiast w przypadku opracowywania programów trudnych i złożonych, realizację których powierza się od razu osobom o wysokich kwalifikacjach oraz z dużą praktyką, efekt przyrostu wydajności w miarę postępu prac projektowych, jest niewielki lub ledwie zauważalny.

NIEZAWODNOŚĆ OPROGRAMOWANIA

W większości prac na temat niezawodności systemów komputerowych wymienia się bardzo często oprogramowanie jako szczególnie słabe ogniwo.

Jakkolwiek w aspekcie produkcyjnym nie ma różnicy między oprogramowaniem podstawowym (systemowym), a użytkowym, to w aspekcie niezawodności i pewności działania sprawa przedstawia się całkiem odmiennie.

Jeden błąd w oprogramowaniu systemowym może w określonych okolicznościach sprawić, że pewna grupa programów użytkowych, a nawet cały system okaże się bez wartości, natomiast błąd w programie użytkowym dyskwalifikuje tylko ten program.

Na podstawie badań przeprowadzonych przez IBM na temat momentu popełnienia błędu podczas opracowywania programu stwierdzono, że błędy robi się w połowie w fazie projektowania, a w połowie w fazie implementacji. Błąd w fazie projektowania wskazuje, że programista rozwiązywanego problemu albo nie rozumie, względnie, że zrozumiał go fałszywie, albo też po prostu nie wie, jak go rozwiązać.

Jeżeli chodzi o błędy popełnione w fazie implementacji, to albo użyto niewłaściwych narzędzi, albo też — gdy użyto właściwych — nie umiano się nimi posłużyć, względnie wykorzystywano je nieprawidłowo.

Według badań przeprowadzonych przez D. T. Rossa na temat przyczyn błędów popełnionych w toku opracowywania systemów oprogramowania 64% wszystkich błędów przypada na fazę analizy i fazę planowania, a tylko 36% na fazę kodowania. Przeważająca część pierwszego oraz ok. 9% drugiego rodzaju błędów wykrywana jest dopiero przy odbiorze programu, bądź nawet później.

STADIA I KOSZTY REALIZACJI OPROGRAMOWANIA

Z przedstawionych rozważań można wyprowadzić postulat wzmożenia wysiłków dla osiągnięcia wyższej wydajności pracy programistów oraz większej niezawodności oprogramowania. Postulat taki powinien objąć wszystkie stadia opracowania systemu — od analizy problemu aż po konserwację programów.

Analiza problemu, to opis zastosowania, które należy zrealizować za pomocą elektronicznego przetwarzania danych. W tym celu najpierw tworzy się w postaci abstrakcyjnej koncepcję, uwzględniającą wymagania w stosunku do danych, przepływ informacji oraz zasady rozwiązania problemu.

Następnym stadium opracowania systemu jest już projekt programu. W stadium tym można wyodrębnić 3 etapy: wybór rozwiązania dla określonej części problemu, podział tego rozwiązania na pojedyncze operacje oraz opisanie tych operacji z punktu widzenia ich wzajemnych związków i zależności.

Kodowanie — kolejne, trzecie stadium prac — polega na tym, że dokładnie już sprecyzowany program poddaje się takiemu przeobrażeniu, że nadaje się on do bezpośredniego opracowania przez komputery.

Zakodowany program powinien być oczywiście dokładnie sprawdzony (przetestowany), a w wyniku testowania program ten powinien już być wolny od błędów formalnych (składniowych) oraz błędów logicznych.

Ostatnim stadium jest opracowanie dokumentacji, a następnie konserwacja systemu. Jakkolwiek czynności te właściwie nie należą już do zakresu prac produkcyjnych w tym znaczeniu, jak prace wykonywane w trzech pierwszych stadiach, niemniej ich znaczenie jest również ważne.

Czynności konserwacji i ewentualnej rozbudowy oprogramowania rzutują szczególnie silnie na jego koszty. Według badań wspomnianego wcześniej D. T. Rossa, ta część realizacji projektu kosztuje przeciętnie od 200 do 400% łącznych nakładów poniesionych na realizację pierwszych trzech stadiów projektowania.

Obecna typowa struktura kosztów realizacji złożonego programu kształtuje się następująco: analiza i projektowanie — 35%, kodowanie i sprawdzenie (przetestowanie) elementów programu — 20%, sprawdzenie całości i instalacja programu — 45%.

USPRAWNIANIE PROCESU PROGRAMOWANIA

Przemysł oprogramowania, a więc zarówno wytwórcy komputerów, jak i wyspecjalizowani, niezależni od nich producenci samego oprogramowania, a także informatyczne przedsiębiorstwa doradcze, w ciągu ubiegłych lat wypracowały liczne metody i sposoby programowania oraz specjalne, służące temu celowi pomoce. Ich zadaniem jest zwiększenie efektywności procesu projektowania, z jednej strony poprzez podniesienie wydajności pracy programistów, z drugiej zaś, poprzez nadanie programowi większej przejrzystości, a także poprawę jakości dokumentacji projektowej. Metody wpływają bezpośrednio głównie na sprawność i wydajność procesu projektowania, natomiast pomoce mają na celu w pierwszym rzędzie ułatwienie i usprawnienie pracy projektanta.

Obecnie różnego rodzaju pomoce (narzędzia) projektowe oferowane są prawie dla wszystkich stadiów opracowania programu — od analizy problemowej aż do nadzoru i konserwacji.

PROGRAMOWANIE STRUKTURALNE

Żadna z metod opracowania programu nie wywołała w przeszłości tylu sprzecznych opinii i tak zaciętych sporów jak „programowanie strukturalne". Niesłychanie żywo, a nawet emocjonalnie prowadzona była dyskusja, w równej mierze przez naukowców — teoretyków, jak programistów — praktyków. Można by przy tej okazji czynić zarzut kręgom naukowym, że przez rok prowadziły wewnętrzną dyskusję, która nie docierała do tych, których właściwie dotyczyła tzn. programistów.

O cóż chodzi? Programowanie strukturalne jest metodą programowania umożliwiającą — poprzez zachowanie określonej dyscypliny postępowania — zbudowanie programu odznaczającego się dużą przejrzystością. Program zbudowany według reguł tej metody jest łatwo czytelny, ma jednolitą strukturę i dzięki temu umożliwia szybkie i łatwe zapoznanie się z nim przez programistów stykających się z nim po raz pierwszy, tzn. nie uczestniczących uprzednio w jego opracowaniu.

Trzeba dodać, że same reguły programowania strukturalnego są bardzo proste, a więc i łatwe do opanowania. Normują one między innymi rozgałęzienia programu, dzięki czemu sterowanie jego przebiegiem jest całkowicie ujednolicone. Program można zatem czytać od góry do dołu, bez potrzeby — jak tego wymaga wcześniej używana metoda „makaronowa" — nawrotów z dołu do góry, stosownie do logiki rozgałęzień. Programowanie strukturalne, dzięki wspomnianej lepszej czytelności programu, zmniejszyło znacznie częstość popełniania błędów, tworząc tym samym dobre warunki dla eksploatacji, konserwacji i ewentualnego rozwijania programu.

GŁÓWNY PROGRAMISTA I JEGO ZESPÓŁ

W dążeniu do zwiększenia wydajności procesu projektowania zwrócono także uwagę na metody natury czysto organizacyjnej. Jedna z nich, znana pod nazwą „zespół głównego programisty" *(ang. chief-programmer-team)*, opiera się na bardziej ścisłym niż dotąd stosowano, rozdzieleniu kompetencji poszczególnych pracowników — członków zespołu oraz poddaniu projektowania poszczególnych części programu nadzorowi i kierownictwu „głównego programisty", osoby najbardziej doświadczonej i — co zwykle idzie w parze — o największym autorytecie w zespole.

Rozwiązania organizacyjne tej metody, a zwłaszcza zróżnicowanie i rozłożenie odpowiedzialności, nastawione są w równej mierze na lepsze „dowodzenie" zespołem projektowym, jak i na wyegzekwowanie większej wydajności pracy poszczególnych jego członków.

Zadaniem głównego programisty, oprócz ogólnego kierowania zespołem, jest zaprojektowanie całości programu, podzielenie go na odpowiednie części z przekazaniem ich do wykonania poszczególnym członkom zespołu, a następnie koordynowanie i kontrolowanie takiego wykonania tych części, aby złożony z nich program spełniał założone wymagania. Główny programista odpowiada za całość procesu projektowania oraz gotowy program. W jego pracy pomagają mu:

1) — zastępca, w równym stopniu znający wszystkie niuanse techniki programowania, współdecydujący w najważniejszych rozstrzygnięciach projektowych i będący w stanie w każdej chwili przejąć kierownictwo zespołu

2) — pracownik prowadzący pełną ewidencję postępu prac nad projektem w postaci wpisów w specjalnej kartotece realizacji i odciążający w ten sposób od „papierkowych" obowiązków pozostałych członków zespołu.

Tak zorganizowany zespół, jak potwierdziła praktyka, pozwala znacznie zwiększyć wydajność pracy poszczególnych programistów, a w konsekwencji zmniejszyć ich liczbę i efektywniej wykorzystać. W przypadku bardzo dużych projektów może być powołanych wiele tego rodzaju zespołów, np. dla opracowania poszczególnych podsystemów.

PROGRAMOWANIE KONWERSACYJNE

Przy tej metodzie projektowania programista, wspomagany przez interekcyjny terminal, pracuje bezpośrednio z komputerem. Ma on do dyspozycji zwykle urządzenie klawiaturowo-ekranowe, bądź klawiaturę i drukarkę. Za pomocą klawiatury przekazuje komputerowi dane, instrukcje lub zapytania. Otrzymuje natychmiast odpowiedzi wyświetlone na ekranie lub w postaci wydruku. Dzięki temu między komputerem a programistą powstaje rodzaj dialogu, co znalazło odbicie w nazwie tej metody.

Programowanie konwersacyjne daje najlepsze wyniki w fazie testowania programu, natomiast w innych fazach projektowania nie można jeszcze w pełni wykorzystać możliwości zaoszczędzenia czasu i pracy, jakie metodę tę charakteryzują. Według analizy przeprowadzonej przez firmę IBM, programowanie konwersacyjne umożliwia podniesienie wydajności projektowania w granicach od 30 do 100% — w zależności od rodzaju programu. To ogólne stwierdzenie oparte jest na przebadaniu dużej liczby bardzo zróżnicowanych programów.

PROJEKTOWANIE PROGRAMU „OD GÓRY DO DOŁU"

Ten sposób projektowania, znany w literaturze zagranicznej pod nazwą „Top-Down Programming", charakterystyczny jest tym, że zarówno opracowanie, jak i testowanie programu, można realizować równolegle. Projekt programu z reguły powstaje w ten sposób, że przy jego budowie postępuje się wg zasady od „ogółu" do „szczegółu". Fakt ten stał się kamieniem węgielnym metody projektowania „od góry do dołu" polegającej na tym, że każdy problem „rozwija" i kontroluje się, idąc poprzez znormalizowane stopnie ku coraz to dokładniejszemu sprecyzowaniu poszczególnych funkcji. Stąd też w metodzie tej każda jednostka projektująca (oprócz pierwszej) może przystąpić do pracy nie wcześniej, jak po wykonaniu i sprawdzeniu bezpośrednio poprzedzającego stadium opracowania programu przez inną jednostkę.

Metoda ta zmusza programistów do ciągłego pamiętania o całości systemu, dając zarazem ścisłą informację, w którym punkcie realizacji projektu trwa jeszcze praca. Dzięki temu można dokładnie planować i dobrze wykorzystywać środki przeznaczone na realizację projektu. Metoda „od góry do dołu" wyklucza bardzo trudne do oszacowania ryzyko scalania programu z części, występujące w innych metodach, i pozwala łatwiej określać jakość opracowanego programu. Stąd też jest ona bardzo często stosowana łącznie z metodą „głównego programisty".

TABLICE DECYZYJNE

Tablica decyzyjna jest ujętym w formie tabelarycznej przyporządkowaniem występujących w programie rozgałęzień i odpowiadających im warunków. Można ją rozpatrywać jako pewnego rodzaju uzupełnienie schematu działania programu.

Zestawienie takie sporządza się na podobieństwo macierzy, gdzie w wierszach zapisane są warunki będące przedmiotem pytania, a w kolumnach czynności, które należy wykonać. Pozwala to uzyskać zwięzły przegląd wszystkich rozgałęzień programu oraz ich warunków. Ponieważ istniejące jako dogodne narzędzie programowania generatory tablic decyzyjnych są w podstawowej mierze określone przez rodzaj stosowanej metody przetwarzania, producenci oprogramowania często oferują różne ich kombinacje. Dlatego też godne jest zalecenia, aby przy zakupie kompilatora tablic zwrócić uwagę na to, jakie są podstawowe wymagania generatora (język programowania, pojemność pamięci, czas przebiegu, warunki testowania) oraz w jakim języku będą sformułowane wygenerowane programy.

* * *

Omówione tu w głównych zarysach podstawowe współczesne metody i narzędzia programowania nie wyczerpują oczywiście wszystkich możliwych, żeby wspomnieć choćby tylko o takich instrumentach usprawnienia programowania, jak systemy oparte na schematach blokowych, czy kartoteki prowadzenia projektu. Podobnie rzecz się ma z generatorami programów rozwiązujących wiele problemów szczegółowych, które programista może — np. przez wprowadzenie parametrów — wykorzystać do określenia i wygenerowania programów dla rozwiązania własnych indywidualnych zadań.

A oprócz tego można wreszcie w ogóle zrezygnować z programowania indywidualnego na korzyść zastosowania tak bogatego już i coraz bardziej efektywnego oprogramowania uniwersalnego.

ZBIGNIEW ŁADOŚ
Narodowy Bank Polski
Warszawa

Skomputeryzowane dokumenty

Zadaniem każdego przedsiębiorstwa jest nie tylko ilościowe i jakościowe wykonanie dostaw i usług w planowanym czasie, ale również przeprowadzenie terminowych rozliczeń pieniężnych z kontrahentami, odbiorcami tych dostaw lub usług. Wpływ środków pieniężnych na rachunek bankowy przedsiębiorstwa jest potwierdzeniem, iż ilość i jakość dostaw lub usług zadowoliła odbiorcę, który nie zgłosił zastrzeżeń i nie odmówił zapłaty.

W ostatnich latach kilkadziesiąt przedsiębiorstw w kraju objęło komputeryzacją rozliczenia finansowe z kontrahentami, włączając do swych systemów elektronicznego przetwarzania danych również ewidencje dotyczące tych rozrachunków. Systemy lub podsystemy te umożliwiają przedsiębiorstwu szybką analizę zbytu, kontrolę przeterminowanych należności oraz sporządzanie różnych sprawozdań, a także zmniejszenie pracochłonności i poprawę terminowości tych prac, realizowanych na własnych komputerach lub minikomputerach albo też w ramach usług sieci ZETO. Szczególnie duże znaczenie dla przedsiębiorstwa ma szybkie wystawienie i wysłanie specyfiki towarów lub usług, a następnie faktur i dyspozycji dotyczących rozliczeń pieniężnych dla banków, gdyż jest to jeden z czynników warunkujących terminowy wpływ środków finansowych, potrzebnych przedsiębiorstwu na opłacenie własnych zobowiązań. Wprawdzie w razie braku tych środków przedsiębiorstwa mogą występować o kredyt bankowy, ale kredyt ten jest zazwyczaj ograniczony, a ponadto wiąże się z opłaceniem odsetek, co powiększa koszty przedsiębiorstwa i wpływa na pogorszenie się wyników finansowych jego działalności. Nieterminowe spłacenie przez przedsiębiorstwo zobowiązań, naraża je również na zapłacenie karnych odsetek kontrahentom rozliczeń. Z kolei kontrahenci rozliczeń, którzy nie otrzymują we właściwym terminie należnych im środków finansowych, napotykają trudności w spłacie własnych zobowiązań. W ten sposób powstaje łańcuch wzajemnych zależności pomiędzy różnymi kontrahentami rozliczeń, powodując w tych przedsiębiorstwach wzrost ilości przeterminowanych zależności i zobowiązań finansowych. Banki podejmują odpowiednie środki zaradcze aby zapobiec tego rodzaju niepożądanym zjawiskom ekonomicznym. Niekiedy jednak przyczyny tych zakłóceń mają charakter organizacyjny i wynikają np. z trudności w bieżącym, terminowym fakturowaniu oraz w ekspedycji dokumentów finansowych do kontrahentów rozliczeń. Dlatego powszechniejsza niż dotychczas komputeryzacja rozliczeń finansowych, ma znaczenie nie tylko dla samych przedsiębiorstw, ale również dla całej gospodarki narodowej, ponieważ zapobiega narastaniu wzajemnych zaległości pomiędzy przedsiębiorstwami w regulowaniu zobowiązań za dostawy i usługi.

Narodowy Bank Polski jako organizator i pośrednik rozliczeń pieniężnych bardzo interesuje się rozwojem krajowej komputeryzacji tych rozliczeń. W związku z tym bank ten wydał w połowie 1978 r. „Wytyczne dla jednostek gospodarki uspołecznionej sporządzających za pomocą komputerów dokumenty dla rozliczeń pieniężnych dokonywanych przez banki". Wytyczne te w fazie opracowania były konsultowane z wieloma przedsiębiorstwami oraz z innymi bankami. Zawierają one szczegółowe zasady sporządzania komputerowych dokumentów rozliczeniowych i powinny ułatwić przedsiębiorstwu prace przy projektowaniu systemów przetwarzania danych finansowych. Wytyczne podają również wzory następujących wymaganych przez banki dokumentów: poleceń przelewu środków pieniężnych, poleceń pobrania środków pieniężnych, zestawień zbiorczych do tych dokumentów oraz żądań zapłaty dotyczących inkasa zależności fakturowych. Załączone wzory powyższych rodzajów dokumentów określają zawartość wierszy i kolumn w dostosowaniu do charakterystyki zapisu drukarki komputerowej.

Za zgodą NBP dopuszczalne są drobne odchylenia od podanych wzorów, uwarunkowane względami programowania, pod warunkiem jednak, iż przyczynią się one do poprawy czytelności i nie spowodują zwiększenia ustalonego formatu dokumentu. Ostatni warunek wiąże się z tym, iż większość przedsiębiorstw w kraju posługuje się w rozliczeniach pieniężnych nadal tradycyjnymi formularzami i dlatego skomputeryzowane wersje muszą być podobne do formularzy wypełnianych ręcznie.

Oprócz podanych wzorów formularzy, wytyczne zawierają szczegółowe zasady kontroli danych w dokumentach rozliczeniowych oraz sposób wypełniania tych dokumentów przez komputer.

Ogólna liczba dokumentów składanych przez przedsiębiorstwa w krajowych placówkach bankowych dla dokonania rozliczeń pieniężnych wynosi obecnie już około 1000000 sztuk dziennie. Są one tam rejestrowane, sortowane i kontrolowane, stanowiąc podstawę do opracowywania wtórnych dokumentów rozliczeniowych. Niezależnie od tego, dokumenty te w większości przypadków muszą być przesyłane do innych placówek tego samego lub innego banku, w których prowadzone są rachunki poszczególnych kontrahentów rozliczeń pieniężnych.

Jak wiadomo w bankach obowiązuje generalna zasada bieżącego wykonywania dyspozycji rozliczeniowych (bez zaległości). Z uwagi na szczególnie duże ilości tych dyspozycji, w bankach musi obowiązywać jednolite, bardzo ścisłe przestrzeganie zasad organizacji obsługi tych rozliczeń, zwłaszcza zaś przestrzeganie stosowania jednolitych wzorów formularzy. Dlatego też komputeryzujące się przedsiębiorstwa nie mogą stosować wzorów dokumentów rozliczeń pieniężnych uwzględniających tylko indywidualne potrzeby organizacyjne. Praktyka wykazuje, że szereg skomputeryzowanych przedsiębiorstw stosuje już nietypowe własne wydruki, powodujące zmniejszenie wydajności pracy banków przy obsłudze masowych rozliczeń pieniężnych. Stąd też omawiane wytyczne nakładają na te przedsiębiorstwa obowiązek dostosowania wzorów dokumentów rozliczeniowych do jednolitych zasad.

Ponieważ wytyczne te są pierwszym tego typu wydawnictwem NBP zawierającym szczegółowe zasady zastosowania komputerów do rozliczeń pieniężnych dokonywanych poprzez banki, przeto wydaje się wskazanym podanie na ten temat kilku informacji dodatkowych.

Wytyczne NBP może uzyskać każde zainteresowane przedsiębiorstwo składając zamówienie w placówce bankowej, w której posiada rachunek bankowy. Również dodatkowe informacje na ten temat można uzyskiwać za pośrednictwem tych placówek bankowych. Aby uniknąć przyszłych kosztownych przeróbek systemów EPD wskazane jest zapoznanie się z wytycznymi już na etapie ustalania tych fragmentów założeń systemu informatycznego, które mają bezpośredni lub pośredni związek z rozliczeniami finansowymi (rachunkowość, zbyt, płace itp.). Wytyczne te trzeba dostarczyć projektantom i programistom systemu.

Dokument ten zawiera również sytematykę różnego typu danych, m.in. nazewnictwa placówek bankowych, kontrahentów rozliczeń itp. W celu ułatwienia przedsiębiorstwom wykorzystania już istniejących zbiorów nazw przedsiębiorstw (kontrahentów rozliczeń) zapisanych na taśmach lub dyskach magnetycznych, uwzględniono możliwość drukowania w dokumentach rozliczeniowych przeznaczonych dla banków jedynie nazwy jednostki oraz nazwy miejscowości, będącej siedzibą tej jednostki (maksymalnie 125 miejsc znakowych w 5 wierszach po 25 znaków), tzn. bez potrzeby podawania nazwy ulicy, numeru domu i kodu pocztowego. Mogą być stosowane również skróty nazw przedsiębiorstw, pod warunkiem jednak, że są one zrozumiałe.

Wytyczne obejmują również szczegółowe informacje dotyczące jednolitych zasad numeracji rachunków w polskich bankach. Pomimo wielu publikacji na ten temat (m.in. „INFORMATYKA" nr 8/77 str. 26), szczegóły te nie są dostatecznie znane projektantom i programistom w przedsię-

dla bankowych rozliczeń pieniężnych

biorstwach, co powoduje, iż obecnie programy komputerowe przeważnie nie kontrolują formalnej poprawności tych numerów. W konsekwencji rozliczenia pieniężne z błędnymi numerami rachunków bankowych mogą być omyłkowo zarachowane na zupełnie niewłaściwych rachunkach bankowych. Dlatego sprawa ta ma bardzo duże znaczenie również dla skomputeryzowanych lub komputeryzujących się przedsiębiorstw, w których błędy w numerach rachunków bankowych mogą powstawać np. przy okazji tworzenia maszynowych nośników danych. Kontrola taka może być przeprowadzana między innymi za pomocą metody „module 10" (patrz „INFORMATYKA" nr 1/79 str. 13).

Oprócz numeru rachunku bankowego w dokumentach rozliczeniowych występuje niekiedy klasyfikacja budżetowa, która jest bezwzględnie potrzebna w banku dla szczegółowego ewidencjonowania operacji pieniężnych finansowanych z Budżetu Centralnego. Brak tej klasyfikacji powoduje w placówce bankowej dodatkową pracę związaną z uciążliwym wyjaśnianiem jej rodzaju. Informacja ta, której systematykę i sposób kontroli podano w wytycznych, powinna być również wprowadzona do zbiorów danych gromadzonych przez system komputerowy. Praktyka wykazała, że szereg już skomputeryzowanych dużych przedsiębiorstw, pominęło przy tworzeniu swych systemów informatycznych tę ważną dla rozliczeń pieniężnych informację.

Inną informacją występującą w dokumentach rozliczeniowych jest symbol planu kasowego, stosowany wyłącznie w poleceniach przelewu jednostek gospodarki uspołecznionej na rzecz ludności (np. przelewy poborów na rachunki oszczędnościowo-rozliczeniowe w PKO lub w bankach spółdzielczych, dokonywane w związku z obliczeniami płacowymi).

Aby zaoszczędzić znaczne ilości papieru tabulogramowego oraz czas pracy komputerów, banki zgodziły się na otrzymywanie tych dwóch odcinków polecenia pobrania lub polecenia przelewu, zamiast czterech egzemplarzy analogicznych formularzy wypełnianych przy stosowaniu metody tradycyjnej (PuK 402 i PuK 403). Ponadto banki zrezygnowały z obowiązku podpisywania poszczególnych poleceń przelewu i pobrania, zaś pieczęć firmowa zamieszczana na tych dokumentach może być również zastąpiona odpowiednim nadrukiem z uwzględnieniem zasad opisanych szczegółowo w wytycznych.

Wydaje się, iż są to znaczne udogodnienia dla tych przedsiębiorstw, które jednorazowo dostarczają do banku kilkaset lub kilka tysięcy dyspozycji rozliczeniowych. Wprowadzenie tych udogodnień było możliwe dzięki sporządzaniu za pomocą komputera przez przedsiębiorstwa specjalnego tabulogramu: zestawienia poleceń przelewu lub zestawienia poleceń pobrania.

Pomimo ogólnie znanych trudności związanych z wykonaniem w drukarniach krajowych formularzy na papierze tabulogramowym, zakłada się, że jednostki gospodarki uspołecznionej będą usilnie dążyć do uzyskania odpowiednich gotowych komputerowych druków formularzy poleceń pobrania, poleceń przelewu i żądań zapłaty.

Ich wypełnienie przez komputer pozwala bowiem uzyskać następujące efekty:

1) zmniejszenie ilości czasu pracy komputera (komputer nie drukuje wówczas linii i stałych tekstów)

2) ułatwienie sortowania dokumentów rozliczeniowych (dzięki nadrukowi wykonanemu w różnych kolorach ustalonych w wytycznych, zależnie od przeznaczenia poszczególnych odcinków tych dokumentów)

3) zmniejszenie liczby błędów przy obsłudze rozliczeń pieniężnych w wyniku lepszej czytelności drukowanego formularza, zarówno dla pracowników przedsiębiorstw, jak i placówek bankowych. Wobec masowego charakteru tego rodzaju dokumentów, ma to istotne znaczenie dla zwiększenia dokładności i wydajności pracy w komórkach finansowych przedsiębiorstw i w placówkach bankowych.

Wiele przedsiębiorstw ma już pozytywne doświadczenia w zastosowaniu formularzy komputerowych produkcji krajowej. Niemniej jednak na wypadek przejściowego braku formularzy drukowanych wytyczne zawierają również dokładne wskazówki dotyczące zastosowania dokumentów w całości sporządzanych za pomocą komputerów.

Stosowanie gotowych formularzy nie jest konieczne dla „Zestawień poleceń pobrania" oraz dla „Zestawień poleceń przelewu", albowiem są to dokumenty, które nie są przesyłane do innych placówek bankowych i do kontrahentów rozliczeń. Jeden egzemplarz pozostaje w macierzystej dla przedsiębiorstwa placówce bankowej, a drugi jest zwracany do skomputeryzowanego przedsiębiorstwa przy wyciągu z rachunku bankowego jako potwierdzenie wykonania przez bank dyspozycji przedsiębiorstwa. Zakładanie różnorodnych formularzy w drukarce komputera powoduje jak wiadomo przerwy w eksploatacji i dlatego proponuje się rezygnowanie z formularzy w tych przypadkach, gdy nie jest to bezwzględnie konieczne.

Szczególnie nieczytelne z uwagi na dużą ilość danych, byłyby skomputeryzowane „żądania zapłaty" bez użycia gotowych formularzy. Format skomputeryzowanego „żądania zapłaty" musi być zbliżony do wymiarów stosowanych w bankach kartotek, w których są okresowo przechowywane żądania zapłaty jeszcze nie zrealizowane. Kartoteki te są dostosowane do żądań zapłaty w formacie A6, wypełnianych metodami tradycyjnymi (formularze powszechnego użytku PuK 406). Dokumenty te muszą być w tych kartotekach często przeglądane oraz przesyłane do innych placówek bankowych i dlatego ważny jest zarówno format, jak i układ treści w tych dokumentach. Należy więc z reguły dążyć do stosowania maszynowych formularzy żądań zapłaty, zaś wzór bezformularzowy tego dokumentu może być wykorzystywany jedynie sporadycznie.

Byłoby wskazane, aby drukowaniem i zaopatrzeniem jednostek gospodarki uspołecznionej w omówione formularze rozliczeń zajęła się centralnie jedna instytucja w kraju, podobnie do zasad przyjętych przy dystrybucji analogicznych formularzy wypełnianych metodami tradycyjnymi (PuK: 406, 403, 402). Drukowanie ujednoliconych formularzy dla każdego przedsiębiorstwa oddzielnie powoduje bowiem dodatkowe nieuzasadnione ekonomicznie nadmierne koszty a także zmniejsza efektywność wykorzystania potencjału przemysłu poligraficznego.

Wytyczne zawierają również wskazówki, które będą wykorzystane w następnym etapie współpracy banku ze skomputeryzowanymi przedsiębiorstwami, polegającym na stosowaniu taśmy magnetycznej z dyspozycjami rozliczeniowymi przedsiębiorstwa („INFORMATYKA" nr 3/78 str. 3). Realizacja tego etapu zależy w dużym stopniu od bezbłędności i kompletności danych dostarczanych do banku w formie dokumentów sporządzanych na komputerach. Obecnie przeprowadzane są przez NBP wspólnie z dużym przedsiębiorstwem w Warszawie próby użytkowe, w wyniku których zostaną przygotowane następne wytyczne zawierające szczegółowe wskazówki na temat sposobu przygotowania przez przedsiębiorstwa taśmy magnetycznej.

Ponieważ, jak już wspomniano, niektóre przedsiębiorstwa niestety nie zastosowały w programach odpowiednich kontroli poprawności danych dotyczących rozliczeń pieniężnych, co nie tylko przysparza obecnie placówkom bankowym wiele trudności przy usuwaniu błędów, ale również uniemożliwia zastosowanie w najbliższym czasie taśm magnetycznych dla współpracy banku z takimi przedsiębiorstwami. Należy mieć nadzieję, iż obecne wytyczne NBP przyczynią się do szybkiego usunięcia tych niedociągnięć oraz ujednolicenia danych przekazywanych do banku w formie dokumentów papierowych, a w konsekwencji przyspieszą realizację wspomnianego następnego etapu automatyzacji, jakim będzie przekazywanie do banków dyspozycji rozliczeniowych przez przedsiębiorstwa w postaci zapisu na taśmach magnetycznych.

Poniższym artykułem rozpoczynamy serię publikacji poświęconą wykorzystaniu maszyn cyfrowych w procesach produkcyjnych. Tematyka ta wydaje się nam niezmiernie istotna, zwłaszcza w obecnej sytuacji, w której liczyć się zaczynają najbardziej wymierne efekty zastosowań techniki komputerowej. Dlatego też, zapraszając Czytelników do udziału w projektowanym cyklu, zamierzamy drukować na naszych łamach nie tylko wyniki dociekań teoretycznych w tej dziedzinie, ale i opisy konkretnych realizacji przemysłowych systemów komputerowych. Dotyczy to zwłaszcza zastosowań maszyn cyfrowych w dyskretnych procesach produkcyjnych, tak ważnych z punktu widzenia naszej gospodarki, o których komputeryzacji pisano dotąd niezmiernie rzadko.

ANDRZEJ GOŚCIŃSKI, EDWARD NAWROCKI
Instytut Informatyki i Automatyki AGH
Kraków

Komputerowe sterowanie procesami dyskretnymi

Prawie każdy proces produkcyjny, przy odpowiednio postawionym celu sterowania, warunkach współdziałania z innymi procesami, przyjętych wskaźnikach jakości oraz horyzoncie czasowym sterowania, może być traktowany jako proces dyskretny. Ogólnie można powiedzieć, że dyskretny charakter procesów implikowany jest nie tylko przebiegiem określonych zjawisk oraz występowaniem zdarzeń w ściśle określonych momentach lecz również tym, że operacje technologiczne (obsługi) są realizowane na skończonych partiach materiału, że procesy (obiekty) mogą się znajdować w różnych stanach technologicznych (wywołanych potrzebą pracy obiektów w szerokim zakresie zmian parametrów), a wreszcie faktem, że same działania sterujące oraz ocenę efektów sterowania dogodnie jest często odnosić do skończonych przedziałów czasu (np. sterowanie typu operatorskiego lub sterowanie procesami periodycznymi).

Przy takim ujęciu bardzo wiele procesów może być traktowane jako dyskretne — wynika to zarówno z ich charakteru, jak też różnorodności stawianych zadań sterowania. Oczywistą konsekwencją rozważania tak szerokiej klasy procesów jest brak jednolitego ich opisu, nawet przy wyższym poziomie uogólnienia stawianych zadań sterowania. W tej sytuacji wydaje się niezmiernie celowe stworzenie, chociażby w formie pewnej próby, w miarę ogólnego ujęcia problematyki sterowania procesami dyskretnymi. Ponieważ aktualnie trudno sobie wyobrazić realizację procesu sterowania złożonym procesem bez użycia komputera, skupimy się tu na niektórych problemach komputerowego sterowania procesami dyskretnymi.

ZADANIA STEROWANIA

Problemy decyzyjne dotyczące dyskretnych procesów produkcyjnych można rozważać w kontekście następujących zagadnień:

— projektowania nowych ciągów lub zakładów produkcyjnych, posiadających dyskretny charakter ze zwróceniem uwagi na dobór optymalnych parametrów całego ciągu, jego struktury, parametrów agregatów produkcyjnych, oprzyrządowania technologicznego oraz sprawdzenie rozmaitych własności przyjętych lub proponowanych technologii

— modernizacji istniejących ciągów produkcyjnych, ze zwróceniem uwagi na poprawę efektywności lub zwiększenie zakresu działania systemu dyspozytorskiego, określenie warunków i możliwości bardziej efektywnego wykorzystania istniejącego ciągu technologicznego, poszczególnych agregatów produkcyjnych, systemu transportowego itp.

— sterowania (kierowania) całym ciągiem technologicznym, współpracy między ciągami oraz współpracy poszczególnych agregatów i operacji technologicznych.

Należy zwrócić uwagę, że rozwiązanie dwóch pierwszych problemów musi zawierać (lub przynajmniej uwzględniać) problemy sterowania i dlatego prowadzone dalej rozważania koncentrują się wokół tego właśnie problemu. Za przypisaniem problemowi sterowania pierwszoplanowej roli przemawia również fakt, że w przypadkach projektowania

Doc. dr hab. inż. Andrzej GOŚCIŃSKI ukończył studia w Akademii Górniczo-Hutniczej im. St. Staszica w Krakowie (1968 r.), doktorat — 1973 r., habilitacja — 1976 r. Obecnie kierownik Zakładu Maszyn Matematycznych w Instytucie Informatyki i Automatyki AGH. Współtwórca kierunku „Informatyka" w Akademii Górniczo-Hutniczej. Specjalizuje się w komputerowych systemach o działaniu bezpośrednim.

Doc. dr hab. inż. Edward NAWARECKI ukończył studia w Akademii Górniczo-Hutniczej im. St. Staszica w Krakowie (1958 r.), doktorat — 1967 r., habilitacja — 1975 r. Obecnie kierownik Zakładu Informatyki w Instytucie Informatyki i Automatyki AGH. Współtwórca kierunku „Informatyka" w Akademii Górniczo-Hutniczej. Specjalizuje się w zastosowaniach informatyki w sterowaniu procesami i w zarządzaniu.

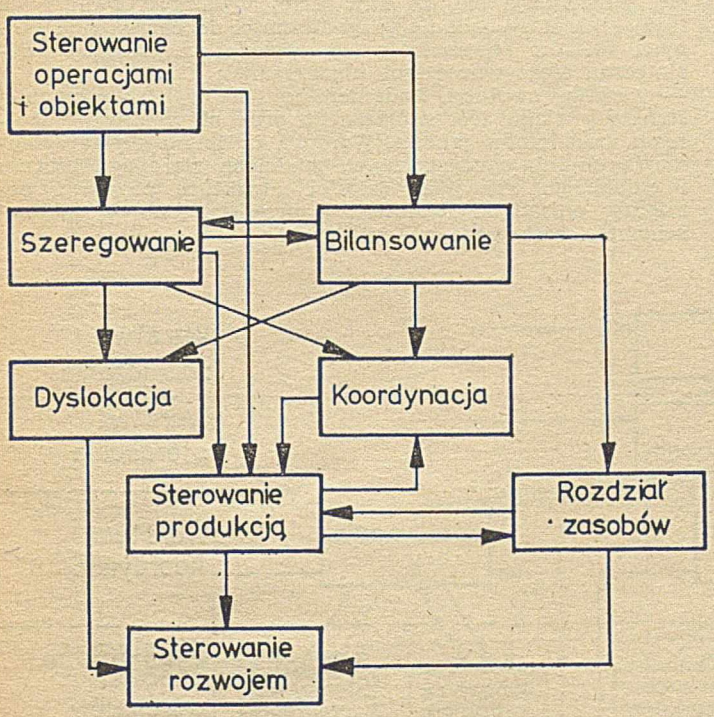
nowych lub modernizacji istniejących procesów (ciągów technologicznych), zwiększa się jedynie stopień swobody przy formułowaniu zadania sterowania i doborze środków jego realizacji. Natomiast sprawą zasadniczą pozostaje w dalszym ciągu wybór modeli, algorytmów i środków technologicznych.

Jak już wspomniano, poszerzona interpretacja pojęcia dyskretności pociąga za sobą dużą różnorodność procesów zaliczanych do tej klasy. Dlatego też usiłując wprowadzić pewną klasyfikację typowych w danym zakresie problemów decyzyjnych, musimy odejść od fizykalnych cech procesu i wysunąć na plan pierwszy aspekty funkcjonalne oraz formalną postać zadania.

Przy takim punkcie widzenia celowe wydaje się wyróżnienie następujących typów zadań:
— sterowanie oddzielnymi operacjami i obiektami
— szeregowanie
— równoważenie linii montażowej
— rozlokowanie prac i zadań (dyslokacja)
— koordynacja agregatów produkcyjnych, oprzyrządowania i środków transportu w ramach ciągu technologicznego, procesu lub całego przedsiębiorstwa
— sterowanie produkcją i zapasami
— rozdział zasobów
— sterowanie rozwojem (w długich horyzontach czasu).

Zadania powyższe zostały podane w kolejności określonej przez wzrastający stopień komplikacji, przy czym każde zadanie o większej złożoności może zawierać w sobie zadania prostsze.

Jakkolwiek ścisłe określenie wzajemnych relacji pomiędzy poszczególnymi zadaniami jest bardzo trudne, to jednak dla wyrobienia sobie niezbędnego ogólnego poglądu na tę sprawę, powiązania najbardziej charakterystyczne przedstawiono schematycznie na rysunku 1.



Rys. 1. Powiązania między zadaniami

MODELE PROCESÓW DYSKRETNYCH

Określana forma modelu wynikać może zarówno z własności opisywanego procesu, jak też sformułowanego zadania, a w tym również postaci wskaźnika jakości.

Problem wyboru wskaźników jakości jest w pewnych przypadkach zagadnieniem naturalnym i wynika z celu sterowania całym zakładem, czy też określonym ciągiem produkcyjnym. Często jednak jest to problem złożony, przy czym złożoność ta jest implikowana zarówno od strony procesu (złożoność obiektów i struktury technologicznej), jak i od strony układu sterowania, któremu te wskaźniki mają służyć (wielopoziomowość struktury sterowania, konieczność dekompozycji lub agregatyzacji). Przy wyborze wskaźników jakości trzeba również często uwzględniać różnorodność stanów technologicznych, w jakich mogą znajdować się poszczególne operacje, obiekty lub całe ciągi produkcyjne.

Przy konstruowaniu modelu konieczne jest zapewnienie zarówno jego wystarczającej adekwatności, jak też względnej prostoty, warunkującej możliwość sterowania w czasie rzeczywistym. Wymagania powyższe pozostają na ogół we wzajemnej sprzeczności, szczególnie gdy chodzi o modele złożonych procesów poddanych oddziaływaniu czynników przypadkowych, lub pozostających w określonych relacjach ze środowiskiem. Dążąc do zachowania rozsądnego kompromisu trzeba niekiedy stosować różne formy modeli dla opisu tego samego procesu w zależności od zadania (fragmentu zadania) sterowania, realizacji którego mają służyć.

Ze względu na sposób realizacji modelu, można wyróżnić dwie zasadnicze klasy:

1) modele formalne
2) modele symulacyjne.

Należy podkreślić, że określenie „model formalny" ma tu raczej charakter umowny, gdyż w obydwu przypadkach konieczne jest wprowadzenie pewnego opisu formalnego, jednakże sposób wykorzystania tych opisów jest różny. (Stosowanie nieraz w literaturze określenia „model analityczny" wydaje się jeszcze bardziej sztuczne).

Jako najbardziej charakterystyczne formy stosowanych opisów należy wymienić:
— modele analityczne, konstruowane przy wykorzystaniu zmiennych stanu, opisów macierzowo-wektorowych, w których zmienne niejednokrotnie przyjmują wartości całkowito-liczbowe, boolowskie itp.
— modele teorio-mnogościowe z wykorzystaniem teorii relacji (w tym również „fuzzy")
— automatowe (zarówno deterministyczne, jak i stochastyczne)
— grafowe (z uwzględnieniem modeli sieciowych oraz grafów skierowanych i funkcyjnych)
— forresterowskie, które swą formą odpowiadają wprawdzie procesom ciągłym, są jednak bardzo efektywne tam, gdzie przedmiotem badania są problemy dynamiki procesów oraz prognozowania ich rozwoju, w tym również przy nadaniu interpretacji dyskretnej
— łańcuchy Markowa.

Wymienione rodzaje formalizmów mogą być w zasadzie stosowane przy konstruowaniu obu podanych poprzednio klas modeli, przy czym na podstawie przesłanek teoretycznych oraz znanych rozwiązań można jednakże określić pewne preferencje w zakresie obszarów ich zastosowań.

Należy zwrócić uwagę, że w badaniach indentyfikacyjnych oraz przy konstruowaniu modeli złożonych procesów należy dążyć zarówno do odwzorowania własności poszczególnych obiektów i operacji, jak też powiązań strukturalnych, występujących w ramach całego procesu technologicznego. W warunkach dużej złożoności, przy szerokim zakresie zmian parametrów oraz przy intensywnych oddziaływaniach środowiska lub czynników przypadkowych, celowe okazać się może konstruowanie oddzielnych modeli dla różnych stanów i sytuacji technologicznych (być może przy zastosowaniu różnorodnych formalizmów). Otrzymany w ten sposób model całościowy nabiera cech ewolucyjnych, działając odpowiednio do wyników prowadzonego na bieżąco rozpoznania sytuacji (stanu).

Dla obiektów lub systemów o szczególnie dużej złożoności (np. reaktor jądrowy, systemy rozwoju), może okazać się, że modele formalne, a nawet symulacyjne są tak złożone, iż nie mogą być wykorzystane bezpośrednio do celów sterowania. Zdarzają się również przypadki, gdy badania identyfikacyjne lub teoretyczne poznanie własności procesu, są niemożliwe lub zbyt kosztowne (czasochłonne). W obu tych sytuacjach, postępowanie z konieczności ogranicza się do konstrukcji modeli typu heurystycznego, opartych na przesłankach intuicyjnych lub pewnych opisach słownych. Przydatność tego typu modeli powinna być szczególnie starannie zweryfikowana na drodze eksperymentalnej lub przynajmniej przy zastosowaniu symulacji.

KOMPUTEROWE SYSTEMY STEROWANIA PROCESAMI DYSKRETNYMI

Jedną z istotnych cech współczesnych systemów sterowania procesami produkcyjnymi jest konieczność traktowania komputera jako jednego z elementów systemu. Należy przy tym uwzględniać podstawowe zadania, jakie komputer w systemie realizuje. Są to w szczególności:
— zbieranie i wstępne przetwarzanie danych (w tym obróbka danych, ocena przydatności i selekcja informacji, agregacja)
— przechowywanie i udostępnianie danych (bazy danych i zarządzanie tymi bazami)
— realizacja funkcji sterowania (rozpoznawanie stanów i sytuacji technologicznych, wybór modeli i ich adaptacja, dobór lokalnych wskaźników jakości, wybór optymalnych struktur technologicznych, wypracowanie sterowań dla danych struktur, modeli i wskaźników jakości).

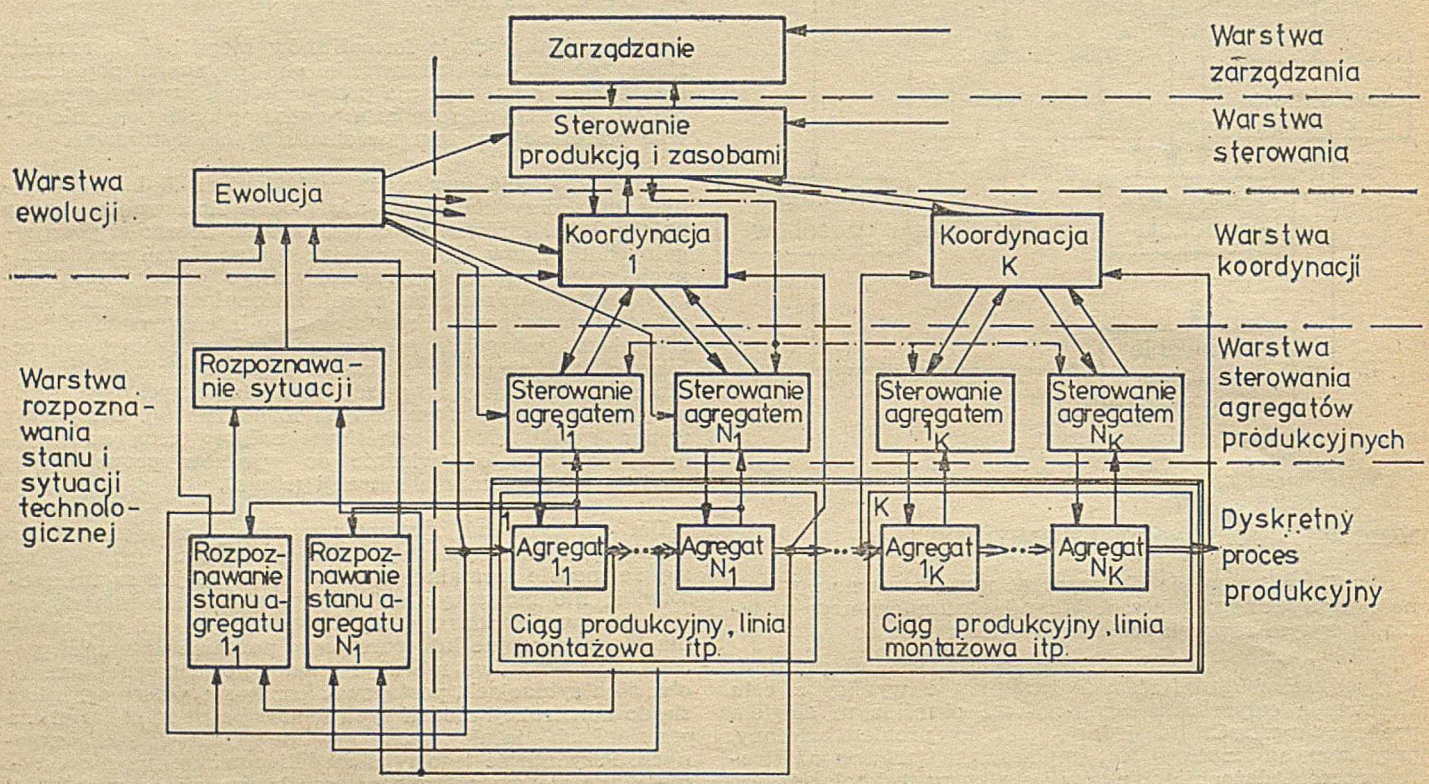
Wymienione funkcje komputera działającego w systemie sterowania złożonym dyskretnym procesem produkcyjnym, omówione zostaną z jednej strony z punktu widzenia zadań realizowanych przez system sterujący, z drugiej zaś ze zwróceniem uwagi na niektóre środki i rozwiązania stosowane w ramach samego systemu komputerowego.

Warstwowe systemy sterowania procesami dyskretnymi

Struktura systemu sterowania procesem dyskretnym jest z reguły hierarchiczna i w najogólniejszym przypadku odpowiada strukturze technologiczno-organizacyjnej procesu. Celowe jest wyróżnienie w ramach tej struktury następujących warstw funkcjonalnych systemu: 1) warstwy sterowania agregatów produkcyjnych (maszyn, urządzeń, manipulatorów oraz napędów). W ramach tej warstwy dokonuje się wyboru chwil i urządzeń, które należy włączyć, wyłączyć, zablokować, czy też zabezpieczyć. Stosowane w tej warstwie modele muszą być dokładne i winny być zrealizowane w oparciu o logikę działania obiektu sterowania. Algorytmy sterowania wykorzystywane w tej warstwie są przeważnie typu binarnego, kombinacyjnego lub sekwencyjnego. Niejednokrotnie wykorzystywane są pomocniczo algorytmy szeregowania i wyboru
2) warstwy koordynacji, która może realizować zadania:
— szeregowania
— równoważenia linii montażowej
— rozlokowania i przydziałów prac
— koordynacji współdziałania agregatów produkcyjnych (maszyn, oprzyrządowania pomocniczego i środków transportu).

W warstwie tej stosuje się modele formalne, symulacyjne lub heurystyczne, mające za zadanie opisać powiązania i współdziałanie wszsytkich elementów w ciągach produkcyjnych oraz pomiędzy tymi ciągami. Stosuje się tu algorytmy decyzyjne, oparte na opisach formalnych, a także algorytmy o charakterze heurystycznym. Nie wnikając głębiej w istotę tych algorytmów, jako najbardziej reprezentatywne można uznać:
— deterministyczne i probabilistyczne algorytmy szeregowania
— deterministyczne i probabilistyczne algorytmy równoważenia linii montażowej
— algorytmy planowania kalendarzowego (często heurystyczne) do sterowania przepływem materiałów (sterowanie wraz z koordynacją)
— algorytmy rozmyte koordynacji współdziałania
3) warstwy sterowania operatywnego. Zadaniem tej warstwy jest wypracowanie decyzji dla sterowania produkcją i zasobami. Stosowane są w tym przypadku przybliżone modele statyczne oraz modele heurystyczne, które stanowią podstawę zarówno dla wykorzystania deterministycznych i probabilistycznych algorytmów sterowania produkcją i zapasami, jak też algorytmów heurystycznych
4) warstwy zarządzania. Jej zadaniem jest opracowanie optymalnych planów produkcyjnych, planów remontów, modernizacji, rozbudowy, zakupów itp. W tej warstwie stosowane są modele różnych klas — przybliżone i dokładne, niejednokrotnie typu forresterowskiego. Algorytmy sterowania dotyczą: minimalizacji kosztu, wyrównywania zapotrzebowania na zasoby, rozdziału ograniczonych zasobów itp.

Należy ponadto zwrócić uwagę na jeszcze dwie warstwy, które mają charakter bardziej organizacyjny. Dotyczy to przypadku sterowania złożonym procesem, który może przebiegać w różnych sytuacjach technologicznych, zaś poszczególne jego obiekty mogą znajdować się w różnych stanach. Konieczne jest wówczas stosowanie:
— warstwy rozpoznawania stanu i sytuacji technologicznej. Przewiduje się, że najczęściej wykorzystywane będą tutaj algorytmy badania przynależności do charakterystycznych zbiorów parametrów (ze względu na prostotę i dostateczną efektywność tych algorytmów);
— warstwy ewolucji. Zadaniem jej jest wybór optymalnej struktury ciągu technologicznego oraz optymalnych lokalnych wskaźników jakości. Warstwa ta opiera się na symulacyjnych lub heurystycznych modelach złożonego procesu i globalnego wskaźnika jakości. Strukturę powiązań pomiędzy opisanymi warstwami przedstawiono w postaci schematu blokowego na rysunku 2.



Rys. 2. Struktura warstwowa systemu sterowania dyskretnym procesem produkcyjnym

Nie wchodząc w szczegóły związane z systemami komputerowymi stosowanymi do sterowania procesów dyskretnych, celowe jest podkreślenie tych aspektów, które wydają się znamienne dla danej klasy systemów sterowania.

Pierwszym z nich jest problem konfiguracji proces—komputer—człowiek. Otóż można stwierdzić, że pierwsza warstwa omówionej wcześniej struktury funkcjonalnej pracuje w konfiguracji on-line, podczas gdy dwie następne — wymagające działania komputera w czasie rzeczywistym — wykorzystują konfigurację *in-line*, w której przy przesyłaniu sterowań (tj. na drodze komputer—proces) w sposób efektywny uczestniczy dyspozytor. Warstwa zarządzania jest z powodzeniem realizowana w konfiguracji *off-line*.

Struktura systemów komputerowych do sterowania procesami dyskretnymi, narzuca się w pewnym sensie po analizie struktury funkcjonalnej; dotyczy to zwłaszcza realizowanych w jej ramach zadań sterowania i przekazywania informacji. Należy tu także uwzględnić aktualne tendencje w zakresie produkcji sprzętu komputerowego. Z reguły jest to struktura o charakterze rozproszonym i hierarchicznym. Wydaje się przy tym celowe uwzględnienie następujących trzech zasadniczych poziomów hierarchii:

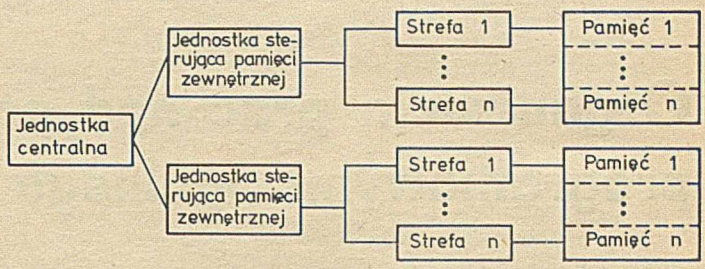
— poziomu pierwszego najniższego, stanowiącego rozproszony system mikrokomputerowy, którego poszczególne elementy realizują procesy sterowania o charakterze binarnym, kombinacyjnym lub sekwencyjnym (dla prostych ciągów sekwencyjnych) oraz sprzężony z nim system minikomputerowy realizujący bardziej złożone procesy sterowania sekwencyjnego

— poziomu drugiego realizującego w strukturze systemu informatycznego zadania drugiej i trzeciej warstwy struktury funkcjonalnej systemu sterowania. Celowe jest tu wykorzystanie minikomputerów, wyposażonych w rozbudowany zestaw urządzeń peryferyjnych dla komunikacji z człowiekiem

— poziomu trzeciego, który realizuje pracę w trybie off-line, oparty jest na minikomputerach zorientowanych na przetwarzanie danych, a w miarę potrzeb również na dużych komputerach.

W tak określonej strukturze sprzętowej systemu komputerowego należy rozwiązać problem baz danych. Wydaje się, że najkorzystniejszą będzie rozproszona baza danych, z tym, że jej podział będzie zależał od przyjętego wskaźnika niezawodności systemu komputerowego oraz przyjętych w projekcie rozwiązań.

Opisana struktura systemu komputerowego wraz z zaznaczeniem funkcji i trybu pracy poszczególnych poziomów, przedstawiona została schematycznie na rysunku 3.



Rys. 3. Struktura systemu komputerowego

Kolejną kwestią jaką należy rozstrzygnąć, jest odpowiednia konstrukcja systemów operacyjnych oraz dobór języków programowania. Przy tworzeniu systemów operacyjnych należy przewidywać częściowe wykorzystanie specjalizowanych systemów operacyjnych czasu rzeczywistego. Dotyczy to przede wszystkim mikrokomputerów oraz minikomputerów działających na pierwszym poziomie struktury sprzętowej. Wybierając język programowania trzeba pamiętać, że oprócz języków asemblerowych wykorzystywane są w szczególności:

— języki problemowo zorientowane do sterowania binarnego i kombinacyjnego

— języki problemowo zorientowane do sterowania sekwencyjnego.

Ze względu na niezbyt ostre wymagania czasowe, na drugim poziomie struktury sprzętowej wykorzystane być mogą z powodzeniem języki zorientowane proceduralnie, w tym najbardziej rozpowszechnione RT FORTRAN oraz CORAL 66. Ta ostatnia klasa języków oraz języki symulacyjne wykorzystywane są w trzeciej warstwie, ponieważ zadania tej warstwy nie są uwarunkowane czasowo.

STANISŁAW MROZIK
Centrum Projektowania i Zastosowań Informatyki ZETO
Warszawa

Organizacja prac nad produktami programowymi ogólnego przeznaczenia

W połowie 1974 r. w Zakładzie Elektronicznej Techniki Obliczeniowej w Warszawie (ZOWAR) opracowano koncepcję uruchomienia nowej formy działania w zakresie usług informatycznych, wzorowaną na zagranicznych wyspecjalizowanych przedsiębiorstwach produkcji oprogramowania (*ang. software houses*).

Koncepcja określała zarówno rodzaj produktów programowanych, jak i zasady organizacyjne ich wytwarzania i upowszechniania. Kierownictwo Zjednoczenia Informatyki zaakceptowało propozycję i skierowało ją do realizacji w Zakładzie Produkcji Oprogramowania RPOLSYSTEM ZOWARU. Od 1975 r. prace były kontynuowane w Ośrodku Badawczo-Rozwojowym Informatyki.

Prace te koncentrowały się wokół następujących trzech grup tematycznych:

1) systemy zarządzania bazami danych

2) oprogramowanie podstawowe, ukierunkowane na obsługę procesu programowania

3) narzędzia programowe, usprawniające proces opracowania oprogramowania użytkowego.

W efekcie opracowano następujące produkty programowe:

- system zarządzania bazą danych RODAN (zob. J. Pasula: Uniwersalny System Zarządzania Bazą Danych RODAN. INFORMATYKA nr 9/78)
- system zarządzania bazą danych SYKON (zob. A. Wacławik: SYKON — system zarządzania bazą danych. INFORMATYKA nr 10/78)
- system sterowania techniczno-ekonomicznego produkcją STEP (zob. M. Klimek, A. Sakowski: STEP — realizacja nowych idei w systemie sterowania produkcją. INFORMATYKA nr 11/77 — oraz A. Klimek: Szybkie wejście w STEP. INFORMATYKA nr 4/78)
- technologiczne wersje systemów operacyjnych DOS oraz OS z własnym oprogramowaniem zarządzania procesem programowania (automatyczne opracowywanie dokumentacji)
- język funkcjonalny wykonywania obliczeń i wyprowadzania wyników JAZ 75 (zob.: Pakiet programowy JAZ 75. INFORMATYKA nr 7—8/78 oraz F. Surmiak: Z doświadczeń eksploatacyjnych pakietu JAZ 75. INFORMATYKA nr 12/78)
- generator macierzy zadania programowania liniowego LPSERWIS (zob.: A. Gens: LPSERWIS — oprogramowanie do rozwiązywania zadań programowania liniowego. INFORMATYKA nr 1/79).

Najlepszym miernikiem oceny wspomnianych osiągnięć jest liczba użytkowników wykorzystujących poszczególne produkty: RODAN — 15, STEP — 24, SYKON — 4, technologiczne wersje systemu operacyjnego DOS — 10 i OS — 12, JAZ 75 — 33 oraz LPSERWIS — 3.

Zważywszy więc, że koncepcję nowej formy działalności realizuje się zaledwie piąty rok, przytoczone wyniki można uznać za sukces potwierdzający słuszność przyjętych zasad organizacyjnych. Omówieniu tych zasad poświęcony jest niniejszy artykuł.

ETAPOWANIE PRAC

Uniwersalny charakter produktów programowych sprawia, że nie są one już tylko opracowaniem technologii eksploatacji systemu komputerowego, która polega na zakodowaniu ściśle zdefiniowanych (można powiedzieć „sfotografowanych") algorytmów przetwarzania celem „wyprodukowania" ściśle zdefiniowanych tabulogramów.

Wykonanie produktów programowych ogólnego przeznaczenia ukierunkowanych na obsługę określonych funkcji w sposób niezależny od konkretnego indywidualnego zastosowania, wymaga znacznie większych środków niż opracowanie zastosowania. Wysoki koszt realizacji oprogramowania funkcjonalnego rekompensowany jest jednak wielokrotnym jego wykorzystaniem dla wielu zastosowań, a także niewątpliwie lepszą jego jakością.

Organizację prac nad realizacją produktów programowych oparto na zasadach stosowanych w pracach badawczych, a ściślej w badaniach skierowanych na określony cel użytkowy.

Zakładają one nie tylko wykonanie projektu końcowego, ale również jego wdrożenie. Ich rozpoczęcie wymaga jasnego i jednoznacznego sprecyzowania celów użytkowych stawianych przed wynikami końcowymi prac. Prace realizowane są etapami.

Struktura badań skierowanych, do których zaliczyliśmy prace nad produktami programowymi, wynika z etapowania prac i może być następująca:

- badania rozpoznawcze—badania modelowe—badania techniczne — opracowanie projektu końcowego i wdrożenie (struktura przyjęta dla realizacji SZBD RODAN)
- badania modelowe — badania techniczne — opracowanie projektu końcowego i wdrożenie (struktura przyjęta dla prac nad systemami STEP, SYKON, LPSERWIS)
- badania techniczne — opracowanie projektu końcowego i wdrożenie (struktura przyjęta dla prac nad produktem JAZ 75 i technologicznymi wersjami systemów operacyjnych).

Cele i wyniki prac realizowanych w poszczególnych etapach powinny być następujące:

Badania rozpoznawcze

Ich celem jest wykazanie realności opracowanej koncepcji oprogramowania poprzez badanie poszczególnych jego składników, a zwłaszcza elementów o decydującym znaczeniu.

W wyniku tych badań powinna powstać ideowa architektura oprogramowania. Koncepcja taka nie musi określać sposobu przetwarzania danych, powinna natomiast określać funkcje poszczególnych elementów oprogramowania oraz ich wzajemne powiązania. Powinna również przewidywać sposób realizacji elementów krytycznych. Wynik końcowy badań rozpoznawczych przyjmuje postać projektu architektury systemu. Prawdopodobieństwo, że praca rozpoczęta na tym etapie zakończy się sukcesem wynosi zaledwie 10—30% i trwa zwykle 3—4 lata.

W przypadku omawianych prac przeprowadzono jedynie częściowe badania rozpoznawcze, które trwały zaledwie od 2 do 4 miesięcy.

● **Badania modelowe**

Ich celem jest sprawdzenie wersji eksperymentalnej oprogramowania (jednej lub wielu). Wersja eksperymentalna dopuszcza wykonanie systemu z elementów zapożyczonych z innych rozwiązań, symulowanie pewnych elementów, a nawet pominięcie tych elementów, co do których istnieje pewność poprawnego funkcjonowania w warunkach eksploatacyjnych. Obowiązkowe jest jedynie stworzenie modelu i dokonanie testów sprawdzających jego funkcjonowanie. Końcową formą badań modelowych jest prototyp oprogramowania. Dla prac rozpoczynanych na tym etapie prawdopodobieństwo sukcesu końcowego wzrasta do 40—60%, przy czym trwają one najwyżej 2 lata.

● **Badania techniczne**

Celem jest tu sprawdzenie w pełnym zakresie walorów eksploatacyjnych wykonanego prototypu oprogramowania. Realizacja prac następuje poprzez przekształcenie prototypu oprogramowania w wersję eksploatacyjną w wyniku sukcesywnego usuwania błędów wykrytych w trakcie wszechstronnej eksploatacji próbnej. Warunki eksploatacji powinny być trudniejsze od przeciętnych.

Zakończenie trwającego średnio 1,5—2 lat etapu badań technicznych w sposób radykalny zwiększa prawdopodobieństwo sukcesu końcowego, które wynosi już około 90%.

● **Opracowanie projektu końcowego oraz wdrożenie**

Ostatni etap prac ma charakter operacyjny. Na etapie tym powinna powstać kompletna dokumentacja produktu oraz materiały szkoleniowe dotyczące jego użytkowania. W oparciu o powyższą dokumentację powinno również nastąpić zastosowanie oraz wdrożenie eksploatacyjne produktu u co najmniej dwu użytkowników.

Prace nad zastosowaniem i wdrożeniem systemu użytkowego, w trakcie których wykorzystuje się produkt końcowy badań, obejmują kolejne fazy wynikające z przyjętej metodyki ich prowadzenia.

Zwróćmy uwagę, że stosując omawianą organizację pracy napisanie programów nie jest celem określonego etapu, lecz stanowi wyłącznie czynność techniczną, wykonywaną praktycznie na każdym etapie, zaś komputer jest jedynie narzędziem realizacji przedsięwzięcia badawczego. Również dokumentacja techniczna produktu nie stanowi celu pracy, jest jedynie niezbędnym narzędziem wzajemnego porozumienia się zespołu i ułatwieniem przy wykrywaniu i usuwaniu błędów. Dokumentacja ta składa się najczęściej z tekstów źródłowych programów opatrzonych komentarzami. W przypadku prac nad SZBD RODAN dokumentacja taka była prowadzona automatycznie przez odpowiednią wersję technologiczną systemu operacyjnego OS. Zgodnie z powyższą zasadą niektóre z prac polegających na kodowaniu programów, a wykonanych już na etapie badań rozpoznawczych, mogą zostać włączone bezpośrednio do produktu końcowego.

WZAJEMNE ZALEŻNOŚCI I UWARUNKOWANIA

Poniżej zostaną przedstawione niektóre zależności zaobserwowane podczas prac badawczych nad produktami programowymi. Zależności te wymagają stosowania określonych zasad prowadzenia prac, które zwiększają ich efektywność oraz pewność osiągnięcia sukcesu końcowego.

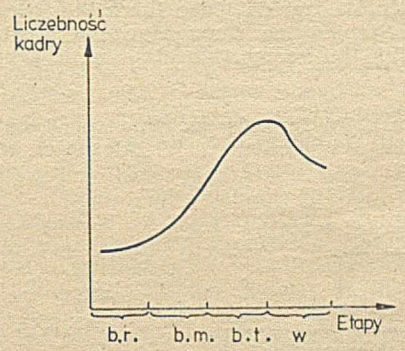
1. Podczas realizacji projektu oprogramowania ilość wykonywanych prac rośnie w miarę gdy posuwamy się od jego zakończenia do rozpoczęcia, tzn. od wdrożenia poprzez badania techniczne, badania modelowe do badań rozpoznawczych. Oznacza to, że jednemu zadaniu końcowemu odpowiada znaczna liczba prac wykonanych na poprzednich etapach, przy czym liczba ta zależy w sposób istotny od liczby elementów projektu, a także od konieczności równoległego prowadzenia różnych wariantów rozwiązań (w przypadkach gdy powodzenie ich realizacji nie jest z góry pewne). Przykładowo, w celu wykonania jednego SZBD RODAN, na etapie badań modelowych przeanalizowano kilkanaście różnych systemów, a na etapie badań technicznych zrealizowano dwie wersje prototypu. Proporcje te często kształtują się następująco:

- wdrażanie — 1 praca
- badania techniczne — 1 do 3 prac
- badania modelowe — 10 do 30 prac.

Należy podkreślić, że poszczególne elementy projektu muszą przejść przez wszystkie etapy prac, i w momencie gdy jeden z elementów jest już na ostatnim etapie, inny może znajdować się dopiero na etapie badań rozpoznawczych. W praktyce prowadzi to do udostępniania produktu końcowego w kolejnych jego wersjach. Np. podczas opracowania produktu STEP udostępniano kolejne wersje STEP 74, STEP 75, STEP 76. Przy systemie RODAN udostępniono wersję W 1 — lokalną jednozadaniową, następnie W2 — rozszerzoną w stosunku do poprzedniej o możliwość równoczesnego przetwarzania wielu zadań, by w końcu udostępnić wersję W3 — rozszerzoną w stosunku do poprzedniej o możliwość zdalnego dostępu.

Operatywne kierowanie projektem polega m.in. na tym, by zharmonizować całość prac, zwracając szczególną uwagę na elementy krytyczne projektu. Oczywiście chodzi tu o zapewnienie w projekcie współpracy elementów nowo tworzonych ze starymi, wypróbowanymi już w innych systemach lub w innych wersjach tego projektu.

2. Praktyka wskazuje, że odwrotnie do malejącej ilości prac, przejście do każdego następnego, bardziej zaawansowanego etapu realizacji pociąga za sobą zaangażowanie środków materialnych o rząd większych, co powoduje, że koszty realizacji kolejnych etapów mają się do siebie jak 1 : 10 : 100 : 1000. Szczególnie odpowiedzialne są decyzje przejścia z etapu badań modelowych do badań technicznych. Trudno jest bowiem zrezygnować z kontynuowania prac nad projektem, w który dużo już zainwestowano, uzyskano prototyp, a osiągnięcie celu końcowego zespołowi autorskiemu wydaje się całkowicie pewne.



Rys. 1. Zapotrzebowanie na kadrę w toku prac nad produktami programowymi: b.r. — badanie rozpoznawcze, b.m. — badanie modelowe, b.t. — badanie techniczne, w — wdrożenie

3. Zapotrzebowanie na kadrę w poszczególnych etapach prac kształtuje się w sposób pokazany na rys. 1. Prawidłowe ukształtowanie obsady kadrowej jest jednym z trudniejszych zadań kierownictwa projektu, wymaga bowiem znacznego zróżnicowania liczebności i kwalifikacji zespołów w kolejnych etapach. W praktyce liczebność zespołów kształtowała się następująco:

- dla SZBD RODAN: badania rozpoznawcze — 6, badania modelowe — 12, badania techniczne — 25, wdrażanie — 15,
- dla systemu STEP: badania modelowe — 5, badania techniczne — 17, wdrażanie — 8.

4. Jedną z konsekwencji nieokreśloności prac badawczych (prawdopodobieństwa uzyskania sukcesu) jest m.in. potrzeba dublowania prac. Jednak wbrew pozorom dublowanie takie w ostatecznym rachunku prowadzi do oszczędności środków i czasu. Przy czym liczba dróg rozwiązań musi być tym większa, im większa jest doniosłość problemu, im większy stopień nieokreśloności, im większe są możliwości zastosowania różnych metod i sposobów programowania.

5. Rezultaty prac prowadzonych na etapach badań modelowych, technicznych i wdrożenia mają duży wpływ na rozwiązania innych prowadzonych równolegle prac. Np. wyniki prac nad systemem operacyjnym OS i nad systemem STEP, miały wpływ na SZBD RODAN. W szczególności dotyczyły one problemów wielozadaniowości jeżeli chodzi o OS oraz udogodnień związanych z zastosowaniem dla potrzeb planowania produkcji jeżeli chodzi o STEP.

ZASADY PROWADZENIA PRAC

Uzyskane doświadczenia pozwalają ustalić następujące, moim zdaniem najbardziej skuteczne zasady prowadzenia prac.

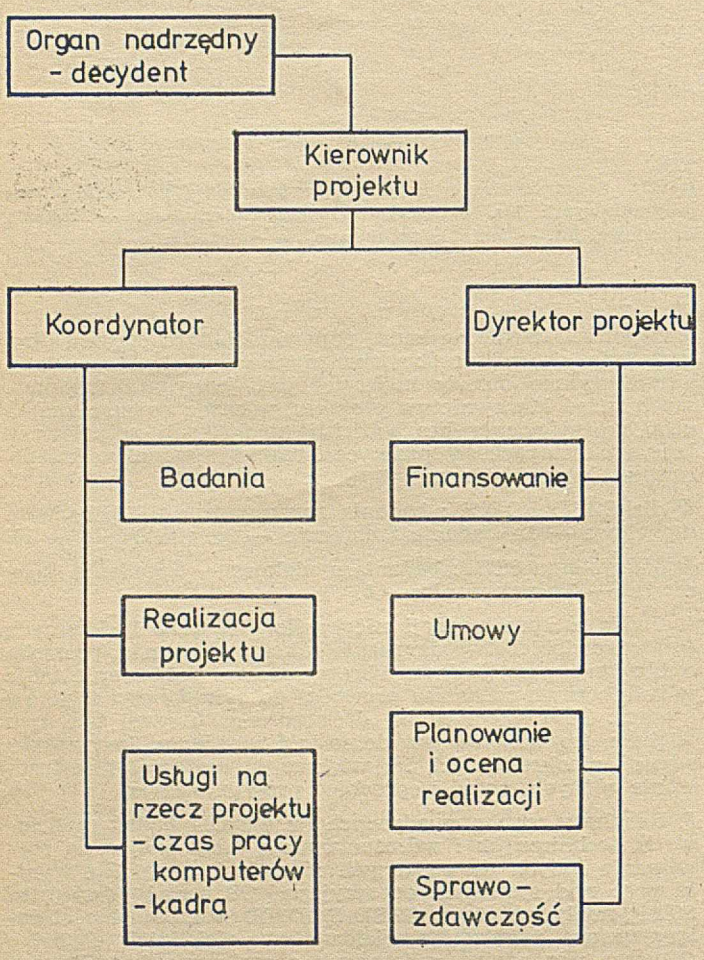
1. Każdy projekt powinien być kierowany przez mały zespół. Kierownik projektu musi mieć pełnomocnictwa umożliwiające synchronizację prac w wielu komórkach (instytucjach) współdziałających w wykonaniu projektu. Powinien być on podporządkowany bezpośrednio szczeblowi, na którym podejmuje się decyzje związane z zakończeniem i rozpoczęciem kolejnych etapów pracy badawczej.

Warunkiem właściwej pracy kierownika projektu jest posiadanie przez niego pełnomocnictwa w zakresie doboru kadr oraz jasne sprecyzowanie jego odpowiedzialności wobec zwierzchników. Muszą również istnieć określone terminy oceny pracy oraz osoba, która w razie konieczności będzie mogła przejąć kierownictwo prac.

2. Podjęcie prac projektowych odbywa się po decyzji szczebla nadrzędnego w stosunku do kierownika projektu i powinno być poprzedzone analizą celów. Zadaniem kierownictwa projektu jest sporządzenie dla każdego etapu sieci PERT, stanowiącej podstawę operatywnego kierowania pracami. Odbiór każdego etapu, a także decyzje związane z przekraczaniem limitów czasowych i finansowych należą do wspomnianego szczebla nadrzędnego, który decyduje także o podjęciu prac następnego etapu. Kierownik projektu może wnioskować o przejście do następnego etapu, mimo że niektóre podsystemy wchodzące w skład projektu są opóźnione. W takim przypadku decyzja szczebla nadrzędnego obciążona jest oczywiście wyższym stopniem ryzyka.

W każdym projekcie istnieje zwykle jeden lub kilka „kluczowych" podsystemów. Stopień ich zaawansowania powinien decydować o przejściu do następnego etapu prac projektowych.

3. Kierownictwo projektu pełni funkcję zarówno koordynatora prac, jak i zbiorowego wykonawcy. W zależności od rodzaju etapu ustala się sposób i częstotliwość składania sprawozdań, a także standardową, maksymalnie prostą formę służących do tego celu dokumentów.



Rys. 2. Schemat funkcjonalny kierownictwa projektu

4. Zaleca się pionową strukturę organizacji prac, która polega na tym, że kierownictwo projektu:
— dysponuje podległym mu aparatem, odpowiedzialnym tylko przed nim za całokształt uzyskanych wyników
— dysponuje funduszami przeznaczonymi na projekt
— ma bezpośrednie powiązanie z kierownictwem szczebla naczelnego.

5. Z punktu widzenia skuteczności prowadzonych prac i trafności przyjętych rozwiązań kierownik projektu spełnia rolę kluczową, a dyrektor powinien zapewnić mu warunki efektywnego działania.

Kierownik projektu nie tylko podejmuje wszelkie decyzje operacyjne, ale również reprezentuje zespół na zewnątrz i samodzielnie prowadzi politykę kadrową, natomiast Dyrektor pełni jedynie funkcje administracyjne. Stworzenie układu funkcjonalnego, który w sposób przykładowy ilustruje rysunek 2, jest jednym z najtrudniejszych, bo niezgodnych z dotychczasową praktyką problemów właściwej organizacji prac projektowych.

6. Z praktyki wynika, że w wielu przypadkach kierownikiem projektu może być osoba nie reprezentująca najwyższego poziomu specjalistycznych kwalifikacji zawodowych, pod warunkiem jednak, że ma odpowiednich doradców. Uzasadnieniem takiego rozwiązania jest fakt, że na organy kierujące spada większość prac nie wymagających specjalistycznych kwalifikacji zawodowych.

Niższe ogniwa organizacyjne zespołu projektowego powinny być maksymalnie odciążone od wszelkich czynności o charakterze administracyjnym.

7. Jak wiadomo, nieodzowne jest wprowadzanie do projektów zmian. Pociąga to za sobą oczywiście zmianę harmonogramów realizacji oraz wzrost kosztów. Każda zmiana, która pociąga za sobą opóźnienie wykonania zadania, musi być zaakceptowana na szczeblu nadrzędnym. Krzywa na rysunku 3 ilustruje kształtowanie się liczby proponowanych do wniesienia zmian oraz wskazuje, do jakich ograniczeń przy ich wprowadzaniu należy dążyć.

8. Organy decydujące o kontynuowaniu prac powinny liczyć się z ewentualnym niepowodzeniem pierwszych prób. Sytuacja taka jest często wynikiem trudności obiektywnych, nie zaś nieudolności kierownictwa projektu.

9. Kierując pracami według określonego i ścisłego harmonogramu należy też liczyć się z możliwością wystąpienia konfliktów:

— ze strony potencjalnych odbiorców, wymagających idealnego produktu, który jednak początkowo zawierać musi jeszcze wiele ukrytych wad

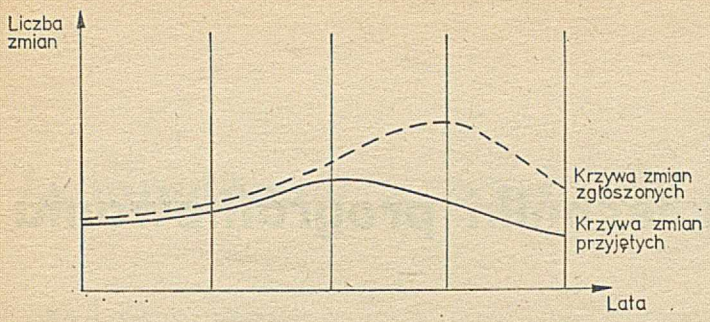
— sprzeczności poglądów z organami finansującymi prace

— ze strony „geniuszy" rozwiązań technicznych, którzy stale dążą do rozwiązań „przodujących", ale powodujących w praktyce znaczny wzrost kosztów i pracochłonności projektu

— ze strony personelu technicznego (wykonawczego), który jest prawie zawsze niezadowolony z pracy projektanta i analityka.

10. Powodzenie prac zależy w znacznym stopniu od osobistego zaangażowania realizatorów. Czynnikiem pobudzającym takie zaangażowanie jest bieżąca informacja o aktualnym stanie zaawansowania prac. Członkowie zespołu muszą jasno zdawać sobie sprawę z podstawowego celu pracy oraz swojej roli w realizacji tego celu.

11. Czynnikiem najbardziej demoralizującym i łamiącym zaangażowanie każdego kolektywu jest poczucie, że cel jest ciągle zmieniany, niepewny. Sytuacja ta staje się szczególnie niebezpieczna, gdy w przypadku zakończenia lub przerwania pracy nie następuje natychmiastowe sprecyzowanie nowego zadania.



Rys. 3. Przebieg zmian projektu w czasie

12. Wewnętrzna struktura organizacyjna zespołów wykonawczych powinna być niesformalizowana. Należy przyjąć zasadę bezpośrednich kontaktów i pełnej odpowiedzialności wobec kierownika projektu.

13. Pomimo bardzo ścisłego planowania projektów oprogramowania, rezultaty uboczne prac są często nie mniej cenne niż wyniki podstawowych zamierzeń.

SPOSÓB PROWADZENIA PRAC

Z przedstawionych powyżej zasad wynika szereg zaleceń dotyczących sposobu prowadzenia prac nad oprogramowaniem.

Podstawowe znaczenie mają warunki i miejsce podjęcia decyzji o przystąpieniu do prac oraz decyzji o zakończeniu poszczególnych etapów i przejściu do następnych etapów badań. Należy wyraźnie rozgraniczyć przy tym rolę decydenta i zespołu realizującego.

Jak już podkreślono, podjęcie decyzji o przystąpieniu do prac nad oprogramowaniem powinno opierać się na wnioskach wynikających z wszechstronnej analizy celowości rozpoczęcia prac nad konkretnym produktem. Wyniki takiej analizy powinny uzasadnić, czy biorąc pod uwagę sprawdzone w praktyce efekty zastosowań celowe jest:

— stosowanie już istniejącego oprogramowania, które eliminuje zasadność prowadzenia prac nad nowym produktem programowym

— sprawdzenie czy istniejące oprogramowanie nie nadaje się do bezpośredniego wykorzystania i wymaga przeprowadzenia badań technicznych, badań modelowych lub nawet badań rozpoznawczych.

Analiza powinna również proponować uzasadnione terminy zakończenia etapów, a w szczególności przewidywany termin oddania systemu użytkownikowi lub zakończenia badań technicznych. Ustalenia te wiążą się również z decyzją, czy wykonawca pracy badawczej weźmie udział w etapie wdrażania.

Miejsce podejmowania decyzji o prowadzeniu prac nad oprogramowaniem wiąże się ze stopniem uniwersalności oprogramowania. Dla przykładu, miejscem podejmowania decyzji o rozpoczęciu prac nad oprogramowaniem systemu przetwarzania danych dla określonego użytkownika i określonego wycinka jego systemu informacyjnego jest np. dyrekcja przedsiębiorstwa. Miejscem zadecydowania o opracowaniu podobnego systemu, lecz o charakterze powielarnym, powinna być instytucja mająca informacje, które pozwalają ocenić spodziewane rezultaty z punktu widzenia celów ogólnospołecznych i ekonomicznych, przekraczających horyzont jednego przedsiębiorstwa.

Przy obecnej skali zastosowań informatyki i na obecnym poziomie rozwoju technologii zastosowań, bardzo często podejmowanie decyzji dotyczącej uruchomienia prac nad oprogramowaniem uniwersalnym na szczeblu bezpośredniego użytkownika jest nieracjonalne z punktu widzenia polityki ekonomicznej i społecznej w skali kraju.

W przypadku oprogramowania o charakterze uniwersalnym, decyzje takie powinny uwzględniać jak najszerszy — ponadbranżowy — zasięg stosowania produktu programowego.

* * *

Postarajmy się zreasumować, jakie nowe elementy wprowadza proponowane podejście do prowadzenia prac i jakich należy z tego tytułu oczekiwać korzyści?

Podstawowym zarzutem w odniesieniu do stosowanego obecnie, tradycyjnego etapowania prac nad oprogramowaniem jest odwlekanie momentu przeprowadzenia decydujących doświadczeń. Zdaje to egzamin wówczas, gdy rozwiązanie problemu wymaga jedynie opracowania technologii i wdrożenia.

Podejście takie w odniesieniu do prac nad produktami programowymi ogólnego przeznaczenia zwiększa w sposób nieuzasadniony ryzyko końcowego niepowodzenia. W przypadku takich produktów jedynie pozytywne wyniki realizacji wcześniejszego etapu prac mogą być przepustką do angażowania dalszych środków na kontynuowanie pracy. Praktycznie już na etapie badań modelowych, podstawowym celem jest przeprowadzenie doświadczenia (poprzez testowanie), które wykaże, czy model oprogramowania jest realny.

Stosując tradycyjne etapowanie prac nad oprogramowaniem, dopiero eksploatacja próbna — jako decydujące doświadczenie — może wykazać, że poniesione nakłady nie dały spodziewanych rezultatów, natomiast niezbędne zmiany założeń pociągają za sobą konieczność dokonania istotnych zmian całego modelu, a więc bardzo poważnego zwiększenia kosztów i przedłużenia terminów realizacji. Z praktyki tradycyjnego etapowania wiadomo bowiem, jak często trudne lub wręcz niemożliwe jest dotrzymanie planowanych terminów i jak trudno zmieścić się w ramach wstępnie ustalonych kosztów realizacji oprogramowania.

Warto jeszcze podkreślić, że znacznie większe od bezpośrednich korzyści ekonomicznych są korzyści polegające na zdecydowanym podniesieniu jakości oprogramowania oraz nowoczesności przyjmowanych rozwiązań w wyniku bardziej wnikliwego analizowania i wykorzystywania rezultatów prac wykonanych przez inne zespoły — zarówno w kraju, jak i za granicą.

Ku nowoczesnej informacji

Usprawnienie procesów gromadzenia i udostępnienia informacji naukowej, technicznej i ekonomicznej wiąże się z ich automatyzacją. Wiadomo, że wprowadzanie automatyzacji przebiegało jak dotychczas dość opieszale. Jedną z przyczyn była niedoskonałość większości opracowanych w kraju systemów dla informacji naukowej, technicznej i ekonomicznej. Ostatnio na podstawie umowy z UNESCO Instytut Informacji Naukowej, Technicznej i Ekonomicznej (Instytut prowadzi wszystkie badania związane z realizacją państwowego systemu SINTO) otrzymał pakiet ISIS. Pakiet ten, opracowany na komputery IBM, umożliwia objęcie prawie wszystkich tradycyjnych funkcji procesów gromadzenia, przechowywania i udostępniania informacji.

Instytut Informacji Naukowej, Technicznej i Ekonomicznej dokonał adaptacji systemu na komputery R-32. Obecnie, po zawarciu pierwszych umów, rozpoczyna się w dniu 1 października br. w Warszawie szkolenie przedstawicieli 10 instytucji z terenu całego kraju. 3-tygodniowe szkolenie ma zapewnić przyszłym użytkownikom sprawną implementację ISIS. Szkolenie organizuje i prowadzi Zespół Systemów INTE.

ZDZISŁAW SZYJEWSKI
Zakład Organizacji Przetwarzania Danych
Politechnika Szczecińska

Niektóre techniki programowania

Pamięć wirtualna, dopomagając do rozwiązania niektórych starych problemów, stawia przed użytkownikami komputerów problemy nowe, dotąd im nie znane. O sprawności programu decyduje własność, którą użytkownik nigdy przedtem nie interesował się: strefa odniesienia [1]) [4].

Liczba komputerów z pamięcią wirtualną jest obecnie w Polsce niewielka i dlatego pisanie programów z uwzględnieniem stronicowania pamięci interesuje bardzo ograniczony krąg użytkowników tych instalacji. Niemniej, w miarę rozwoju maszyn tzw. drugiej generacji Jednolitego Systemu (RIAD-2) i wprowadzania na rynek krajowy komputerów tej serii, wyposażonych jak wiadomo w pamięć wirtualną, liczba osób zainteresowanych programowaniem w warunkach pamięci wirtualnej będzie szybko wzrastać.

Wiadomo jednak, że program napisany dla komputera bez możliwości stronicowania pamięci może być bez żadnych zmian eksploatowany na maszynie z pamięcią wirtualną. Nie ma więc ograniczeń formalnych, które zmuszałyby programistę do zmiany techniki pisania programu. Ale technika pisania programów, bardzo dobra w przypadku eksploatacji programu na komputerze ze zwykłą adresacją pamięci, może się okazać nieefektywna w eksploatacji programów na maszynie z pamięcią wirtualną. Pisząc programy na obecnie dostępne komputery można zatem i trzeba — przyswajać sobie nawyki programowania pamięci wirtualnej bez szkody dla jakości tych programów.

Cechą charakterystyczną programów pisanych dla komputerów nie wirtualnych jest oszczędne gospodarowanie pamięcią operacyjną. Stałe dążenie do minimalizacji zajętości pamięci operacyjnej przez program, spowodowało powstanie specyficznych technik pisania programów. Techniki te są powszechnie stosowane przez przytłaczającą większość programistów. W przypadku wprowadzenia nowego typu komputera programista uczy się wprawdzie języka programowania i poznaje specyfikę nowego sprzętu, ale zupełnie nie zmienia techniki pisania programu. Dzieje się tak tym bardziej jeśli nie zmienia się język programowania, a specyfika nowego sprzętu nie wymusza takich zmian.

W niniejszym artykule pragnę zwrócić uwagę na niektóre elementy techniki pisania programów, które zastosowane w warunkach pamięci wirtualnej spowodują, że programy takie będą efektywniej wykonywane.

CO JEST INACZEJ?

Na co należy zwrócić szczególną uwagę pisząc program, który będzie eksploatowany na komputerze z pamięcią wirtualną?

Przy zwykłej adresacji pamięci operacyjnej wykonywany program w całości znajduje się w pamięci operacyjnej maszyny. Wyjątek stanowią programy pisane techniką nakładkową, ale i w tym przypadku działanie programu odbywa się na obszarze jego części głównej i aktualnie wykonywanej nakładki. Biorąc pod uwagę możliwość dostępu do każdego adresu pamięci zajętej przez program, programista nie musi zastanawiać się nad rozmieszczeniem danych i instrukcji programowych w ramach programu.

W przypadku wykonywania programu na komputerze z pamięcią wirtualną należy pamiętać o tym, że nie cały program znajduje się w pamięci operacyjnej, ale tylko niektóre jego stronice [1].

[1]) *Strefą odniesienia* Reisman nazywa własność programu mierzoną zakresem adresów wywoływanych w danym czasie i związaną z transferami stron programu między pamięcią realną a wirtualną

Z punktu widzenia programisty dostęp do danych i instrukcji nie ulega zmianie, ale odwołania do adresów programu, które znajdują się na stronie przechowywanej w pamięci zewnętrznej, powodują, że realizacja takiego odwołania inicjuje najpierw przeniesienie strony z żądanym adresem z pamięci zewnętrznej do pamięci realnej. Aby zminimalizować liczbę przesłań stronic, należy w trakcie konstrukcji programu zastanowić się nad rozmieszczeniem danych i instrukcji w ramach programu.

Pisząc większy, bardziej skomplikowany program programista stawał przed dylematem, czy program ten zmieści się w pamięci maszyny. Często decydującym czynnikiem ograniczającym był rozmiar dostępnej pamięci komputera. W przypadku pamięci wirtualnej ograniczenie to znika, gdyż programista dysponuje praktycznie nieograniczoną pamięcią maszyny. Ta istotna zmiana nie zawsze jest efektywnie wykorzystywana z uwagi na wspomniane już nawyki oszczędzania pamięci. Tymczasem nieograniczona pojemność pamięci pozwala na wykonywanie w ramach programu np. prostych sortowań, które mogą wydatnie uprościć dalszy przebieg programu. W pamięci programu można również przechowywać i wykorzystywać np. tablicę indeksów lub inne informacje ułatwiające przetwarzanie.

Podobnie w przypadku pisania dużych programów, składających się z kilku modułów (segmentów), powstaje problem przekazywania danych. Programiści robią to najczęściej poprzez wykorzystywanie wspólnych obszarów danych. Obszar pamięci, zadeklarowany jako obszar wspólny, jest obszarem niezależnym od każdego segmentu, który z niego korzysta, ale przyjmuje się zasadę, aby w pamięci znajdowały się równocześnie obszar danego segmentu i obszar wspólny. Przy stosowaniu techniki programów nakładkowych obszar ten znajduje się zwykle w części głównej programu. Zastosowanie takiego sposobu przekazywania danych pomiędzy modułami programu przy istnieniu pamięci wirtualnej może spowodować wiele niepotrzebnych odwołań do stronic, które aktualnie znajdują się na dysku. Wskazane jest, aby segment operował jedynie na obszarach wchodzących w jego skład lub na obszarach, które znajdują się na innych stronach znajdujących się w pamięci realnej. Można to osiągnąć przez zastosowanie specjalnych zbiorów zewnętrznych, do przekazywania danych pomiędzy modułami, w miejsce stosowanych obszarów wspólnych. Wpływa to na zmniejszenie liczby przesłań stronic programu pomiędzy pamięcią realną a pamięcią zewnętrzną.

Podobny problem występuje w przekazywaniu danych do wywoływanego podprogramu. Często w instrukcji wywołującej podprogram podaje się adres zmiennej, która zawiera dane lub parametry niezbędne dla wykonania tego podprogramu. Znacznie bardziej efektywne rozwiązanie w warunkach stosowania pamięci wirtualnej polega na jawnym określaniu danych lub parametrów w momencie wywołania podprogramu.

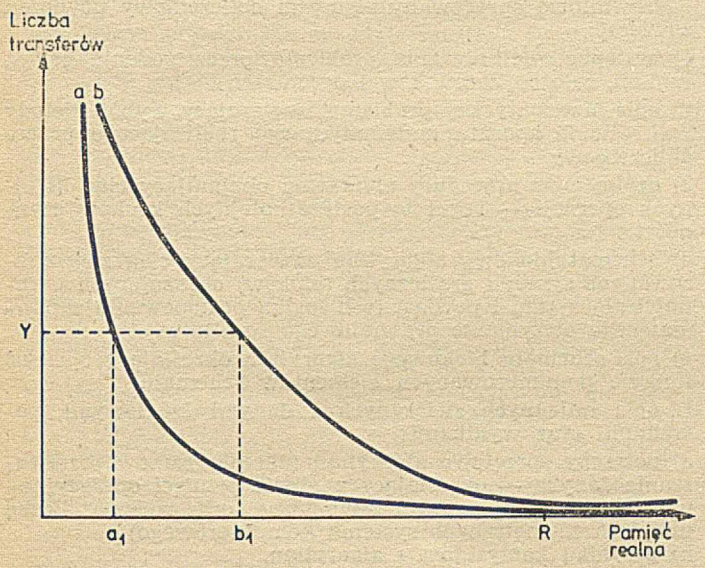
Z tych kilku prostych przykładów różnego podejścia do pisania programu w zależności od sposobu adresowania pamięci komputera widać, że podstawowym problemem w programach pisanych z uwzględnieniem pamięci wirtualnej staje się minimalizacja liczby przesłań stronic programu pomiędzy pamięcią zewnętrzną a pamięcią realną. Wymóg ten staje się wytyczną w stosowaniu wszystkich znanych technik i metod programowania.

Dla zilustrowania tego problemu porównajmy dwa programy tej samej klasy napisane przy użyciu odmiennych technik programowania na maszynę wirtualną. Jako wielkości charakteryzujące te programy rozważmy:
— pojemność dostępnej pamięci realnej oraz
— liczbę przesłań pomiędzy pamięcią realną a pamięcią zewnętrzną [3].

komputerów z pamięcią wirtualną

Na rysunku pokazano przykładowe charakterystyki tych programów w postaci wykresów.

Zakładając, że Y jest liczbą przesłań potrzebnych do wykonania programów **a** i **b**, z wykresu wynika, że rozmiar niezbędny dla realizacji programu pamięci realnej wynosi a_1, zaś dla realizacji programu **b**—b_1, oraz że zachodzi relacja $a_1 < b_1$. Aby wykonać oba programy przy takiej samej liczbie odwołań do pamięci zewnętrznej, dla programu **b** wymagany jest większy obszar realnej pamięci zajmowanej przez ten program niż w przypadku programu **a**. Widać stąd, że efektywność wykonywania programu **a** w pamięci wirtualnej jest wyższa niż programu **b**.



Wykresy charakterystyk programów

Z wykresu krzywych reprezentujących oba programy widać, że nie zbliżają się one asymptotycznie do osi, ale od pewnego miejsca przyjmują stałą wartość. Wynika to stąd, że istnieje pewna graniczna wielkość pamięci realnej *R*, i to taka, że dalsze zwiększenie pamięci realnej nie będzie już powodować zmniejszenia się liczby przesłań pomiędzy pamięcią realną a pamięcią zewnętrzną.

JAK DO TEGO DOJŚĆ?

Technika, która wydaje się być najlepszą przy programowaniu pamięci wirtualnej, polega na pisaniu programu w linii *(in-line)*. Liniowa metoda pisania programu w maksymalnym stopniu ogranicza przeplatanie się dróg przebiegu programu. Do konstrukcji programu wykorzystuje się trzy podstawowe struktury: 1) sekwencyjną, 2) wyboru i 3) powtarzania.

Przy konstrukcji programu należy rezygnować z instrukcji skoku bezwarunkowego GO TO, który w największym stopniu powoduje przeplatanie się dróg przebiegu programu [2]. Szczególną uwagę należy zwrócić na konstrukcję pętli programowych. Sytuacja byłaby najlepsza, gdyby do realizacji pętli potrzebna była jedna strona programu.

Przy programowaniu pamięci wirtualnej bardzo przydatna jest technika programowania strukturalnego. Jedna z głównych wad programowania strukturalnego, jaką jest długość programu i związana z nią większa zajętość pamięci operacyjnej, przy stosowaniu pamięci wirtualnej traci te wady, ale wymaga od programisty odpowiedniego rozmieszczenia bloków instrukcji. Wskazane jest oddzielanie normalnego przebiegu programu od obsługi sytuacji wyjątkowych. Analiza sytuacji wyjątkowej, powodującej nietypowy przebieg programu, winna być dokonana w części głównej programu, natomiast samo działanie wyjątkowe należy zlokalizować poza podstawową linią przebiegu programu. Przykładowo, w przypadku programowania w języku PL/1 można to bardzo łatwo osiągnąć przez odpowiednie oprogramowanie warunków typu ON.

PODPROGRAM CZY MAKROINSTRUKCJA?

Technika stosowania podprogramów jest niezmiernie efektywna i wygodna przy pisaniu większych programów. Oprócz zalet wynikających z ułatwienia konstrukcji programu, osiąga się również wyraźne oszczędności zajętości pamięci operacyjnej. Rozwiązaniem podobnym do techniki podprogramów jest stosowanie odpowiedniej makroinstrukcji. Stosowanie makroinstrukcji nie powoduje jednak oszczędności pamięci operacyjnej, a w niektórych językach programowania jest nawet bardziej kłopotliwe, prowadząc w praktyce do rzadkiego ich stosowania.

W przypadku pamięci wirtualnej sytuacja jest odwrotna, ponieważ technika makroinstrukcji jest znacznie bardziej efektywna niż technika podprogramów. Dzięki bowiem przywołaniu makroinstrukcji i wstawieniu na etapie translacji odpowiedniego ciągu instrukcji powstaje program napisany techniką liniową, minimalizującą niepożądane przesłania.

Jeśli programów nie można zastąpić makroinstrukcjami, należy dążyć do tego, aby — jak już wspomniano — parametry do podprogramów były podawane w sposób jawny. Ponadto wskazane jest umieszczanie podprogramów na końcu programu.

Oddzielnym problemem jest obsługa danych w pamięci wirtualnej. W miarę możliwości należy unikać równoczesnej obsługi kilku zbiorów i raczej obsługiwać je po kolei. Korzystne jest również grupowanie danych w jednostkach równych rozmiarowi stronic, przy czym dane często używane należy zgrupować na jednej stronicy.

Na zakończenie warto jeszcze raz podkreślić, że rozwiązania programowe zalecane do stosowania w programowaniu pamięci wirtualnej, mogą być użyte bez żadnej szkody dla programów wykonywanych w komputerach bez pamięci wirtualnej. Przyzwyczajanie się do tych rozwiązań, a zwłaszcza szkolenie nowych programistów w tym duchu, może przynieść efekty już dzisiaj, a z pewnością przyniesie je wówczas, gdy pamięć wirtualna będzie powszechnie stosowana.

LITERATURA:

[1] Bergstresser R.: Virtual Storage Operation. „DATAMATION", luty 1973

[2] Dijkstra E.: GO TO statements Considered Harmfull. Communications of the ACM, vol. 11 nr 3, 1968

[3] Kurtz J., Cnozzo D.: How to get real benefits from virtual storage. „DATAMATION", luty 1973

[4] Reisman R.: Your software: wirtually at a standstill? „Computers Decisions", lipiec 1973

WIESŁAW BĄBA
Biuro Projektów Przemysłu Materiałów Ogniotrwałych „BIPROMOG"
Gliwice

Usprawnienie projektowania algorytmów

Algorytm był i jest stosowany w informatyce jako środek formalnego — najczęściej graficznego — zapisu strategii rozwiązywania problemów za pomocą komputerów. Obecnie zaczęto go używać do interpretacji aktów prawnych i normatywnych, ujawniania defektów w maszynach i urządzeniach, poszukiwania przyczyn wad produktów, przeprowadzania gier wojennych, sterowania ruchem ulicznym, układania sieci czynności przy montażu urządzeń, sterowania procesem przemysłowym i innych. Definicja algorytmu ulegała zmianom i rozszerzeniom w miarę zwiększania się jego obszaru zastosowań. W rezultacie takiego stanu można obecnie zaproponować następującą ogólną formułę algorytmu:

„algorytm jest to oparty o odpowiednie reguły ściśle określony formalnie sposób postępowania z dokładnie sprecyzowaną kolejnością elementarnych operacji i jednoznacznością interpretacji, doprowadzający do rozwiązania problemu na podstawie danych źródłowych".

W zależności od klasy zastosowań algorytmy można podzielić na dwa podstawowe rodzaje:
a) algorytmy „obliczeniowe" (z przewagą czynności obliczeniowych)
b) algorytmy „decyzyjne" (z przewagą czynności decyzyjnych).

Algorytm „obliczeniowy" używany do rozwiązywania problemów obliczeniowych jest dokładną receptą matematyczną, określającą jednoznacznie proces obliczeniowy od danych źródłowych do żądanego rezultatu. Algorytm „decyzyjny", używany do rozwiązywania pozostałych problemów, najczęściej decyzyjnych, jest ścisłą procedurą prowadzącą przez ciąg hierarchicznie uporządkowanych operacji logicznych do podjęcia prawidłowej decyzji przy uwzględnieniu wszystkich czynników mających dla decyzji tej istotne znaczenie.

Stosowane w informatyce algorytmy obliczeniowe przedstawiane są przede wszystkim w postaci schematów blokowych działania programów (sieci działania programów). Należy dodać, że schematy blokowe używane są również do opisu działania systemów informatycznych lub ich części (jednostek przetwarzania) oraz do obrazowania konfiguracji systemu komputerowego.

Wynikiem opracowania algorytmu decyzyjnego jest również schemat blokowy jego działania. Schematy blokowe działania algorytmów nazywane są w literaturze [2, 11, 12, 13] kartami przepływu *(ang. flow charts)*.

Celem niniejszego artykułu jest przedstawienie propozycji wytycznych projektowania algorytmów wykorzystywanych do opracowania programów przetwarzania danych oraz do rozwiązywania problemów decyzyjnych. Osobne miejsce poświęcono metodyce tworzenia schematów blokowych działania algorytmów.

Mgr inż. Wiesław BĄBA jest absolwentem Wydziału Elektrycznego Politechniki Śląskiej i Studium Podyplomowego Automatyki przy Politechnice Śląskiej w Gliwicach. W latach 1968—1970 pracuje w Instytucie Metalurgii Żelaza w Gliwicach wykonując badania oparte na obliczeniach zrealizowanych na komputerze ICL 4-50, a następnie w latach 1970—1974 w ZETO Katowice. Od 1974 w Biurze Projektów Przemysłu Materiałów Ogniotrwałych w Gliwicach, gdzie zajmuje się automatyzacją prac projektowych oraz projektowaniem struktur technologiczno-organizacyjnych ośrodków obliczeniowych.

Wykorzystane symbole graficzne do tworzenia schematów działania algorytmów odniesiono do obowiązującej normy PN-75/E-01226 „Przetwarzanie danych. Symbole graficzne" [10]. Mimo wymienionego ujednolicenia postaci symboli graficznych użytkownicy komputerów [3, 4, 8], a zwłaszcza programiści, wykorzystują nadal dość dowolne symbole do tworzenia schematów blokowych działania programów.

Zaproponowano również do rozpatrzenia kilka symboli graficznych uzupełniających oraz zmianę kilku istniejących, których użycie powinno uprościć schematy, zwłaszcza dotyczące działania programów.

OGÓLNA POSTAĆ ALGORYTMU

Struktura algorytmu obliczeniowego, aczkolwiek zawiera składniki wspólne z algorytmem decyzyjnym, jest od niego inna.

Prawidłowo opracowany algorytm obliczeniowy powinien zawierać:

a) **wstępne zdefiniowanie problemu** przez podanie celów, funkcji, założeń, warunków i ograniczeń

b) **opis matematyczny problemu** zawierający wybór metody i określenie aparatu matematycznego realizującego proces obliczeniowy

c) **ogólny opis procedury algorytmu** obejmujący ciąg kolejnych operacji — reguł w postaci kolejnych kroków czynności

d) **schemat blokowy algorytmu** zawierający ciąg znormalizowanych symboli graficznych (bloków) obrazujących czynności algorytmu i powiązanych logiką postępowania zapewniającego otrzymanie prawidłowego rezultatu

e) **opis schematu blokowego algorytmu** określający opisowo funkcje ponumerowanych elementów schematu

f) **opis zmiennych związanych z danymi źródłowymi, pośrednimi oraz wynikami**

g) **potrzeby sprzętowe algorytmu** (dla programów przetwarzania danych — określający wielkość pamięci operacyjnej, wielkość i rodzaj pamięci zewnętrznej oraz rodzaj i liczbę urządzeń zewnętrznych systemu komputerowego)

h) **przykład korzystania z algorytmu.**

Wydaje się, że przedstawiony powyżej zestaw składników algorytmu wyczerpuje w całości opis problemu przeznaczonego do oprogramowania.

Niektórzy programiści utożsamiają algorytm obliczeniowy przede wszystkim ze schematem blokowym działania programu. Jak wiadomo schemat blokowy ma na celu przedstawienie zasadniczych czynności algorytmu i zobrazowanie jego działania za pomocą języka graficznego, ilustrującego tylko ogólną koncepcję i organizację obliczeń. Schemat blokowy działania algorytmu pozwala wyjaśnić strukturę programu bez wchodzenia w szczegóły samego oprogramowania. Oprócz tego do zakodowania programu potrzeba jednak szeregu informacji uzupełniających, które nie tylko ułatwiają programowanie, ale również pełne zrozumienie całości problemu.

Jest to szczególnie ważne w przypadku algorytmów złożonych, gdzie zrozumienie powiązań i współzależności występujących w programie staje się bardzo trudne i nieczytelne, i to nie tylko dla przeciętnego użytkownika, ale nawet dla samego twórcy programu, lub bardzo doświadczonego programisty, jeśli musi on dokonać zmian i uzupełnień w istniejącym programie.

Algorytmy decyzyjne najczęściej tworzy się w oparciu o gotowe sformułowanie lub informacje ustne uzyskane od użytkownika.

Jakkolwiek algorytmy decyzyjne mogą przybierać różną postać, można wyróżnić w nich następujące składniki spójne:

a) **opisowe zdefiniowanie problemu** przez określenie celów, funkcji, warunków głównych i dodatkowych oraz wymagań specjalnych

b) **konwersję definicji opisowej problemu** w strukturę umożliwiającą tworzenie schematu blokowego
c) **schemat blokowy algorytmu**
d) **opis schematu blokowego** (jeżeli jest konieczny).

Istnieją następujące struktury algorytmu decyzyjnego [13]:
a) logiczny wykaz czynności
b) logiczne drzewo decyzji
c) logiczna tablica decyzyjna
d) obliczenia kierowane.

Logiczny wykaz czynności zawiera uporządkowany hierarchicznie ciąg pytań, na które podane są odpowiedzi w postaci instrukcji postępowania lub gotowych wyników. Wykaz czynności może wystąpić w postaci tekstu lub diagramu czynności. Istnieją dwie zasadnicze drogi tworzenia drzew decyzyjnych:
a) przez hierarchiczne uporządkowanie pytań z instrukcjami lub bez instrukcji
b) przez tworzenie tablicy kanonicznej stanów, uwzględniającej operatory (warunki) logiczne i arytmetyczne.

Logiczne drzewo decyzji, składające się z uporządkowanych pytań, przyczynia się do uzyskania jednoznacznego rozwiązania, bez umieszczania po drodze instrukcji dla przekazywania treści np. przepisów i regulaminów lub też z umieszczonymi instrukcjami dla rozwiązywania problemów z dziedziny np. wyszukiwania uszkodzeń maszyn [13].

Drzewo decyzji tworzone przy pomocy tablicy kanonicznej stanów opiera się o sformułowanie warunków logicznych i arytmetycznych problemu. Warunki logiczne badają zależność relacji, a arytmetyczne określają stan liczebny operacji powiązanych przez odpowiednie zależności. Tablica kanoniczna przedstawia wszelkie możliwe kombinacje warunków logicznych w powiązaniu z operatorami arytmetycznymi w postaci tablicy binarnej.

Do rozwiązywania pewnych rodzajów problemów wykorzystuje się tablice decyzyjne, które są tabelarycznym układem czynności uzależnionych od spełnienia określonych warunków.

Algorytm w postaci tablicy decyzyjnej, pomimo jej znacznej pracochłonności, jest obecnie często spotykaną postacią użytkową algorytmu, która jest stosowana ze względu na pewność uwzględnienia wszystkich możliwych stanów. Niekiedy opracowanie i stosowanie „czystego" algorytmu decyzyjnego jest niecelowe ze względu na dużą jego pracochłonność i uciążliwość użytkowania. W takich przypadkach problem należy rozwiązać w formie sterowanego obliczenia, które jest środkiem dochodzenia do decyzji przy rozpatrywaniu tylko tych czynników (arytmetycznych), które wpływają na jej treść. Kierowane obliczenie jest więc precyzyjnym sposobem przechodzenia od faktów zdeterminowanych operacjami obliczeniowymi problemu do jego rozwiązania, np. przy obliczaniu podatków [13].

SCHEMATY BLOKOWE I PROPOZYCJE ZMIAN SYMBOLI GRAFICZNYCH

Przedstawione poniżej elementy schematów blokowych algorytmów mogą być wykorzystane przede wszystkim dla schematów blokowych algorytmów obliczeniowych, a część z nich (1.1, 1.2, 1.3, 1.4, 2.1, 2.2, 2.10, 2.11, 2.12) w algorytmach decyzyjnych.

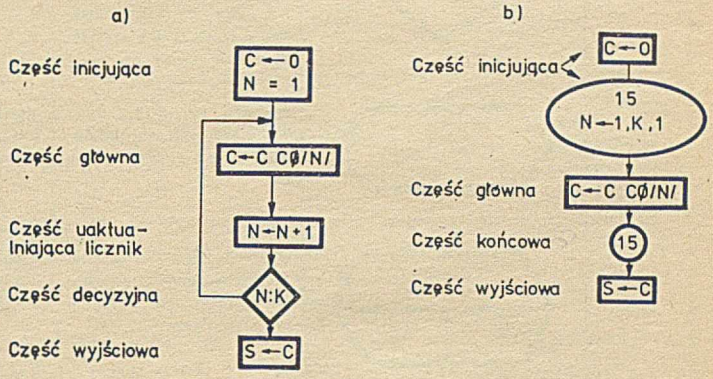
Istniejące programy przetwarzania danych można ogólnie podzielić na programy proste, segmentowane nienakładkowe i nakładkowe. Mając na uwadze powyższy podział, w schematach blokowych działania programów można wyróżnić następujące czynności zobrazowane przez elementy (bloki):
a) operacja lub grupa operacji, w wyniku których ulega zmianie wartość, postać lub miejsce zapisu danych
b) decyzja (przełącznik, rozgałęzienie, predykat) — logicznie sprawdzająca warunki lub kryteria
c) wprowadzanie danych
d) wyprowadzanie danych
e) pętle
f) podprogramy
g) segmenty nienakładkowe
h) nakładki
i) składowanie danych
j) informacje sterujące.

Tablica 1 zawiera propozycje głównych symboli graficznych dla powyższych elementów schematów blokowych.

Tablica 2 zawiera propozycje pomocniczych symboli graficznych, których część koresponduje z symbolami głównymi jak np.: 2.5 i 2.6 z 1.5; 2.7 i 2.8 z 1.6; 2.9 z 1.7.

Opracowując propozycję starano się uwzględnić następujące cele:
a) dobrać ograniczoną ilość indywidualnych symboli graficznych pozwalających rozróżnić podane poprzednio czynności w sposób jednoznaczny i bez dodatkowych opisów
b) zmniejszyć ilość informacji tekstowej i symbolów literowych
c) zmniejszyć pracochłonność sporządzania schematów blokowych
d) zwiększyć przejrzystość schematów blokowych.

Wynikające z powyższego pewne zwiększenie ilości symboli graficznych pozwala poprawić przejrzystość schematów blokowych oraz uwzględnić dodatkowe funkcje, zwłaszcza występujące w algorytmach obliczeniowych.



Rys. 1. Istniejący (a) i proponowany (b) sposób przedstawiania pętli

Rys. 1 przedstawia sposób zastąpienia tradycyjnego opisu graficznego pętli poprzez symbole nr 1.7 i 2.9. Symbol początku pętli (1.7) ma postać zbliżoną do symbolu 2.4. wg polskiej normy PN-75/E-01226 oznaczającego modyfikację rozkazu. Przedstawione dalsze zmiany postaci symboli istniejących oraz wprowadzenie symboli dodatkowych skomentowano w kolumnie uwagi tablic 1 i 2.

Przedstawiona zmiana postaci niektórych symboli graficznych w stosunku do normy PN-75/E-01226 oraz nowe symbole graficzne są oczywiście propozycją do przedyskutowania. Zaproponowane formy geometryczne symboli są proste i tym samym łatwe do wprowadzenia w szablonach. Powinny one moim zdaniem w istotny sposób skrócić i uprościć rysowanie schematów blokowych działania programów, nie mówiąc już o poprawie ich czytelności.

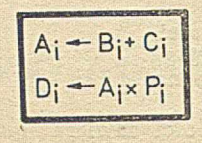
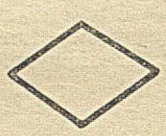
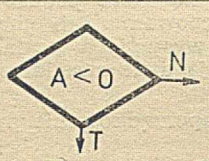
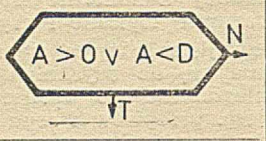
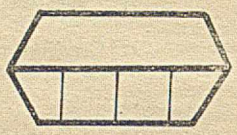
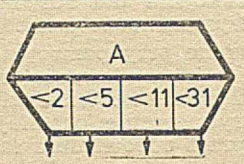
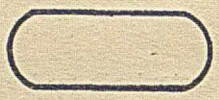
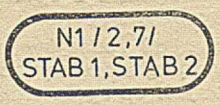
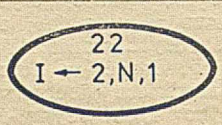
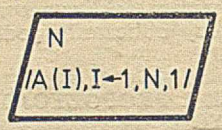
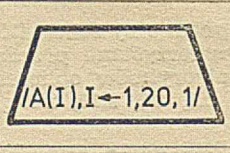
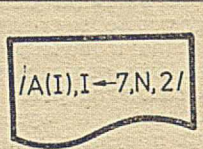
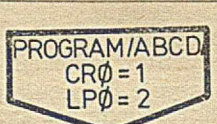
WYTYCZNE PROJEKTOWANIA ALGORYTMÓW

Projektant algorytmu musi dokładnie poznać problem przeznaczony do rozwiązania, wniknąć w istotę i specyfikę zagadnienia poznając wszelkie jego mutacje, wszystkie możliwe układy warunków redundantnych i wzajemnie sprzecznych oraz wagę ograniczeń, prowadząc tym samym do ścisłego i jednoznacznego zdefiniowania problemu.

W odniesieniu do algorytmów decyzyjnych wytyczne ich projektowania są zróżnicowane w zależności od zastosowanej struktury algorytmu. Jeżeli korzystamy z logicznego drzewa decyzji tworzonego procedurą hierarchicznego uporządkowania pytań, należy wyróżnić następujące kroki postępowania:
a) ustalenie wszelkich możliwych przypadków
b) wybranie pytań kluczowych, które dzielą liczbę pozostałych alternatyw możliwie na połowy
c) podział pytań na grupy logiczne wg zależności o podobnych treściach
d) konstrukcję drzewa logicznego w graficznej postaci schematu blokowego.

Drzewo decyzji tworzone przy pomocy tablicy kanonicznej stanów obejmuje następujące procedury:
a) sformułowanie operatorów logicznych i arytmetycznych
b) utworzenie tablicy kanonicznej stanów, określającej współzależność pomiędzy operatorami logicznymi i arytmetycznymi
c) konwersję tablicy kanonicznej w logiczne drzewo decyzji o postaci źródłowej
d) redukcję źródłowego drzewa decyzji w drzewo wynikowe
e) konwersję wynikowego drzewa decyzji w schemat blokowy decyzyjny.

Tablica 1. Symbole graficzne główne

Lp.	Symbol graficzny	Opis funkcji symbolu	Opis zawartości symbolu	Przykład zastosowania	Uwagi
1.1		operacja lub grupa operacji powodująca zmianę działania lub zmianę wartości postaci albo miejsca zapisu danych	opis działania lub czynności, albo opis operacji arytmetycznej		symbol jak w 2.1 normy PN-75/E-01226
1.2		decyzja określająca wybór jednej z alternatywnych dróg działania w oparciu o symboliczne (krótkie) warunki logiczne lub arytmetyczne w liczbie do trzech	proste warunki logiczne lub arytmetyczne w liczbie do trzech (najczęściej dwa)		symbol jak w p. 2.3 normy PN-75/E-01226
1.3		decyzja jak wyżej, lecz dla złożonych warunków, albo treść z długim opisem	złożone warunki logiczne lub arytmetyczne w liczbie do trzech (najczęściej dwa)		symbol dodatkowy, kształt pokrywa się z p. 2.4 normy PN-75/E-01226
1.4		operacja jak wyżej lecz dla badań złożonych wielowarunkowych (co najmniej trójwarunkowych)	wiele warunków logicznych lub arytmetycznych o złożonej strukturze		symbol dodatkowy
1.5		wywołanie nakładki	nazwa nakładki (numer), parametry sterujące nakładką		symbol dodatkowy
1.6		wywołanie podprogramu	nazwa podprogramu i parametry aktualne		symbol dodatkowy, kształt pokrywa się z p. 2.39 normy PN-75/E-01226
1.7		określenie początku pętli programowej	symbol etykiety końca pętli oraz wartości indeksu pętli: początkowej, końcowej, kroju pętli		symbol dodatkowy
1.8		wprowadzenie danych z zewnątrz programu	adresy symboliczne (nazwy zmiennych prostych i indeksowanych), gdzie umieszczono wczytane dane		symbol zgodny z p 2.2 normy PN-75/E-01226 w zakresie wprowadzania informacji
1.9		wprowadzenie danych umieszczonych w programie	adresy symboliczne, gdzie umieszczono wczytane dane		symbol dodatkowy
1.10		wprowadzenie danych na zewnątrz programu	adresy symboliczne skąd wyprowadzane są dane		symbol dodatkowy- kształt pokrywa się z p. 2.17 normy PN-75/E-01226 — najczęściej dane wyprowadza się na nośnik czytelny dla człowieka
1.11		składowanie danych w pamięci zewnętrznej	adresy symboliczne zmiennych składowych w pamięci zewnętrznej w kolejności zapisu jaki występuje w rekordzie danych		symbol dodatkowy
1.12		informacje sterujące programem	instrukcje sterujące programem, segmentem; deklaracje konfiguracji systemu		jw.

Tablica 2. Symbole graficzne pomocnicze

Lp.	Symbol graficzny	Opis funkcji symbolu	Opis zawartości symbolu	Przykład zastosowania	Uwagi
2.1	START	początek programu	—	—	symbol zmieniony, niezgodny z p. 2.39 normy PN-75/E-01226
2.2	STOP	koniec programu	—	—	symbol zmieniony, niezgodny z p. 2.39 normy PN-75/E-01226
2.3		początek grupy segmentów nienakładkowych, powiązanych ze sobą logicznie lub własnościami	numer segmentu (cyfra rzymska)	I	symbol dodatkowy (przy wykonywaniu programu przez kilku programistów)
2.4		koniec grupy segmentów nienakładkowych (patrz 2.3)	numer segmentu (cyfra rzymska)	II	jw.
2.5		początek nakładki	nazwa nakładki lub numer, parametry sterujące nakładką, nazwy segmentów umieszczonych w nakładce	N1/2,7/ STAB1, STAB2	symbol dodatkowy
2.6	KONIEC	koniec nakładki	nazwa nakładki lub numer	N1 KONIEC	jw.
2.7		początek podprogramu	nazwa podprogramu, parametry formalne	SORT /IO, AØ, CØ, DØ/	symbol dodatkowy
2.8	POWRÓT	koniec podprogramu	—	—	jw.
2.9		łącznik międzyoperacyjny	etykieta lub numer wiersza programu	1E	symbol dodatkowy-kształt pokrywa się z p. 2.37 normy PN-75/E-01226
2.10		łącznik stronicowy	numer łącznika stronicowego	2	symbol zmieniony, niezgodny z p. 2.37 normy PN-75/E-01226
2.11		łącznik międzystronicowy	numer strony	7	symbol jak w p. 2.38 normy PN-75/E-01226
2.12		skierowany przepływ danych lub droga działania	—	—	symbol jak w p. 2.34 normy PN-75/E-01226
2.13		skrzyżowanie powiązanych logicznie dróg działania		1E	oznaczenie dodatkowe
2.14		skrzyżowanie nie powiązanych logicznie dróg działania	—	—	oznaczenie jak w p. 2.35 normy PN-75/E-01226

Jakkolwiek źródła [5, 13] podają dokładną metodykę tworzenia tablic decyzyjnych w oparciu o odpowiednie przykłady, spróbujmy dokonać syntezy procedury układania tychże tablic, którą można sformułować następująco:

a) analiza problemu w celu określenia podproblemów opisanych przez elementarne tablice decyzji

b) identyfikacja pytań tworzących listę warunków

c) identyfikacja wyników tworzących listę akcji

d) wypełnienie tablicy decyzji, wiążąc listę warunków z listą akcji

e) kontrola zawartości tablicy decyzyjnej (prawidłowości opisu problemu, kompletność, prawidłowość ułożenia zgodnie z zasadami logiki)

f) kompresja tablicy przez eliminację reguł decyzji redundantnych i sprzecznych oraz łączenie prostych reguł decyzji w reguły złożone

g) optymalizacja tablicy przez takie uporządkowanie tablicy, aby otrzymać najkrótszy czas dojścia do wyniku (krótki czas pracy programu) lub uwzględnienie czynników szczególnych

h) kontrola poprawności zapisu, struktury i logicznej treści gotowych tablic

i) układanie schematu blokowego od warunku kompletnego (tzn. od warunku posiadającego wszystkie odpowiedzi), który wyznacza pierwsze pytanie algorytmu.

Uwzględnienie w algorytmie czynników szczególnych w jego logice ma swój sens np. dla wyszukiwania uszkodzeń w urządzeniach, gdzie istotną rolę odgrywa częstość występowania zdarzenia oraz zagadnienie ergonomii.

Tablice decyzji służą również do konwersji algorytmów ułożonych do postaci uwzględniającej dodatkowe warunki stawiane przez użytkownika. Z algorytmu oryginalnego otrzymuje się np. tak zwany algorytm operacyjny, uwzględniający czynniki ergonomiczne [13]. Oprócz tego tablice decyzyjne są fragmentami programów lub systemów informatycznych.

Algorytm kierowanego obliczenia wymaga:

a) hierarchicznej identyfikacji pytań,

b) identyfikacji akcji arytmetycznych powiązanych z pytaniami

c) utworzenia sieci powiązań między pytaniami i akcjami arytmetycznymi

d) przekształcenia sieci powiązań w schemat blokowy działania algorytmu.

Struktura algorytmu obliczeniowego podana w punkcie „Ogólna postać algorytmu" narzuca automatycznie odpowiednią przy jego tworzeniu procedurę postępowania.

Oprócz podanych wyżej wytycznych, przy tworzeniu algorytmów obliczeniowych należy uwzględnić również pewne zasady ogólne, dzięki którym algorytm ma zapewnić:

a) pełną kompletność i ścisłość funkcji głównych oraz pomocniczych

b) prawidłowe wyniki dla dowolnych danych

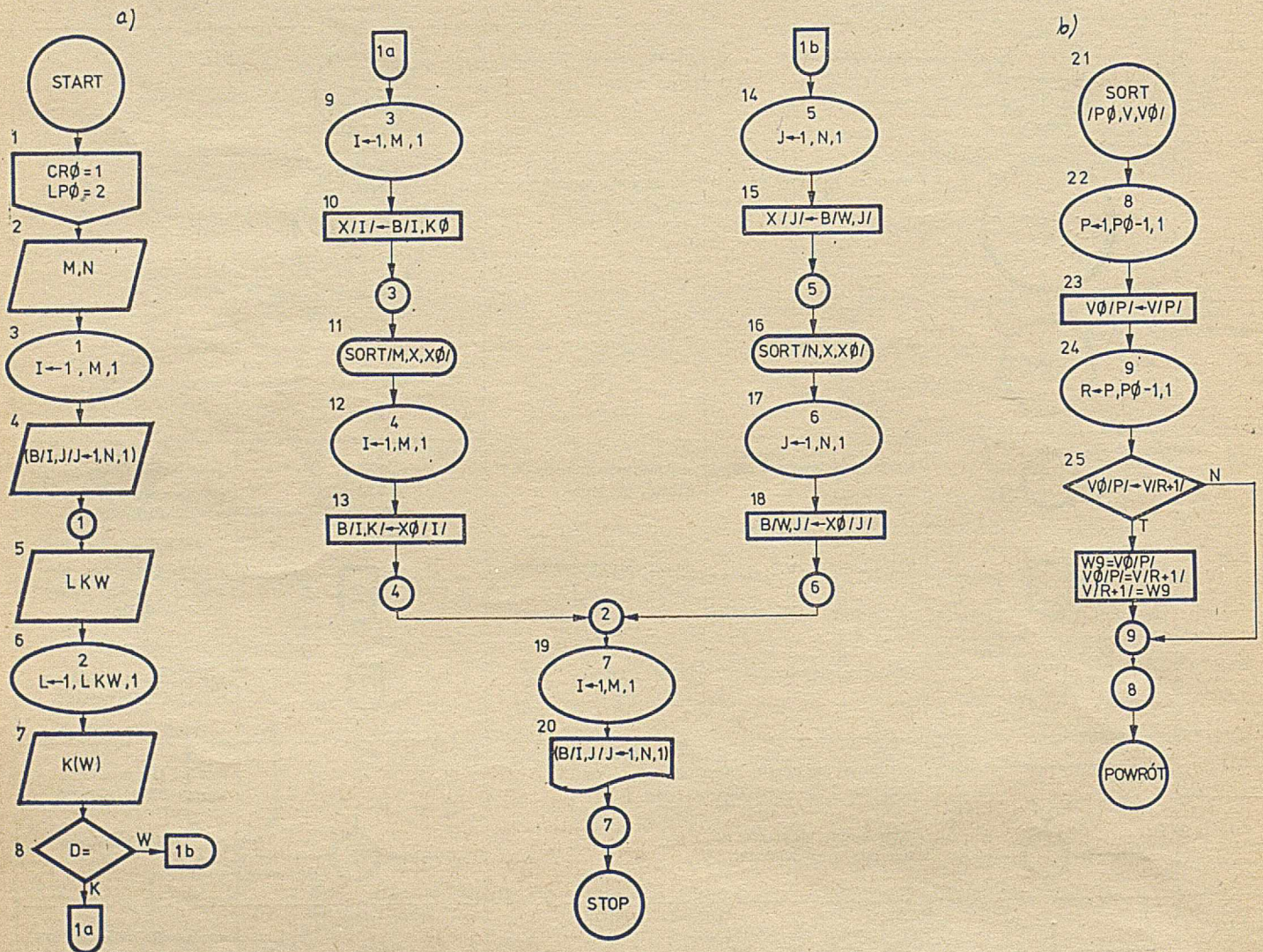
c) najkrótszy i najbardziej zwięzły program

d) najszybsze wykonywanie programu na danym komputerze

e) możliwie małe obciążenie pamięci (optymalną gospodarkę zasobami pamięci)

f) najlepszą przejrzystość schematów blokowych

g) najmniejszą pracochłonność zaprogramowania.



Rys. 2. Poglądowy przykład zastosowań symboli 1 i 2 dla schematu blokowego działania programu porządkowania kolumn lub wierszy tablicy dwuwymiarowej: a) program ogólny, b) podprogram sortowania w porządku niemalejącym

WYTYCZNE PROJEKTOWANIA SCHEMATÓW BLOKOWYCH ALGORYTMÓW

Jak już wspomniano, jednym z głównych celów projektowania algorytmu jest utworzenie schematu blokowego jego działania w ustalonej postaci graficznej. Korzystając z ustalonego repertuaru symboli graficznych, dostosowanych do potrzeb projektowania schematu blokowego działania algorytmu, należy kierować się następującymi zasadami (część z nich ujęta jest w normie PN-75/E-01226):

a) jako linie łączące symbole graficzne należy wykorzystywać przede wszystkim linie pionowe i poziome; w uzasadnionych przypadkach można wyjątkowo skorzystać z linii ukośnej, natomiast nie można korzystać w żadnym przypadku z linii krzywych

b) nie dopuszczać do krzyżowania linii; jeżeli jest to niemożliwe, używać łączników stronicowych lub w ostateczności symbolu 2.14.

c) do danego symbolu graficznego doprowadzać tylko jedną linię; jeżeli jest to niemożliwe, używać etykiet (patrz p. 2.9 w tablicy 2)

d) linia użyta w schemacie musi być jednoznacznie połączona z elementem schematu i oznaczona strzałką (ułatwia to drogę śledzenia czynności algorytmu)

e) kierunek śledzenia algorytmu ma przebiegać od góry w dół i od lewej strony ku prawej

f) jeżeli istnieją dwie ścieżki logiczne algorytmu, to główna ma być po lewej stronie arkusza; jeśli istnieje wiele ścieżek, to główna ma być umieszczona w środkowej części arkusza

g) należy uwzględnić wszystkie możliwości algorytmu

h) w celu ułatwienia kodowania programu używać symboli tekstowych ogólnie znanych i używanych w informatyce

i) użyte w schemacie środki dodatkowe (poza symbolami) muszą być kompletne i ułatwiające rozwiązanie problemu

j) w przypadku złożonego problemu należy najpierw wykonać główny schemat blokowy, zawierający poszczególne podproblemy, a dopiero w drugiej kolejności — schematy szczegółowe podproblemów.

Jeżeli projektujemy schematy blokowe działań programu przetwarzania danych należy dodatkowo stosować następujące zasady:

a) w symbolach operacji przetwarzania zawrzeć jedną operację lub ich grupę, jeżeli nie są rozdzielone etykietą lub symbolem graficznym innej odrębnej czynności

b) w symbolach wprowadzania i wyprowadzania danych operować masywami danych w postaci tablic jedno-, dwu-, i trzywymiarowych, stosując odpowiednie oznaczenia (patrz p. 1.8, 1.9, 1.10 w tablicy 1). Rozwiązanie to eliminuje konieczność tworzenia pętli.

c) w symbolach należy umieszczać nazwy zmiennych prostych i indeksowanych

d) wszystkie bloki schematu blokowego — z wyjątkiem etykiet i łączników należy numerować w ramach programu, umieszczając liczby z lewej strony powyżej symbolu graficznego.

Schemat blokowy działań złożonego programu powinien zawierać:

a) ogólny schemat blokowy powiązania oddzielnych części programu

b) schematy blokowe działania programu głównego, segmentów, nakładek

c) schematy blokowe działania podprogramów.

Rysunek 2 przedstawia przykład zastosowania proponowanych symboli graficznych i użycia podanych wcześniej wytycznych przy tworzeniu schematu blokowego działania algorytmu obliczeniowego.

* * *

Przedstawiona próba syntezy zasad projektowania algorytmów, oparta o literaturę oraz doświadczenia autora, jest ze zrozumiałych względów skrótem fragmentu tworzonego obecnie przez autora opracowania „Wytyczne tworzenia dokumentacji systemów przetwarzania danych naukowo-technicznych na potrzeby B.P. „Bipromog". Zawarte w artykule propozycje wytycznych projektowania algorytmów, ze szczególnym uwzględnieniem schematów blokowych działania programów, mogą być jednym z materiałów do dyskusji nad ostatecznym kształtem metod projektowania algorytmów.

Sprawdzone w praktyce przez autora procedury tworzenia algorytmów obliczeniowych i decyzyjnych wskazują na duże korzyści wynikające z kompletnego i optymalnego opisu problemu, a zwłaszcza przejrzystych schematów blokowych działania algorytmu. W przypadku akceptacji zaproponowanych symboli graficznych przez dostatecznie szeroki krąg użytkowników, a następnie przez PKN w formie nowelizacji normy PN-75/E-01226, mogłyby się one stać jednym z elementów dalszego ujednolicenia i udoskonalenia postaci dokumentacji programowej systemów przetwarzania danych.

LITERATURA:

[1] Bauer F. L., Goos G.: Informatyka. WNT Warszawa 1977

[2] Benice D. D.: Introduction to computers and data processing. Prentice — Hall Inc. Englewood Cliffs, New Jersey 1970

[3] Kierzkowski Z.: Elementy informatyki. PWN Warszawa — Poznań 1976

[4] Niederliński A.: Mikrokomputery i minikomputery. WSiP Warszawa 1978

[5] Niedźwiecki J.: Logiczne tablice decyzyjne. Podstawy teoretyczne i Zastosowania praktyczne. Problemy Informatyki, OBRI Warszawa 1975

[6] Praca zbiorowa pod redakcją Z. Hellwiga: Maszyny cyfrowe i ich zastosowanie. PWE Warszawa 1975

[7] Praca zbiorowa pod redakcją P. Mullera: Leksykon informatyki. WNT Warszawa 1977

[8] Praca zbiorowa pod redakcją E. Niedzielskiej: Informatyka — Poradnik dla ekonomistów. PWE Warszawa 1977

[9] Praca zbiorowa pod redakcją M. F. Woltersa: Klucz do maszyny cyfrowej. WNT Warszawa 1977

[10] PN-75/E-01226. Przetwarzanie danych. Symbole graficzne

[11] Ralston A., Wilf H. S.: Mathematical methods for digital computers. vol. 1, 2, John Wiley & Sons Inc. 1976

[12] Schriber T. J.: Fundamentals of flowcharting. John Wiley & Sons Inc. 1969

[13] Wheatley D. M., Unwin A. W.: Algorytmy. PWE Warszawa 1975

AP - 80: Komunikat nr 1

W dniach 15—16 maja 1980 r. odbędzie się w Białymstoku III konferencja na temat automatyzacji projektowania AP-80.

Organizatorzy konferencji — Ośrodek Elektronicznej Techniki Obliczeniowej Politechniki Białostockiej i Komisja Informatyki Zarządu Głównego PZITB — **zapraszają do zgłaszania referatów związanych z tematyką konferencji.**

Termin nadsyłania propozycji (konspekt referatu — w trzech egzemplarzach) upływa z dniem 15 października br.

O zakwalifikowaniu referatu przez komitet organizacyjny zostaną Autorzy powiadomieni do dnia 1 listopada br. Pełny tekst referatu należy nadesłać do 1 lutego 1980 r. Referaty zostaną wydrukowane w materiałach konferencyjnych. Szczegóły dotyczące uczestnictwa w konferencji zostaną podane w następnym komunikacie.

Korespondencję należy kierować pod adresem:
mgr inż. Barbara Jakubowska
Politechnika Białostocka — Ośrodek Elektronicznej Techniki Obliczeniowej
ul. Wiejska 45, 15-351 Białystok.

RYSZARD GILECKI, MAŁGORZATA SZUBERT
Centrum Informatyki Energetyki i Energii Atomowej
Warszawa

Praktyczne aspekty zastosowania systemu GEORGE 2

Maszyna cyfrowa ODRA 1305, zainstalowana w Centrum Informatyki Energetyki i Energii Atomowej, pracuje od niedawna pod kontrolą systemu operacyjnego GEORGE 2. Poprzednio działaniem maszyny zarządzał jedynie program sterujący Egzekutor. Średni czas obrotu rozwiązywanego zadania był wówczas bardzo długi, a jego skrócenie praktycznie niemożliwe. Wiązało się to z następującymi wadami Egzekutora:

- działanie bezpośrednie czytnika kart i drukarki wierszowej, uniemożliwiające praktycznie pracę wieloprogramową
- konieczność licznych interwencji operatora, powodująca często przestoje całości sprzętu.

Jako metodę poprawy efektywności działania omawianej instalacji wybrano zastosowanie systemu operacyjnego GEORGE 2. Dokonując wyboru między systemami GEORGE 2 i GEORGE 3 kierowano się następującymi kryteriami, przemawiającymi na korzyść zastosowania tego pierwszego:

- stosunkowo szczupła konfiguracja sprzętowa (jedna drukarka wierszowa, trzy jednostki pamięci dyskowej EDS 30)
- względna łatwość wdrożenia, bez konieczności jakichkolwiek modyfikacji sprzętu i przerywania normalnego toku pracy ośrodka
- niższy narzut na działanie programów systemu operacyjnego.

KIERUNKI PRAC WDROŻENIOWYCH

W trakcie wstępnej analizy ustalono dwa główne kierunki prac wdrożeniowych:

- wybór i wygenerowanie odpowiedniej wersji systemu operacyjnego
- opracowanie metod i narzędzi współpracy użytkowników z systemem operacyjnym.

Przebieg procesu wdrożenia wykazał — wbrew początkowym przypuszczeniom — względną łatwość realizacji zamierzenia pierwszego, jednocześnie znaczną pracochłonność drugiego.

GENERACJA SYSTEMU

Proces generacji (wybór parametrów i realizacja) nie przedstawiał większych trudności. Po wstępnych próbach ustalono, że znacznie bardziej przydatna będzie wersja strumieniowa. Postanowiono także wyeliminować drukowanie znacznej części komunikatów do operatora.

Zagadnieniem, na które naszym zdaniem należy zwrócić największą uwagę w procesie generacji systemu, jest określenie wielkości i sposobu rozmieszczenia zbiorów systemowych, zwłaszcza zbiorów wejściowych i wyjściowych. Zbyt mała wielkość tych zbiorów może być przyczyną przestojów w pracy systemu, natomiast ich niewłaściwe rozmieszczenie powodować nadmierne ruchy głowic pamięci dyskowej i w konsekwencji znaczne spowolnienie procesu przesyłania danych. Dyskusyjny jest zwłaszcza problem, czy wszystkie zbiory wejściowe i wyjściowe umieszczać na jednym wspólnym pakiecie dyskowym, czy też je rozdzielać.

Biorąc pod uwagę wspomnianą ograniczoność zasobów pamięci dyskowej (3 jednostki EDS 30), postanowiono w pierwszej fazie umieścić wszystkie zbiory systemowe na jednym pakiecie, a dopiero później na podstawie wyników badań efektywności systemu zdecydować, czy pozostawić ten stan, czy też dokonać rozdzielenia zbiorów. W chwili obecnej (po 4 miesiącach eksploatacji systemu) nie potrafimy jeszcze takiej decyzji podjąć. Sądzimy, że eksploatacja wersji dwupakietowej może napotykać duże trudności w przypadku zadań wykorzystujących wiele zbiorów użytkowych na dyskach.

Wybrane podczas generacji wielkości zbiorów (8 zbiorów wejściowych po 1 cylindrze i 8 wyjściowych po 5 cylindrów) okazały się wystarczające. Pozwalają na zapamiętanie maksymalnie informacji odpowiadających około 15 000 kart dziurkowanych z danymi i około 50 000 wierszy wyników (maksymalna wielkość danych i wyników jednego zadania wynosi odpowiednio około 7500 kart i około 25 000 wierszy). Nie wystąpił dotąd przypadek zatrzymania pracy modułu centralnego na skutek zapełnienia zbiorów wyjściowych — sytuacji takiej można spodziewać się jedynie w przypadku kilkugodzinnej awarii drukarki.

Przygotowanie programistów oraz eksploatowanych programów do współpracy z systemem GEORGE 2 sprawiało duże trudności. Najogólniej można wyodrębnić dwie grupy przyczyn tego faktu:

- wykorzystywanie w istniejących programach szczególnych własności poprzedniego trybu pracy (działania jednoprogramowego pod kontrolą Egzekutora)
- mała przydatność standardowych sekwencji zdań sterujących dostarczanych przez producenta.

DOTYCHCZASOWY TRYB PRACY

Wiele eksploatowanych w naszym ośrodku programów przystosowanych było jedynie do działania w trybie jednoprogramowym i pod bezpośrednią kontrolą operatora. Autorzy programów bardzo często korzystali w pracach projektowo-implementacyjnych z tych szczególnych własności dotychczasowego trybu przetwarzania.

Przy wdrażaniu systemu GEORGE kłopotliwe okazały się szczególnie następujące dwie cechy programowania opartego na takich założeniach:

- konwersacyjny charakter programów, np. wprowadzanie parametrów i danych z pulpitu operatora oraz decydowanie o kontynuowaniu lub zaniechaniu wykonywania programu na podstawie wizualnej obserwacji pracy maszyny
- nieodpowiedni sposób etykietowania zbiorów na taśmach magnetycznych (brak numerów generacji i zerowe okresy zachowania), przy którym wyłącznie operator odpowiedzialny był za udostępnienie programowi właściwych zbiorów taśmowych i zagwarantowanie bezpieczeństwa danych.

Wzory etykietowania zbiorów taśmowych zalecane przez producenta i stosowane przez programistów naszego ośrodka (mamy tu na myśli szerokie stosowanie etykiety PROGRAM TAPE) okazały się nieprzydatne w warunkach pracy wieloprogramowej. Odróżnianie zbiorów o jednakowych nazwach za pomocą numerów seryjnych taśm jest kłopotliwe w realizacji. Dlatego w naszym ośrodku przyjęto koncepcję unikalnych nazw zbiorów taśmowych, np. biblioteka podstawowa ma etykietę PROGRAM PODS, a bliblioteka naukowa — PROGRAM NAUK. Również programistom użytkowym zalecono unikanie stosowania etykiety PROGRAM TAPE.

Problemem, który znacznie zmniejszył efekty wdrożenia systemu GEORGE 2, był niski stopień wykorzystania pamięci dyskowej. Większość programistów przechowywała swoje zbiory stale wyłącznie na taśmach magnetycznych. Przyczyny takiego stanu rzeczy były historyczne — pamięć dyskowa została zainstalowana znacznie później niż podstawowa konfiguracja, a w początkowym okresie eksploatacji charakteryzowała się dużą zawodnością. Problemy sprzętowe wiązały się z faktem, że dyski zostały wyprodukowane przez firmę TRANSAMERICA i wymagają niestandardowego złącza z jednostką centralną ODRA 1305.

Wśród przyczyn niechęci użytkowników do dysków niemałą rolę odgrywa też fakt, że korzystanie z nich wymaga stosowania odrębnej grupy programów standardowych. Zdaniem autorów, rozłączność programów organizacyjnych dla pamięci taśmowej i dyskowej jest bardzo istotną wadą oprogramowania standardowego maszyn ODRA (problem ten nie istnieje w systemie OS/JS, gdzie nie ma podziału programów standardowych na dyskowe i taśmowe, a zdania języka sterującego opisują, jakiego rodzaju pamięci używamy).

JĘZYK STERUJĄCY

W procesie wdrażania systemu zrezygnowaliśmy z zastosowania dostarczanych przez producenta standardowych sekwencji zdań sterujących (makroinstrukcji), uznając je w większości za nieprzydatne w naszych warunkach. Za podstawowe kryteria oceny przydatności zestawu makroinstrukcji uważamy bowiem:

- uniwersalność
- czytelność i łatwość korzystania.

Do kryteriów tych przywiązujemy ogromną wagę ze względu na wymagania naszych programistów. Dostosowując się do istniejącej sytuacji, zrezygnowaliśmy z przekazywania programistom szczegółowej wiedzy o działaniu systemu operacyjnego i języku sterującym, dając gotowe narzędzie współpracy z tym systemem w postaci samodzielnie stworzonych makroinstrukcji.

Makroinstrukcje nasze obejmują wykonywanie programów należących do następujących grup:

- kompilatory języków FORTRAN, PLAN i COBOL
- programy użytkowe napisane w powyższych językach
- programy organizacyjne dla pamięci taśmowej i dyskowej
- programy korekty zbiorów taśmowych i dyskowych
- programy sortowania i łączenia zbiorów
- programy kopiowania zbiorów.

Proces tworzenia tych makroinstrukcji nie jest zakończony, a w miarę potrzeb dołączamy nowe i modyfikujemy istniejące. Aktualnie mamy do dyspozycji:

- 27 makroinstrukcji o zastosowaniu ogólnym, związanych z programami standardowymi
- 1 makroinstrukcję uniwersalną, pozwalającą na wykonywanie szerokiej klasy programów użytkowych
- 38 makroinstrukcji specjalnych dla wykonywania konkretnych programów użytkowych (makroinstrukcje tej grupy są bardzo łatwe w użyciu na skutek maksymalnego ograniczenia liczby parametrów).

Praktycznie wszystkie wykonywane w naszym ośrodku programy są objęte makroinstrukcjami, co w sposób kardynalny ułatwia pracę programistom i znacznie zmniejsza prawdopodobieństwo popełnienia błędów w zdaniach sterujących. Liczba zadań nie wykonanych z powodu błędów języka sterującego wyniosła:

— 10,4% w dwóch pierwszych miesiącach eksploatacji systemu
— 9,0% w dwóch następnych miesiącach.

UZYSKANE EFEKTY

Wdrożenie systemu GEORGE 2 w naszym ośrodku przyniosło wymierne efekty. Najważniejszym rezultatem jest niewątpliwie wzrost efektywności działania instalacji komputerowej (poprawa wykorzystania zasobów sprzętowych i skrócenie średniego czasu obrotu zadania).

Za podstawowe miary efektywności działania systemu komputerowego uważamy:

- średnie wykorzystanie jednostki centralnej
- średni czas rzeczywisty wykonywania zadania.

Pierwsza wielkość jest głównym miernikiem globalnej efektywności systemu, natomiast druga najistotniejszym kryterium, na podstawie którego użytkownik ocenia działanie instalacji. Zmiany wartości tych dwóch parametrów po wdrożeniu systemu GEORGE posłużą nam za syntetyczne mierniki efektów jego zastosowania.

Przez wykorzystanie jednostki centralnej w badanym odcinku czasu rozumiemy iloraz czasu działania jednostki centralnej przez czas rzeczywisty pracy systemu w tym okresie.

Natomiast wykorzystanie jednostki centralnej na potrzeby jednego zadania jest to iloraz czasu działania jednostki centralnej zaangażowanej w rozwiązanie tego zadania przez rzeczywisty czas jego wykonywania.

Czas rzeczywisty wykonywania zadania to — w naszym rozumieniu — odcinek czasu, jaki upłynął od momentu zainicjowania zadania przez system operacyjny do momentu zakończenia jego wykonywania. Do czasu rzeczywistego wykonywania zadania pod kontrolą systemu GEORGE 2 nie wliczamy więc czasu wczytywania danych przez program wejściowy, ani też czasu drukowania wyników przez program wyjściowy.

Taka definicja czasu wykonywania zadania pozwoli na ilościowe określenie bardzo istotnego czynnika — przyspieszenia wykonywania programów na skutek wyeliminowania pracy bezpośredniej wolnych urządzeń zewnętrznych i ręcznego wprowadzania rozkazów przez operatora.

Pominiemy natomiast analizę pełnego czasu obrotu zadania, gdyż na jego wartość składa się wiele czynników organizacyjnych, leżących poza systemem komputerowym. Stwierdzimy jedynie, że średni czas obrotu zadania po zastosowaniu systemu GEORGE 2 zmalał bardzo poważnie (z około jednej doby do kilku godzin).

Tak zdefiniowane wartości związane są następującą zależnością: czas rzeczywisty wykonywania zadania jest odwrotnie proporcjonalny do wykorzystania jednostki centralnej na potrzeby tego zadania.

Aby zapewnić pełną porównywalność wartości zdefiniowanych miar w okresie przed i po wdrożeniu systemu GEORGE 2, konieczne jest obliczenie tych wartości na podstawie tego samego strumienia zadań. Tego rodzaju kontrolowane eksperymenty przy zastosowaniu czy to wybranej próbki zadań rzeczywistych, czy to pewnego strumienia syntetycznego, nie są jednak w naszym ośrodku realne ze względu na zbyt wysoki koszt ich przeprowadzenia. Musimy więc polegać na wynikach uzyskanych w toku codziennej eksploatacji maszyny. Sądzimy jednak, że założenie o niezmienności strumienia obliczeniowego w czasie (stabilności jego podstawowych parametrów) jest w naszym ośrodku uzasadnione. Dominującą część obciążenia maszyny stanowią bowiem programy eksploatowane cyklicznie, a reszta to przede wszystkim kompilacje i uruchomienia niewielkich programów.

Należy stwierdzić, że po zastosowaniu systemu operacyjnego GEORGE 2 zaobserwowano znaczną poprawę wartości wymienionych podstawowych miar.

Średnie wykorzystanie jednostki centralnej przez programy użytkowe wzrosło z 11,1% do 21,0% (1,89-krotnie).

Wartość pierwsza dotyczy okresu 9 dni z 1977 r. (116 godzin czasu rzeczywistego działania maszyny), które wybrano jako próbkę w badaniach nad strumieniem obliczeniowym, wykonanych w naszym ośrodku niezależnie od wdrożenia systemu GEORGE.

Wartość druga jest obliczona z okresu czteromiesięcznego luty — maj 1979 (próbka liczy 2610 zadań). Obliczenia wykonano w pierwszym przypadku na podstawie ręcznej analizy protokołów z pulpitu operatora, w drugim przypadku — w pełni automatycznie na podstawie danych rejestrowanych przez system GEORGE w zbiorze informacyjnym.

Całkowite wykorzystanie jednostki centralnej przy pracy pod kontrolą systemu GEORGE 2 wyniosło 25,5%. Liczba ta zawiera jednak w sobie również wykorzystanie „pozorne" — przez programy systemu operacyjnego.

W zaobserwowanym całkowitym wzroście wykorzystania udział mają trzy czynniki:

1) eliminacja działania bezpośredniego wolnych urządzeń wejścia/wyjścia (czytnika kart, drukarki wierszowej, pulpitu operatora)
2) zastosowanie pracy wieloprogramowej
3) pojawienie się narzutu systemowego (wykorzystania „pozornego").

Stwierdzony wzrost średniego wykorzystania jednostki centralnej jest iloczynem wartości tych trzech czynników. Interesująca jest analiza ilościowego wpływu na ten wzrost poszczególnych czynników, która dała następujące wyniki.

Eliminacja wolnych urządzeń we/wy

Wartość tę wyznaczymy pośrednio, znając globalny wzrost wykorzystania jednostki centralnej oraz wartości czynników 2) i 3). Zgodnie z wcześniejszym stwierdzeniem, szukana wartość jest ilorazem całkowitego wzrostu wykorzystania jednostki centralnej przez iloczyn czynników 2) i 3). Obliczenie daje w wyniku 1,51.

Wieloprogramowość

Przez średni poziom wieloprogramowości rozumiemy ilorazy sumy czasów rzeczywistych wszystkich zadań wykonanych w pewnym okresie do czasu pracy systemu w tym okresie.

Średni poziom wieloprogramowości obliczony na podstawie pełnych danych z czteromiesięcznego okresu luty — maj 1979 wynosi 1,25.

Narzut systemowy

Narzut systemowy obliczamy jako iloraz czasu pracy programów systemu operacyjnego związanej z obsługą strumienia zadań przez czas pracy programów użytkowych tego strumienia. Mamy tu na myśli oczywiście czasy działania jednostki centralnej.

Wykonane obserwacje wielkości narzutu opierały się na próbce przypadkowo wybranych 40 zadań o różnych charakterystykach. Wartość średnia uzyskana z próbki wynosi 22,0%. Pełniejsza analiza tego zjawiska wymagałaby bardzo dużych nakładów pracy, ponieważ dane o narzucie systemowym nie są bezpośrednio rejestrowane przez system operacyjny i muszą być odczytywane z wydrukowanych wyników poszczególnych zadań.

Uzyskaliśmy więc następujące wartości poszczególnych czynników wzrostu wykorzystania jednostki centralnej:

1) eliminacja wolnych urządzeń we/wy	1,51
2) wieloprogramowość	1,25
3) narzut systemowy	1,22

Czynnik 1) możemy również rozumieć jako średni wzrost wykorzystania jednostki centralnej na potrzeby jednego zadania. Wzrost tej wartości o 51% oznacza zmniejszenie średniego czasu rzeczywistego wykonywania zadania o 34%.

Wartość iloczynu czynników 1) i 2) to globalny wzrost wykorzystania jednostki centralnej na potrzeby programów użytkowych. Wartość ta wynosi 1,89. Taki jest więc praktyczny wzrost przepustowości instalacji po zastosowaniu systemu GEORGE 2.

W powyższej analizie pominięto zgodnie z założeniem działanie programów wejściowego (wczytującego dane z kart do zbioru dyskowego) i wyjściowego (drukującego wyniki ze zbioru dyskowego). Wpływ pracy tych programów na wartości rozpatrywanych miar efektywności systemu jest jednak minimalny. Czas jednostki centralnej potrzebny na obsługę tych programów kształtuje się bowiem poniżej 2% czasu rzeczywistego wykonywania programów użytkowych. Spowolnienie działania programów użytkowych na skutek zajmowania jednostki centralnej przez programy wejściowy i wyjściowy ma więc bardzo mały wpływ na wartości rozpatrywanych miar efektywności.

WNIOSKI

Uzyskane wyniki, osiągnięte w ciągu czterech pierwszych miesięcy działania systemu operacyjnego GEORGE 2 w naszym ośrodku, są bardzo korzystne. Możliwość dalszej poprawy efektywności działania tkwi w uzyskaniu wyższej wartości czynnika 2). Przy ustalonym strumieniu wejściowym nie mamy bowiem praktycznie żadnego wpływu na wartości czynników 1) i 3).

Zasobem ograniczającym aktualnie poziom wieloprogramowości w naszym ośrodku są jednostki taśm magnetycznych. Skutkiem faktu, że znaczną część strumienia obliczeniowego stanowią zadania wykorzystujące 3—4 jednostki (na 6 istniejących), krzyści ilościowe z zastosowania systemu GEORGE są mniejsze od faktycznie możliwych.

Przeniesienie niektórych zbiorów taśmowych na dyski powinno niewątpliwie przynieść znaczne efekty. Przypuszczamy, że wzrost wykorzystania dysków przy równoczesnym zmniejszeniu obciążenia pamięci taśmowej przyniesie dalszą poprawę efektywności działania naszego systemu liczącego.

Planujemy, że wdrażane w przyszłości oprogramowanie użytkowe będzie wykorzystywało przede wszystkim pamięć dyskową. Niezbędne jest jednak zachowanie właściwych proporcji między przetwarzaniem dyskowym — a taśmowym, ponieważ nadmierne obciążenie dysków może także powodować odczuwalne zmniejszenie efektywności działania.

Uważamy, że sam system operacyjny dobrze spełnia swoje zadanie, natomiast dla uzyskania jeszcze większych efektów niezbędne są zmiany na poziomie programów użytkowych. Obserwujemy już zresztą niemałą poprawę w dziedzinie stylu i efektywności programowania użytkowego, co uważamy za bardzo istotny pośredni skutek wdrożenia systemu GEORGE 2.

LITERATURA:

[1] System operacyjny GEORGE 1 i 2. Wrocławskie Zakłady Elektroniczne Elwro, 1974

[2] GEORGE 2 Disc-based Operating System. International Computers Ltd., 1976

[3] Madnick S., Donovan J.: Operating Systems. McGraw-Hill, 1974

[4] Hellerman H., Conroy T.: Computer System Performance. McGraw-Hill, 1975

MAREK CIEŚLICKI, KRZYSZTOF GERWIN, RYSZARD PLEWAKO
Centrum Informatyki Gospodarki Morskiej
Gdynia

Zabezpieczenie informacji w systemie rezerwacji miejsc na promach

System rezerwacji miejsc na promach BUKPROM został wykonany na zlecenie Polskiej Żeglugi Bałtyckiej w Centrum Informatyki Gospodarki Morskiej. System ten realizuje następujące funkcje:

- dokonywanie wpisów do list rezerwacji miejsc (tzw. list bukingowych) oraz zmian lub kasowań rezerwacji
- śledzenie bieżącego stanu zajętości promu
- drukowanie list pasażerów, pojazdów i towarów
- ilościowe i wartościowe rozliczanie podróży zakończonych.

Zasadniczym elementem systemu jest wspomniana lista bukingowa. Jest ona tworzona odrębnie dla każdej podróży promu, obejmując rezerwacje na całej trasie, tzn. z uwzględnieniem portów pośrednich. Lista bukingowa uwzględnia również wszystkie rodzaje rezerwacji (rezerwacje dla pasażerów kabinowych i pokładowych oraz dla pojazdów i towarów), a także obejmuje wykaz pasażerów oczekujących na zwolnienie się miejsca w kabinie lub na miejsce dla samochodu. Konstrukcja listy bukingowej pozwala również na wyszukiwanie wolnych miejsc przy rezerwowaniu miejsc kabinowych. Okres utrzymywania zbioru listy bukingowej w systemie wynika z okresu ważności biletu i wynosi około 6 miesięcy. Lista bukingowa jest więc zakładana z 6-miesięcznym wyprzedzeniem w stosunku do podróży promu, a kasuje się ją w niedługim czasie po zakończeniu podróży. Warunkiem usunięcia listy z systemu jest wykonanie rozliczeń. Wynikiem tego jest duża liczba list bukingowych utrzymywanych w systemie (90—200 zbiorów na każdy prom), co jeszcze wzmacnia i tak ostry w systemach rezerwacyjnych wymóg możliwie pełnego zabezpieczenia danych.

ZAŁOŻENIA SYSTEMU ZABEZPIECZEŃ

Po szczegółowej analizie problemu rezerwowania miejsc w systemie BUKPROM przyjęto następujące założenie systemu zabezpieczeń:

1) zabezpieczenia dotyczą wyłącznie zbiorów list bukingowych

2) dane w listach bukingowych powinny być chronione zarówno przed nieupoważnionym dostępem, jak i przed zniszczeniem (np. wskutek awarii sprzętu)

3) system zabezpieczeń powinien poprzez odpowiednie blokady zapewnić ochronę przed efektem tzw. powielania się błędu przy niepewnej pracy sprzętu

4) straty czasu spowodowane koniecznością zabezpieczeń nie mogą prowadzić do istotnego wydłużenia czasu reakcji systemu

5) ze względu na dużą liczbę list bukingowych, niezbędne dla zabezpieczenia informacji kopiowanie i reformatowanie zbiorów musi być sterowane automatycznie

6) odtwarzanie stanu systemu po awarii musi być wykonywane przy możliwie ograniczonych i prostych czynnościach operatora systemu oraz w możliwie krótkim czasie

7) w celu zmniejszenia kosztów i skrócenia terminów uruchomienia systemu konieczne jest wykorzystanie w szerokim zakresie oprogramowania firmowego

8) zabezpieczenie informacji nie może prowadzić do zwiększenia zestawu sprzętu poza zestaw minimalny, wynikający z funkcji czysto użytkowych.

Jak widać z powyższych założeń, na realizację zabezpieczeń danych wpływają nie tylko czynniki o charakterze organizacyjno-technicznym, ale również sprawy kosztów.

System BUKPROM został opracowany na minikomputer DATAPOINT 6600, z możliwością eksploatacji również na modelu niższym tj. DATAPOINT 5500. Minimalna konfiguracja obejmuje: jednostkę centralną z pamięcią operacyjną minimum 48 Kb i wbudowanymi dwoma jednostkami kasetowej pamięci taśmowej, dwie jednostki pamięci dyskowej o pojemności po 25 Mb, monitor operatora oraz drukarkę systemu. Podstawowymi terminalami są krajowe drukarki z klawiaturą typu DZM 180 KSR, pracujące poprzez modemy i linie transmisji danych z szybkością 600/1200 bodów, bądź poprzez konwertery TGF z szybkością 100 bodów. Terminalami pomocniczymi są monitory ekranowe typu DATAPOINT 3600. Terminale są lub będą zainstalowane w krajowych i zagranicznych biurach rezerwacji miejsc na promach PŻB, m. in. w Gdańsku, Kołobrzegu, Warszawie, Sztokholmie, Helsinkach oraz w innych miastach i portach.

ZABEZPIECZENIE PRZED NIEUPOWAŻNIONYM DOSTĘPEM

Zabezpieczenie przed nieupoważnionym dostępem realizowane jest na dwóch poziomach sprawdzania haseł. Hasło na poziomie pierwszym zabezpiecza wejście do systemu, tzn. otwiera terminal do pracy w systemie. Hasła na tym poziomie są indywidualne dla poszczególnych osób. Część hasła jest automatycznie wprowadzana przez program otwierający terminal do pracy. Dzięki takiemu rozwiązaniu nie ma możliwości podszycia się pod inne biuro podróży, a ściślej pod innego agenta mającego wyłączność rezerwowania (bukowania) miejsc na danym terenie. Agent ten może posiadać placówki w różnych miejscach. Można również użyć wyróżnika innej osoby w danym biurze. Hasła na poziomie pierwszym mogą być łatwo uzupełniane lub zmieniane przez operatora systemu.

Hasła na poziomie drugim chronią poszczególne funkcje systemu i dlatego są odrębne dla każdej z tych funkcji. Hasła te również mogą być zmieniane. Na poziomie tym wprowadzono dodatkowe hasło, które zamyka pracę terminala i przeprowadza go z powrotem na poziom pierwszy sprawdzania haseł. Użytkownik ma więc możliwość wymuszenia na innych osobach przejścia przez poziom sprawdzania ich indywidualnych haseł. Oczywiście hasła nie są wyświetlane ani drukowane w czasie wprowadzania.

Elementem, który nie stanowi co prawda bezpośredniego zabezpieczenia, ale ma z nim ścisły związek, jest zbiór zwany logiem wejść, w którym odnotowuje się każde otwarcie i zamknięcie terminala, jak również każde wybranie jakiejkolwiek funkcji systemu. W zbiorze tym są rejestrowane: numer terminala, data i czas z dokładnością do 1 sekundy, nazwisko osoby, symbol funkcji lub tekst oznaczający otwarcie/zamknięcie terminala. Zapisywane są tam również wszystkie nieudane próby wejścia do systemu. Wszystkie te zapisy wykonywane są automatycznie, bez możliwości ingerencji ze strony użytkownika.

Przy realizacji powyższych funkcji systemu wykorzystano firmową koncepcję programów podstawowych terminala ANSWER i MASTER, modyfikując ją w niezbędnym zakresie, natomiast bez większych zmian przejęto programy tworzące zbiór użytkowników oraz log wejść, uzupełniając jeszcze oprogramowanie firmowe selektywnym wydrukiem logu wejść.

Poza oprogramowaniem firmowym wprowadzono jeszcze jeden mechanizm zapisywania śladów. Jest on realizowany w programach użytkowych i polega na automatycznym uzupełnianiu wszystkich wpisów do list bukingowych o datę i czas dokonania wpisu oraz o symbole biura i osoby, pobrane z odpowiednich fragmentów hasła, a więc po kontroli na poziomie pierwszym w programie ANSWER i bez możliwości ich zmiany przez użytkownika.

ZABEZPIECZENIE PRZED ZNISZCZENIEM I PROCEDURA ODTWARZANIA

Pierwszym elementem zabezpieczenia danych jest okresowe kopiowanie całych list bukingowych na rezerwowy nośnik, którym jest pakiet dyskowy. Wykonywane po każdym dniu pracy, kopiowanie to wymagało starannego rozważenia sposobu jego realizacji, ponieważ na każdym dysku znajduje się 100—120 list bukingowych, a sam proces kopiowania musi być połączony z reorganizacją, niezbędną dla minimalizacji obszarów zajmowanych przez zbiory. Aby osiągnąć pełną automatykę kopiowania i reformatowania list bukingowych określono taką strukturę ich nazw oraz taki sposób ich rozmieszczenia w pamięciach dyskowych, ażeby można było przy pomocy programu firmowego FILES automatycznie wykonać i umieścić na dysku wykaz list znajdujących się na danym pakiecie dyskowym. W oparciu o ten wykaz działa procedura kopiowania i reorganizacji. Procedura ta wydaje operatorowi systemu polecenia dotyczące czynności przy zakładaniu pakietów rezerwowych na odpowiednie napędy pamięci dyskowych. Pozostałe operacje przy kopiowaniu wykonywane są automatycznie w oparciu o firmowe programy CHAIN i REFORMAT. Procedurę kopiowania uzupełniono o dodatkową czynność tworzenia tablic indeksowych dla list bukingowych. W rezultacie utworzona po każdym dniu pracy uporządkowana kopia list jest używana w dniu następnym jako dysk roboczy, natomiast dysk roboczy z dnia poprzedniego stanowi rezerwę na wypadek awarii. Uniknięto w ten sposób straty czasu na reorganizację dysku roboczego. Ze względu na kluczowe znaczenie zebezpieczenia danych, system BUKPROM działa na trzech kompletach pakietów — roboczym oraz rezerwowych z dwóch kolejnych dni.

Kolejnym elementem zabezpieczenia danych jest podwójny zapis wszystkich transakcji przez programy użytkowe. Istnieją tylko trzy następujące rodzaje zmian wprowadzanych do list bukingowych:

- zmiany istniejących rekordów bez zmian ich indeksów
- dopisywanie nowych rekordów z dopisaniem nowych indeksów
- kasowanie rekordów wraz z odpowiadającymi im indeksami.

Powyższa klasyfikacja ma charakter ogólny i stanowi jeden z elementów wyjściowych do opracowania procedur odtwarzania. Poza zapisem podstawowym dokonywany jest zapis dodatkowy do zbioru, zwanego logiem transakcji. Każdy zapis w tym logu składa się z nazwy listy bukingowej, na której dokonuje się transakcji, aktualnej daty i czasu zapisu, numeru terminala, rodzaju dokonanej zmiany (modyfikacja, dopisanie lub skasowanie rekordu) oraz z pełnego „tekstu" zapisu. W logu transakcji rekord zerowy wykorzystywany jest do sterowania zapisem do logu oraz do blokowania zapisu do list bukingowych w przypadku wystąpienia błędu.

Firmowy system operacyjny DOS/DATASHARE zawiera możliwość wykrywania i sygnalizowania kilku rodzajów błędów, między innymi tych, które wynikają z przyczyn technicznych jak np. błąd parzystości czy chwilowa awaria dysku. Istnieje również instrukcja języka programowania, pozwalająca wyłapywać większość błędów (instrukcja TRAP) oraz przekazująca sterowanie podprogramowi reakcji na błędy. Jest to więc mechanizm pozwalający na stworzenie w programach użytkowych procedur analizy błędów i ochrony przed ich skutkami. W programach systemu BUKPROM mechanizm ten nie jest jeszcze wykorzystywany. W przypadku wystąpienia błędu podczas pracy dowolnego programu, na terminalu, na którym ten program działa, pojawia się odpowiedni komunikat systemu operacyjnego, a wykonywanie tego programu zostaje zawieszone, powodując przejście do programu sterującego tego terminala (programu MASTER). Program MASTER wykonuje w tej sytuacji następujące czynności:

- zapisuje komunikat z systemu operacyjnego do logu wejść do systemu, dzięku czemu log pracy danego terminala jest pełny
- pozostawia niewyzerowany wskaźnik ostatniej transakcji w zerowym rekordzie logu transakcji (w polu zapisu danego terminala)
- zapisuje do rekordu zerowego logu transakcji nazwę listy bukingowej, w której wystąpił błąd, datę i czas oraz numer terminala sygnalizującego błąd, eliminując tym samym możliwość wykonywania jakiegokolwiek programu przeznaczonego do zapisu transakcji; w tym przypadku pole z numerem terminala stanowi wskaźnik błędu, powodując przejście do programu ANSWER.

Wszystkie programy, mające możliwość zapisu do list bukingowych, rozpoczynają każdą transakcję od sprawdzenia stanu wskaźnika błędu. Jeśli jest on niezerowy, to programy transakcyjne reagują odpowiednim komunikatem i przerywają pracę, a na pulpicie operatora pojawia się komunikat ostrzegawczy. Działają natomiast programy czytające listy bukingowe oraz podsystem przekazywania depesz, pozwalając na wzajemne komunikowanie się użytkowników systemu.

Pierwszą czynnością operatora systemu jest analiza informacji zawartych w logu wejść oraz w logu transakcji. Narzędziem tej analizy może być wprawdzie firmowy program LIST, ale bardziej efektywne okazały się w tym przypadku specjalnie napisane proste programy pomocnicze, przeszukujące i listujące oba logi. W wyniku przeprowadzonej analizy, operator systemu może np. wezwać służbę techniczną i po usunięciu awarii przystępuje do ponownego uruchomienia systemu. W pierwszej kolejności zawsze bada się, czy istnieją w logu transakcji jakiekolwiek uszkodzenia. Podstawowym narzędziem tego badania i ewentualnej korekty jest firmowy program edycji zbiorów FEDIT. Z kolei operator przechodzi do badania, czy w liście bukingowej, przy której wystąpił błąd, występują uszkodzenia. Używa się przy tym programów listujących (firmowego LIST lub niektórych programów użytkowych), a do ewentualnej korekty także programu FEDIT. Jeżeli nastąpiły poważniejsze uszkodzenia struktury zbiorów lub tablic systemowych, niezbędnym staje się zastosowanie firmowego programu naprawiającego REPAIR, a w przypadku wystąpienia w liście bukingowej błędów nieusuwalnych, należy przystąpić do jej odtworzenia. Procedura odtworzenia polega na uruchomieniu programu przejmującego sterowanie pracą, łącznie z wydawaniem odpowiednich instrukcji operatorowi systemu.

Program ten realizuje następujące funkcje:

- odczytuje nazwę listy bukingowej do odtworzenia z rekordu nr 0 logu transakcji (nazwa ta może być także podana przez operatora)
- ustala numer jednostki dyskowej, w której znajduje się dana lista
- żąda założenia odpowiedniego pakietu dyskowego na miejsce pakietu z listą do odtworzenia (pakiet z kopią listy z dnia poprzedniego)
- kopiuje listę z pakietu rezerwowego na pakiet nr 0
- żąda ponownej zamiany pakietu rezerwowego na roboczy
- kopiuje listę z pakietu nr 0 na roboczy i usuwa ją z pakietu 0
- uaktualnia listę na podstawie logu transakcji
- reformatuje i indeksuje odtworzoną listę
- zeruje wskaźnik blokady oraz wznawia pracę systemu instrukcją DSBACKTD.

UWAGI REALIZACYJNE

Przy tworzeniu systemu zabezpieczeń danych przed zniszczeniem, szukano w pierwszej kolejności odpowiedzi na pytanie, jakie dane i jakie zbiory muszą być chronione, uwzględniając w analizie kryteria ważności informacji, aktywności zbiorów oraz posiadanych środków i wymaganego stopnia niezawodności. Analiza taka doprowadziła do wniosku, że podwójnym zapisem należy zabezpieczyć jedynie dane w listach bukingowych. Pozostałe zbiory, jak np. rozkład podróży promów mogą być chronione tylko poprzez codzienne kopiowanie, a do ich ewentualnego uaktualnienia po awarii wystarczą przewidziane w systemie programy korygujące i proste środki organizacyjne jak np. rejestrowanie w odpowiednim zeszycie zmian w rozkładzie podróży promów.

Zagadnienie zabezpieczenia danych oraz odtwarzania systemu po awarii musi być rozwiązane w sposób pełny, z dokładnością do funkcji poszczególnych procedur oraz nazw zmiennych i etykiet, już na etapie projektowania systemu. W przeciwnym przypadku, uzupełnianie już napisanych programów pociąga za sobą znaczne nakłady pracy i może wykluczyć opracowanie dla wszystkich programów jednolitych procedur oraz pociągnąć za sobą nadmierne skomplikowanie ich struktury logicznej.

Przy tworzeniu logu transakcji powstaje zwykle pytanie, w jaki sposób należy chronić sam log. Wydaje się, że pytanie to może uzyskać sensowną odpowiedź jedynie w kontekście konkretnych warunków realizacji i eksploatacji systemu, przy znajomości charakterystyki logu, rezerwy sprzętowej i programów odtwarzania. W systemie BUKPROM log transakcji nie jest chroniony, a jedynie kopiowany na rezerwowy nośnik oraz inicjowany w końcu każdego dnia pracy. Na rozwiązanie takie zdecydowano się po analizie niezawodności systemów o podobnym trybie eksploatacji (między innymi opracowanego przez Centrum Systemu dla biura GAL w Nowym Jorku), a także wobec prostej struktury tego logu i skuteczności programu FEDIT.

Kolejnym problemem są straty czasu na rzecz operacji związanych z zabezpieczeniami. W systemach transakcyjnych jest bardzo czuły punkt, szczególnie wtedy, gdy log transakcji jest wspólny dla wszystkich terminali i dla kilku typów transakcji. W systemie BUKPROM zapisy logu realizowane są w trybie dostępu przypadkowego i ocenia się, że przy każdym zapisie użytkowym strata spowodowana podwójną rejestracją mieści się w przedziale 0,1—0,3 sekundy, co dla programów czysto transakcyjnych w systemie BUKPROM, gdzie występuje 1 do 5 zapisów użytkowych, przy sumarycznym czasie dokonania jednej transakcji rzędu od 0,05 do 1 minuty (wliczając w to oczywiście czas wprowadzania danych), jest stratą możliwą do zaakceptowania.

Bardzo istotnym wskaźnikiem jakości systemu zabezpieczeń jest również czas odtwarzania zbioru po awarii. W systemie BUKPROM występuje specyficzna sytuacja, wynikająca z dużej liczby list bukingowych. Spowodowało to konieczność wpisywania w rekord zerowy logu transakcji danych, które pozwalają na natychmiastową identyfikację listy bukingowej zawierającej błąd. Dzięki temu cała procedura odtwarzania zajmuje czas ok. 15—30 minut. Biorąc pod uwagę faktyczną częstotliwość występowania awarii np. we wspomnianym systemie GAL (średnio jedna awaria na 6 miesięcy) uznano, że taki czas odtwarzania jest do przyjęcia.

Rozwiązując zabezpieczenia w systemie BUKPROM od początku ukierunkowano się na maksymalne wykorzystanie oprogramowania firmowego. Droga taka na ogół nigdy nie jest kwestionowana, jednak nie zawsze pamięta się, że dopasowywanie takiego oprogramowania do konkretnych zastosowań, wymaga posiadania jego źródłowych wersji oraz wysoko wykwalifikowanych specjalistów.

Przy analizie systemu zabezpieczeń warto również poświęcić nieco czasu na możliwość wykorzystania jego efektów ubocznych. W systemie BUKPROM okazało się, że osiągnięto taki stopień dokładności danych w logu wejść do systemu i w logu transakcji, że zbiory te dają pełny i wystarczający materiał dowodowy w przypadku zaistnienia sporów, np. o nieprawidłowo dokonaną rezerwację, a sam log transakcji może być podstawą bardzo prostego rozwiązania wymaganej przez PZB funkcji naliczania wpływów ze sprzedaży biletów z maksymalnie jednodniowym opóźnieniem oraz w podziale na poszczególne krajowe i zagraniczne biura podróży.

WALENTY OSTASIEWICZ
Akademia Ekonomiczna
Wrocław

Szkic z historii programowania

Podobno już Platon powiedział, że człowiek jest stworzeniem najrozumniejszym dlatego, że umie liczyć. Proces liczenia zaś jest ściśle uzależniony od metod zapisu liczb, które nazywane są **numeracjami.**

Jedną z najstarszych numeracji była numeracja attycka, stosowana w starożytnej Grecji. Numeracja ta związana była z tzw. **abakiem,** pierwszym przyrządem ułatwiającym człowiekowi dokonywanie obliczeń, którego wynalezienie przypisuje się Pitagorasowi. Pierwotnie abakiem była niewielka deszczułka z wyżłobionymi rowkami, w których przesuwano pestki od owoców lub kamyki. Później przyrząd stopniowo doskonalono, aż przekształcił się w znane współcześnie liczydła.

Rachunek na abaku po raz pierwszy został opisany przez Boethiusa,[1]) następnie — po wprowadzeniu numerowanych żetonów zamiast kamyków — rachunek przeprowadzony na tym przyrządzie opisał francuski mnich Gerbert[2]). On też zwolenników rachunku na abaku nazwał *abacystami.*

Poza rachunkiem na abaku istniał jeszcze **rachunek na palcach,** przez długi czas powszechnie stosowany i wykładany na uniwersytetach. Po raz pierwszy rachunek ten opisał jeden z europejskich pionierów matematyki, mnich irlandzki V. Beda (ok. 673—735). Opis rachunku znalazł się również w „Arithmetica integrorum" (1620), dziele słynnego polskiego uczonego Jana Brożka (1585—1652).

Oprócz wymienionych dwóch stosowano również wiele innych rachunków. W Indiach np. posługiwano się zupełnie odmiennym sposobem rachowania, „przewyższającym wszystko, co da się opisać". Sposób ten polegał na wykorzystaniu tzw. pozycyjnego systemu numeracji, wynalezionego przez Hindusów w VI w. Począwszy od IX w. **rachunek indyjski**[3]) szeroko stosowano również w krajach arabskich, głównie w Bagdadzie, gdzie ok. 770 roku z inicjatywy kalifa Al-Mamuma zbudowano „Dom Mądrości" *(Bajt al-Hikma)* — pierwszą arabską akademię nauk. Sprowadzono tu najwybitniejszych uczonych, między innymi wielkiego matematyka Al-Chorezmiego.

Oprócz wielu prac astronomicznych i matematycznych Al-Chorezmi napisał książkę o hinduskim systemie numeracji i metodach rachowania w takim systemie. Arabski oryginał tej pracy zaginął, istnieje zaś przekład łaciński z XII wieku, zatytułowany „Algorithmi de numero Indorum", co w tłumaczeniu na język polski znaczy „Algorithmiego o liczbie hinduskiej", czyli „książka Algorithmiego, który pisze o liczbie hinduskiej". Algorithmi to zlatynizowane nazwisko Al-Chorezmiego. Od tego nazwiska **algorytmami** zaczęto nazywać metody rachowania w pozycyjnym systemie numeracji lub w ogóle arytmetykę. Natomiast zwolenników metod rachunku indyjskiego, zwanego też arabskim lub hindoarabskim, zaczęto nazywać **algorystami.**

Mniej więcej w XII wieku rozpoczyna się pewna rywalizacja między algorystami i abacystami. Do zdecydowanego zwycięstwa algorystów przyczynił się Leonardo z Pizy[4]),

[1]) Anicius Manilius Severinus Boethius (ok. 480—524) — filozof i działacz państwowy. Posądzony o udział w spisku, zginął w więzieniu w Padwie

[2]) Gerbert (ok. 930—1003) — jeden z matematyków europejskich X wieku. W 999 roku zostaje papieżem i przyjmuje imię Sylwester I. Ze względu na umiejętności matematyczne posądzono go nawet o konszachty z diabłem. Wg H. Greniewskiego był pierwszym chrześcijaninem, który nauki swe uzyskał od Arabów

[3]) Ponieważ obliczenia przeprowadzano na deskach pokrytych pyłem, nazywano je „pracą z pyłem" *(dhuli-karma)*

[4]) Leonardo z Pizy (1180—1240) — agent handlowy, który podróżując po Bliskim Wschodzie, zetknął się z nauką arabską. W literaturze naukowej znany jest pod pseudonimem Fibonacci (skrót słów Filius Bonacci — syn Bonacciego)

wydając w 1202 roku „Liber abaci". W dziele tym podał zasady przeprowadzania obliczeń na liczbach zapisanych według hinduskiego systemu numeracji, wykazując przewagę metod tego systemu nad rachunkiem na abaku.

Jeśli chodzi o adaptacje nowego systemu numeracji i nowych metod rachunku, Polska przodowała wśród państw europejskich. Pojęcie **algorysta** występuje już w Kronice Kadłubka (1150—1223): „a jeśli masz czas po temu, poradź się algorystów, którzy są biegli w rachowaniu". Poza tym chyba pierwsza w Europie praca arytmetyczna [5]), nazwana algorytmem, powstała w Akademii Krakowskiej. Nie oznacza to jednak, że stosowano wyłącznie metody systemu pozycyjnego. Pierwsza książka arytmetyczna, opracowana w Polsce w 1517 roku, zawierała opis rachunku na abaku, zwanego rachunkiem na liniach lub rachunkiem [6]).

Rachunek za pomocą liczb systemu pozycyjnego, zwany rachunkiem cyfrowym, po raz pierwszy w Polsce opracowany został przez B. Herberta [7]) w 1561 r., w podręczniku „Arithmetica linearis".

Pojęcie **algorytmu** jako określenie nauki arytmetyki przetrwało aż do wieku XVI, kiedy to pojęcia tego zaczęto używać zamiast takich pojęć jak sposób lub metoda.

Po raz pierwszy pojęcia algorytmu jako sposobu rozwiązywania zadań arytmetycznych użył w swoim podręczniku algebry Ch. Rudolff [8]) w 1525 roku. Natomiast G. W. Leibniz [9]) w swych pracach o rachunku różniczkowym z 1684 roku, używa pojęcia algorytmu jako procesu rachunkowego prowadzącego do rozwiązania wszystkich problemów z pewnej klasy zagadnień. Leibniz zamierzał też opracować uniwersalny język, w którym można by było rozwiązywać nie tylko zadania z pewnych klas zagadnień, lecz wszelkie problemy naukowe oraz społeczne poprzez manipulowanie symbolami (rachowanie) według określonych reguł — uniwersalny algorytm dla wszystkich zadań. Pomysł ten zapożyczył co prawda od R. Lulla [10]), który w swym dziele „Ars magna et maxima" (1274) zaproponował po raz pierwszy opracowanie ogólnej metody uzyskiwania wszelkich zdań prawdziwych (nie nazywa jej jednak algorytmem). W tym celu proponuje nawet pewne urządzenie mechaniczne, zwane obecnie „młynkiem Lulla". Jest to zespół kół współśrodkowych z napisami, które poprzez odpowiednie ich obracanie pozwalają uzyskać różne kombinacje napisów.

Podobną filozofię na temat znaczenia algorytmu w matematyce reprezentuje wybitny polski uczony, Józef Maria Hoene-Wroński (1778—1853), który uważał, że całą różnorodność bytów można w sposób algorytmiczny wyprowadzić z jednego prawa, które nazywał prawem stworzenia. Całą matematykę jako naukę badającą formy bytowania świata fizycznego dzieli on na geometrię i algoritmię, a tę ostatnią na teorię i technikę.

Tak więc pojęcie algorytmu zaczęto stosować nie tylko na oznaczanie procesów prowadzących do otrzymywania rozwiązań zadań arytmetycznych, lecz określono tym pojęciem dowolny proces przekształcania dowolnych symboli według określonych reguł.

Warto zauważyć, że człowiek w swej działalności intelektualnej już od najdawniejszych czasów zajmował się również manipulacją symbolami nienumerycznymi według określonych reguł, głównie przy stosowaniu pism tajnych. Przykładowo można tu wymienić anagramy stosowane jeszcze przed naszą erą [11]). Nie zapominajmy, że do XVII wieku nie istniały czasopisma naukowe; toteż w celu uniknięcia plagiatów uczeni zapisywali swe wyniki w postaci anagramów [12]). Pierwszy formalny system manipulacji symbolami został sformułowany na początku wieku XX przez matematyka norweskiego A. Thuego (1863—1922). System ten, zwany **rachunkiem słów,** odegrał ogromną rolę w rozwoju późniejszej teorii programowania [13]).

Poszukiwania metod rozwiązywania zadań prowadzone były bez najmniejszego podejrzenia, że dla niektórych problemów po prostu metody takie nie istnieją. Problemy takie nazywa się algorytmicznie nierozstrzygalnymi. Po raz pierwszy algorytmiczną nierozstrzygalność pewnych zagadnień matematycznych udowodnił K. Gödel w 1931 roku, wykazując, że w dość bogatym systemie sformalizowanym istnieją zdania (twierdzenia), o których nie można powiedzieć, że są prawdziwe lub fałszywe. Od tego czasu zaczęto się zastanawiać co to właściwie jest algorytm. Uściślenia tego pojęcia w większości przypadków były związane z pewną propozycją uniwersalnego języka algorytmicznego. Jednak celem uściśleń nie było opracowanie języka zapisu algorytmu, lecz utworzenie z algorytmu obiektu matematycznego, który podobnie jak liczba, zbiór, funkcja itp. stałby się przedmiotem badań matematycznych. Z chwilą pojawienia się maszyn cyfrowych, które mogą realizować dowolny algorytm, powstała też konieczność opracowania specjalnego języka, w którym można by było zapisywać ten algorytm. Kiedy zaś możliwa była automatyczna realizacja dowolnego algorytmu (w latach czterdziestych obecnego stulecia), powstała konieczość opracowania języka niezbędnego do maszynowego przeprowadzenia obliczeń, nazywanych **rachunkiem komputerowym.** Specjalistów od stosowania takiego rachunku zaczęto w ostatnich latach nazywać **informatykami.**

Przed stosowaniem komputerów jako narzędzi ułatwiających przeprowadzenie rachunków algorytmy można było zapisywać w języku werbalnym (por. cytowany na wstępie algorytm Brożka), natomiast maszyny cyfrowe wymagają języka bardzo sformalizowanego. Najlepszym takim językiem jest język wewnętrzny maszyny, zwany też kodem maszynowym. Jest on niestety niezwykle uciążliwy dla człowieka. Stąd też starano się znaleźć pewien kompromis między wygodą człowieka, a wymaganiem maszyny. Pierwszym ułatwieniem zapisu algorytmów maszynowych było wprowadzenie w 1947 r. przez H. H. Goldstine'a adresów symbolicznych. Mniej więcej w tym samym czasie J. von Neumann [14]) proponuje schematy blokowe *(ang. flowcharts),* pomoce przy układaniu programów maszynowych. W latach 1945—46 K. Zuse [15]) jako pierwszy proponuje pewien maszynowy język algorytmiczny, nazwany Plankalkül, w którym wykorzystywany był tylko jeden typ wartości, a mianowicie bit maszynowy (Ja-Nein-Wert), oznaczony symbolem SO. Zuse w języku tym wprowadza po raz pierwszy znak podstawienia (Ergibt-Zeichen), który najpierw oznacza symbolem = zastępując go później symbolem =>.

---

[5]) Chodzi o „Algorithmus Anno 1397" nieznanego autora; według tej pracy wykładano arytmetykę w Akademii Krakowskiej

[6]) Chodzi o „Algorithmus linealis" Jana z Łańcuta

[7]) Benedykt Herbert (1531—1593) — wielki polski uczony, humanista i matematyk. Po raz pierwszy wprowadza do polskiego piśmiennictwa pojęcie arytmetyki jako nauki „sztuki rachunkowej, którą dawni algorytmem przezwali, a którą możnaby nazwać arytmetyką empiryczną czyli paciorkową" — jak to określił I. Chodynicki

[8]) Christoph Rudolff (1500— ok. 1545), pochodził z Jaworu na Śląsku

[9]) Gotfryd Wilhelm Leibniz (1646—1716) — jeden z najwybitniejszych uczonych wszystkich czasów. Wg N. Wienera był on ostatnim człowiekiem, który opanował całą istniejącą współcześnie wiedzę

[10]) Roman Lull (1232—1316) — kataloński poeta, uczony i misjonarz, który nawracając muzułmanów na wiarę chrześcijańską zapoznał się z nauką arabską

[11]) Anagram oznacza słowo z przestawionymi literami; za wynalazcę anagramów uważany jest grecki poeta i gramatyk Lykophron (3 w. p.n.e)

[12]) Mimo to zdarzyły się nadużycia. Tak np. w celu wykrycia plagiatujących „przyjaciół" Archimedes zapisywał błędne wyniki. Ciekawą historię miał też Ch. Huygens, którego anagram rozszyfrował znany matematyk J. Wallis i podał Huygensowi jako swój wynik (później oczywiście przyznał się do tego)

[13]) Rachunek słów stanowił podstawę formalizacji pojęcia gramatyki, dokonanej w latach pięćdziesiątych przez N. Chomsky'ego — współczesnego lingwistę amerykańskiego

[14]) John von Neumann (1903—1957), uważany jest za jeden z najbardziej genialnych umysłów naszego stulecia. Od 1930 roku przebywał w USA, pracując w słynnym Institute for Advanced Study

[15]) Konrad Zuse, niemiecki inżynier, który nic nie wiedząc o pracach prowadzonych w USA, w 1941 roku skonstruował maszynę cyfrową Z1, a nieco później następne — Z2, Z3. Maszyna Z4 (zainstalowana na politechnice w Zurychu) przez pewien czas była jedyną pracującą maszyną cyfrową w Europie

Wykorzystując idee języka Plankalkül, H. Ruthishauser[16] proponuje w 1951 roku bardziej doskonałą wersję języka algorytmicznego[17]. Oto zapis algorytmu mnożenia macierzy w języku Ruthishausera

Für $i = 1$ (1) n:
Für $k = 1$ (1) n:
$O \Rightarrow h_0$
Für $j = 1$ (1) n:
$h_{j-1} + (a_{ij} + b_{jk}) \Rightarrow h_j$
Ende Index j
$h_n \Rightarrow c_{ik}$
Ende Index k
Ende Index i

Podaje on przy tym metodę automatycznego tłumaczenia wyrażeń arytmetycznych na kod wewnętrzny maszyny. Niezależnie od Ruthishausera w 1953 roku Wilkes[18] proponuje automatyczne tłumaczenie programu zapisanego w adresach symbolicznych na program w adresach bezwzględnych.

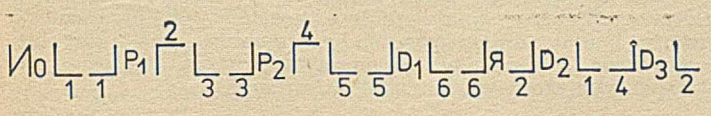
Idee Ruthishausera zostały wykorzystane w 1954 roku przez J. Backusa i H. Herricha przy opracowywaniu języka FORTRAN dla maszyn IBM 701. Z kolei doświadczenia zebrane przy eksploatacji tego języka były podstawą opracowania uniwersalnego języka algorytmicznego[19] ALGOL-58. Ulepszoną jego wersję nazwano ALGOL-60. Zauważmy, że wśród trzynastu autorów tego języka znaleźli się Ruthishauser i Backus.

Mniej więcej w tym samym czasie, bo w 1960 roku, opracowano w Polsce język SAS dla pierwszej krajowej maszyny cyfrowej YXZ. Równocześnie L. Łukaszewicz i A. Mazurkiewicz zdefiniowali język SAKO (System Automatycznego Kodowania) — w istocie swej bardzo zbliżony do języków FORTRAN i ALGOL.

Wszystkie wymienione języki są bardzo podobne do języka powszechnie stosowanego w matematyce, a w szczególności w algebrze. Dlatego zalicza się je czasem do klasy tzw. języków algebraicznych.

Nieco odmienną koncepcję automatyzacji programowania zaproponował w 1953 roku A. A. Lapunow[20], który wprowadził pojęcie operatora i schematu obliczeń.

J. I. Janow dokonał formalizacji pojęcia operatora i zaproponował metody równoważnego przekształcenia ciągu operatorów stanowiących zapis algorytmu. Zapis algorytmu w postaci ciągu operatorów wraz z ich wyjaśnieniem schematycznym stanowi tzw. język schematów logicznych. Ilustracją sposobu zapisu w tym języku może być algorytm szukania największego wspólnego dzielnika:



$$P_1 x > y; \quad P_2 y > x; \quad D_1 z := x; \quad D_2 x_1 := x - y,$$

$$D_3 u : y; \quad y := x; \quad x := u$$

W pierwszym wierszu podany jest schemat logiczny algorytmu, w drugim zaś wyjaśnienie operatorów występujących w tym schemacie. Oto algorytm ten zapisany w języku ALGOL-60:

```
L 1 : if xy then go to L 5; if yx then go to L 6; z := x; stop
L 5 : x := x - y; go to L 1;
L 6 : u := y; y := x; x := u; go to L 1;
```

[16]) Heinz Ruthishauser (1918—1970) — matematyk szwajcarski

[17]) Pojęcie „język algorytmiczny" (algorithmische Sprache) po raz pierwszy wprowadził w 1958 roku Bottenbruch, wykorzystując określenie Ruthishausera (algorithmische Schreibweise)

[18]) M. Wilkes — uczony angielski, pod którego kierunkiem została uruchomiona pierwsza w świecie maszyna cyfrowa z pamiętanym programem (koncepcja von Neumanna). On też po raz pierwszy zaproponował zasady mikroprogramowania

[19]) Na konferencji w Zurichu, gdzie został przyjęty projekt tego języka pod naciskiem przedstawicieli z USA, symbol => zamieniony został symbolem: =

[20]) A. A. Lapunow (1911—1973) — radziecki logik i matematyk

Język schematów logicznych był następnie wielokrotnie korygowany i ulepszany. Na ich podstawie w 1963 roku. D. A. Pospiełow zaproponował język tzw. form warstwowo-równoległych *(Jarusno-parallelnyje formy)*, służący do zapisu algorytmów przeznaczonych do realizacji na systemach wieloprocesorowych. Podobne przeznaczenie mają też tzw. języki macierzowe zaproponowane w 1965 roku przez E. W. Jewrinowa i J. G. Kaserewa.

Wszystkie wymienione wyżej języki, mimo iż są uniwersalne, tzn. umożliwiają zapis dowolnego algorytmu, najbardziej jednak nadają się do programowania problemów naukowo-technicznych, charakteryzujących się stosunkowo małą liczbą danych początkowych i skomplikowanym (matematycznie) sposobem ich przetwarzania.

Typowe problemy z dziedziny administracyjno-ekonomicznej charakteryzują się natomiast dużą liczbą danych (często tekstowych) oraz względnie prostym z matematycznego punktu widzenia przetwarzaniem. Do zapisu algorytmów niezbędnych do rozwiązywania tego typu zagadnień w 1956 roku opracowano język o nazwie MATH-MATIC z translatorem na maszynę UNIVAC. Podobne przeznaczenie miał też język FLOW-MATIC, pochodzący z tego samego okresu, który stał się podstawą opracowania w 1959 roku szeroko obecnie rozpowszechnionego języka COBOL.

LITERATURA:

[1] A computer perspective. Ed. G. Fleck, Harvard University Press, 1973
[2] Bottenbruch Ubersetzung von algoritmischen Formel sprachen. Zeitschrift für mathematische logik und Grundlagen der Mathematik. Vol. 4, 1958
[3] Bauer F. L., Goos G.: Informatik. Spriger-Verlag 1974
[4] Bauer F. L., Woessner H.: The Plankalkue of Konrad Zuse. CACM 1972 nr 7
[5] Bocheński F.: 500 zagadek ze świata arabskiego. Warszawa 1974
[6] Bołgarskij B. W.: Oczerki po istorii matiematyki. Moskwa 1974
[7] Brożek: Wybór pisma w opracowaniu J. Dianni. Warszawa 1956
[8] Cherry C.: Czełowiek i informacja. Moskwa 1972
[9] Chodynicki I.: Dykcjonarz uczonych Polaków. Lwów 1933
[10] Curry H. B.: Osnowanije matematiczeskoj logiki. Moskwa 1969
[11] Dahe O. J., Dijkstra E. W., Hoare C. A. R.: Structurod Programming. Academic Press 1972
[12] Dianni J., Wachułka A.: Tysiąc lat polskiej myśli matematycznej. PZWS 1963
[13] Digitale Informationswandler. Viewag 1962
[14] Gordon G.: Symulacja systemów. Warszawa 1974
[15] Greniewski H.: Elementy logiki formalnej. Warszawa 1955
[16] Higman B.: Srawnitelnoje uzuczenije języków programmirowanija. Moskwa 1974
[17] Historia matematyki (pod red. A. P. Juszkiewicza). Warszawa 1975
[18] Hoene-Wroński J. M.: Wstęp do filozofii matematyki oraz technika algorytmii. Warszawa 1937
[19] Information Processing 71 — Proceedings of the IFIP Congress 71. Ed. C. V. Freim an, North-Holland 1972
[20] Kofler E.: Z dziejów matematyki. Warszawa 1962
[21] Korolew L. N.: Struktury EWM i ich matematiczeskije obiespieczenija. Moskwa 1974
[22] Krynicki N. A.: Rawnosilnoje preobrazowanije algoritmow i programirowanije. Moskwa 1970
[23] Problemy przetwarzania informacji. T. 1 (pod red. R. Marczyńskiego), Warszawa 1970
[24] Problemy przetwarzania informacji. T. 2 (pod red. A. Mazurkiewicza), Warszawa 1974
[25] Ruthishauser H.: Automatische Recheplafertigung bei programmgestenerten Rechenmaschinen, Mitteilungen aus dem Institut für Angewante Mathematik Nr 3, 1956
[26] Sammet J. E.: Programming Languages. History and Future. CACM 1972 nr 7
[27] Struik D. J.: Kratkij oczerk istorii matiematyki. Moskwa 1974
[28] The skyline of information processing. Ed. H. Zemanek, North-Holland 1972
[29] Wielenkin N. J.: Kombinatoryka. Warszawa 1972
[30] Wiener N.: Cybernetyka. Moskwa 1968
[31] Wyczislitielnyje maszyny i myszlenije (red. Feigenaum E., Feldman J.), Moskwa 1967
[32] Zuse K.: Dei Computer mein Lebenswerk. Moderne Industrie 1970

KAZIMIERZ KOWALSKI, ALEKSANDER ZGRZYWA
Biblioteka Główna i OINT
Politechniki Wrocławskiej
Wrocław

Kumulacja danych bibliograficznych w systemie SDI

Racjonalizacja gospodarowania taśmami magnetycznymi zmusza ośrodki obliczeniowe do zmniejszenia liczby taśm zapisanych. Tego rodzaju działania uzasadniają następujące czynniki:
— znaczne koszty zakupu i przechowywania taśm
— ograniczona pojemność magazynów dla przechowywania taśm zapisanych
— konieczność dysponowania znaczną rezerwą taśm niezapisanych dla potrzeb bieżącej eksploatacji.

Znaczne koszty przechowywania zapisanych taśm magnetycznych wynikają zarówno z konieczności ich zabezpieczenia przed skasowaniem informacji, jak i konieczności okresowego kopiowania taśm dla uniknięcia skutków procesów starzeniowych. W celu zabezpieczenia taśm przed skasowaniem lub zniekształceniem informacji pod wpływem różnych czynników zewnętrznych (np. wysoka temperatura, silne pole magnetyczne), przechowuje się je w odpowiednio przystosowanych pomieszczeniach magazynowych, których pojemność z natury rzeczy jest ograniczona [5]. Gromadzenie znacznej rezerwy taśm magnetycznych dla potrzeb bieżącego przetwarzania uzsadnione jest trudnościami ich bieżącego zakupu w warunkach krajowych (artykuł importowany).

Wszystkie wymienione czynniki szczególnie wyraźnie ujawniły się w funkcjonującym w Politechnice Wrocławskiej systemie Selektywnej Dystrybucji Informacji (SDI), w którym wykorzystywana jest duża liczba taśm magnetycznych. Ośrodek ten otrzymuje systematycznie z zagranicy opisy bibliograficzne dokumentów zarejestrowane na taśmach magnetycznych. Te bibliograficzne bazy danych wykorzystywane są w systemie SDI do bieżącego i retrospektywnego wyszukiwania informacji. Zwłaszcza potrzeba prowadzenia wyszukiwań retrospektywnych zmusza do przechowywania wszystkich taśm napływających do ośrodka, a konieczność sporządzania kopii podwaja i tak już dużą liczbę przechowywanych szpul taśmy. Tabela 1 pozwala zorientować się w skali omawianego problemu.

Obecnie przechowuje się już ok. 610 oryginalnych taśm, a przyrost roczny wynosi 202 taśmy. Ponieważ przechowuje się zarówno oryginały jak i kopie, zatem w systemie SDI zgromadzono dotychczas łącznie 1220 taśm magnetycznych.

Tabela 1. Intensywność napływu i wielkość zbiorów systemu SDI w Politechnice Wrocławskiej

Lp.	Nazwa bazy danych	Intensywność napływu taśm (rocznie)	Dotychczasowa liczba taśm	Stopień zapełnienia taśm
1	CAC	52	208	50—60%
2	INSPEC	24	96	50%
3	PASCAL	10	35	75—85%
4	ISMEC	24	84	25—35%
5	SCI	52	156	60%*)
6	INIS	12	12	60%
7	AGRIS	12	12	—
8	ASSISTENT	12	4	—
9	SEBAN	4	3	25%

*) Poszczególne numery bazy SCI mieszczą się na dwóch krążkach taśm

Duża liczba taśm magnetycznych nie tylko wpływa w istotny sposób na zwiększenie kosztów przechowywania, które w chwili obecnej kształtują się na poziomie ok. 100 tys. zł miesięcznie, ale również utrudnia operatywne posługiwanie się tymi taśmami. Zmniejszenie liczby taśm magnetycznych, poza zmniejszeniem kosztów ich przechowywania i ułatwieniem procesu operowania, ma również wpływ na skrócenie czasu przetwarzania w procesie retrospektywnego wyszukiwania informacji.

Z tych właśnie powodów, jak również mając na uwadze powstawanie (w ramach krajowego systemu SINTO) branżowych ośrodków rozpowszechniania informacji, opierających swą działalność na wykorzystywaniu taśm magnetycznych, Biblioteka Główna i OINT Politechniki Wrocławskiej opracowały metody kumulacji danych bibliograficznych na taśmach magnetycznych.

Dr inż. Kazimierz KOWALSKI ukończył studia w 1970 r. na Wydziale Elektroniki Politechniki Wrocławskiej. Pracował w Instytucie Cybernetyki Technicznej PWr. i uzyskał tam w 1974 r. stopień doktora nauk technicznych. Obecnie pracuje w Bibliotece Głównej i OINT PWr. Zajmuje się zagadnieniami organizacji systemów wyszukiwania informacji.

Dr inż. Aleksander ZGRZYWA ukończył studia na Wydziale Elektroniki Politechniki Wrocławskiej w 1970 r. Pracował w Instytucie Cybernetyki Technicznej PWr., gdzie w 1976 r. uzyskał stopień doktora nauk technicznych. Obecnie pracuje w Bibliotece Głównej i OINT PWr. Zajmuje się optymalizacją systemów wyszukiwania informacji bibliograficznej.

SPOSOBY ZMNIEJSZANIA LICZBY TAŚM

Zmniejszenie liczby taśm magnetycznych używanych w systemie SDI osiągnąć można następującymi sposobami:

- ograniczania okresu przechowywania
- rezygnowania ze sporządzania kopii
- zwiększenia gęstości zapisu
- kompresji danych
- kumulacji danych.

Wyznaczenie optymalnego okresu przechowywania dokumentów, a zatem i ich opisów bibliograficznych w systemie SDI jest przedmiotem szeregu publikacji [1, 4]. Na ich podstawie można stwierdzić, że dotychczasowy okres eksploatacji systemu SDI nie upoważnia jeszcze do podjęcia decyzji o wycofaniu ze zbioru najwcześniejszych opisów dokumentów. Występują już natomiast problemy związane z ograniczoną pojemnością magazynów do przechowywania taśm i z rosnącymi kosztami przechowywania. Wydaje się, że te dwa czynniki w decydującym stopniu wpłyną na ograniczenie okresu przechowywania, który w obecnych warunkach można określić na 5 lat.

Drugi z wymienionych sposobów zmniejszenia liczby taśm magnetycznych używanych w systemie SDI może być zastosowany tylko w przypadku istnienia bardzo niezawodnych systemów zabezpieczeń przed zniszczeniem lub uszkodzeniem oryginału. W obecnie eksploatowanych systemach sposób ten jednak nie jest stosowany, a nawet istnieją systemy, w których przechowuje się nie jedną, lecz kilka kopii [3]. W przypadku systemu SDI ograniczenie się tylko do oryginału jest absolutnie niewystarczające. Dotychczasowa praktyka eksploatacji systemu wykazała, że wskutek awarii urządzeń lub błędu operatora należy liczyć się ze stosunkowo częstym uszkodzeniem oryginałów. Koszty powtórnego uzyskania utraconych informacji przewyższałyby w takim przypadku koszty utworzenia i przechowywania jednej kopii. Posiadanie takiej kopii jest konieczne również z tego względu, że oczekiwanie na powtórne otrzymanie informacji w istotny sposób zakłóca rytm przetwarzania i tym samym negatywnie wpływa na efektywność systemu. Natomiast sporządzanie większej liczby kopii jest niecelowe, gdyż kopie takie znajdują się u producenta bazy danych oraz w innych ośrodkach prenumerujących te bazy, a zatem nawet w stosunkowo rzadkim przypadku uszkodzenia zarówno oryginału, jak i kopii zawsze istnieje realna możliwość odzyskania utraconej informacji.

Zmniejszenie liczby taśm magnetycznych poprzez zwiększenie gęstości zapisu informacji uzależnione jest oczywiście od parametrów technicznych dostępnego sprzętu.

Kompresja danych bibliograficznych, jako sposób zmniejszenia liczby taśm magnetycznych używanych w systemie, jest przedmiotem szeregu opracowań [2, 6]. W niniejszym artykule ograniczymy się tylko do stwierdzenia, że w porównaniu z kumulacją jest ona procesem trudniejszym oraz bardziej pracochłonnym i tym samym kosztownym. Kompresja, podobnie jak i inne omówione sposoby, może być stosowana równocześnie z kumulacją danych bibliograficznych.

KUMULACJA DANYCH BIBLIOGRAFICZNYCH

Pod pojęciem kumulacji danych bibliograficznych rozumieć należy proces skupiania na taśmie magnetycznej opisów bibliograficznych pochodzących z kolejnych edycji bazy danych, a także łączenie uzupełniających się informacji w opisie jednego dokumentu w przypadku jego powtarzania się w różnych bazach danych.

Kumulacja danych bibliograficznych, która jest szczególnie potrzebna w systemach retrospektywnego wyszukiwania informacji, może być przeprowadzana:

1) w ramach pojedynczej bazy danych

2) w ramach całego systemu wyszukiwania informacji.

Jak wynika z tabeli 1 stopień zapełnienia poszczególnych taśm z danymi bibliograficznymi wynosi obecnie w systemie SDI średnio 65%. W bieżącym wyszukiwaniu informacji poszczególne taśmy z kolejnymi edycjami bazy danych stanowią odrębne całości i tak też są wykorzystywane. W systemie retrospektywnego wyszukiwania informacji całością taką jest zbiór wszystkich taśm magnetycznych zawierających odpowiednie bazy danych. Ponieważ w tym przypadku nieistotny jest numer edycji, lecz tylko chronologia, nie ma zatem formalnych przeszkód w zapełnianiu taśm w 100%. Przy takim stopniu zapełnienia taśm, nie wnikając w dodatkową możliwość łączenia informacji w ramach opisów jednego dokumentu pochodzących z różnych baz danych, w konkretnym przypadku systemu SDI możliwe jest więc zmniejszenie ogólnej liczby wykorzystywanych taśm o około 1/3.

Proces całkowitego zapełniania taśm magnetycznych pociąga za sobą koszty wynikające z konieczności przekopiowywania taśm. Jeśli jednak operację tę przeprowadzimy w trakcie konserwacyjnego kopiowania taśm, wówczas nie poniesiemy żadnych dodatkowych kosztów. Oczywiście kosztów tych nie ponosimy tylko w przypadku prostego kopiowania w obrębie jednej bazy danych, bowiem wtedy równocześnie z kumulacją możemy przeorganizować zbiór dla celów wyszukiwania retrospektywnego.

Znacznie bardziej złożona sytuacja występuje w przypadku kumulacji danych bibliograficznych w obrębie całego systemu wyszukiwania informacji. Jak wiadomo, bazy danych bibliograficznych zawierają między innymi informacje o publikacjach ukazujących się w określonych czasopismach naukowych. Szereg baz danych korzysta w pewnej części z tych samych czasopism, zatem zawarte w nich informacje są częściowo identyczne. Przeprowadzając kumulację danych bibliograficznych w obrębie całego systemu wyszukiwania, informacje, które powtarzają się w wielu bazach danych zapamiętujemy tylko jeden raz, łącząc z sobą tylko te fragmenty opisu określonego dokumentu, które występują w różnych bazach danych i wzajemnie się uzupełniają, jak np. abstrakt z bazy INSPEC z wykazem literatury cytowanej z bazy SCI.

Tabela 2. Pokrywanie się bibliograficznych baz danych

Nazwa bazy	Liczba tytułów czasopism	Liczba tytułów wspólna z bazą		
		INSPEC	ISMEC	CAC*)
SCI	3 800	1038	167	696
INSPEC	2 700	—	239	440
ISMEC	320	—	—	44
CAC	12 000	—	—	—

*) Uwzględniono tylko 1000 najbardziej popularnych czasopism

Tabela 2 podaje liczbę tytułów czasopism, z których pochodzą opisy bibliograficzne dokumentów w poszczególnych bazach danych oraz liczbę tytułów wspólnych dla głównych baz. Z danych tych wynika, że duża część informacji powtarza się w różnych bazach, co pozwala wnioskować, że zmniejszenie liczby taśm magnetycznych w systemie SDI drogą kumulacji tych samych informacji źródłowych może również przynieść znaczne efekty.

* * *

Przedstawiona kumulacja danych bibliograficznych, jako sposób zapewniający zwiększenie wykorzystania pojemności taśm magnetycznych, ma szczególne znaczenie w systemach SDI korzystających z wielu baz danych. Jego zastosowanie pozwala zmniejszyć koszty eksploatacji systemu oraz skrócić czas wyszukiwania retrospektywnego. Pomimo, że sposób ten odnosi się do zastosowania taśm magnetycznych, będących podstawowym nośnikiem informacji w systemie SDI, może i powinien być on stosowany również w odniesieniu do dysków magnetycznych, preferowanych zwłaszcza w systemach wyszukiwania retrospektywnego. Kumulację należy stosować we wszystkich branżowych ośrodkach przetwarzania informacji bibliograficznych, a w szczególności w ośrodkach produkujących branżowe bazy danych bibliograficznych.

LITERATURA:

[1] Burton R.E., Kebler R. W.: The half-life of some scientific and technical literature. American Documentation 40, Jan. 1960, s. 18—22

[2] Lynch M. F.: Compression of bibliographic files using an adaptation of run-length coding. Inf. Stor. Retr. vol. 9, 1973 s. 207—214

[3] Martin J. T.: Programowanie maszyn cyfrowych w systemach uwarunkowanych czasowo. Warszawa, WNT 1970

[4] Sandinson A.: The use of older literature and its obsolescance. Journal of Documentation 17 (Nr 3) 1971 s. 184—199

[5] Sokołowski A.: Techniczne metody ochrony zbiorów informacji. INFORMATYKA 12, 1975 s. 12—15

[6] Wolf J. G.: Recording of natural language for economy of transmission or storage. Computer Journal vol. 21 Nr 1 1978 s. 42—44

Sprzęt informatyczny na Wiosennych Targach Lipskich 79

Jak co roku Wiosenne Targi Lipskie 79 (11—18 marca) zainaugurowały tradycyjny cykl międzynarodowych prezentacji sprzętu informatycznego.

Doceniając szczególne znaczenie informatyki dla postępu technicznego i rozwoju gospodarczego, organizatorzy Targów Lipskich po raz piąty już starali się wyodrębnić grupę sprzętu informatycznego, prezentowaną pod hasłem „Przetwarzanie danych" (niem. *Fachgruppe DATENVERARBEITUNG*). W ten sposób próbują oni unowocześnić tradycyjny ogólnotechniczny profil imprezy lipskiej, wprowadzając element coraz częściej postulowanej współczesnej koncepcji wystaw specjalistycznych. Niestety, zamierzenie to nadal nie osiąga podstawowego celu, jakim powinna być koncentracja ekspozycji sprzętu informatycznego w sposób ułatwiający zwiedzanie. Sprzęt prezentowany pod hasłem „Przetwarzanie danych" omawiany jest jedynie bardzo ogólnikowo w informatorach targowych oraz występuje w licznych tablicach informacyjnych na terenie wystawowym podających ogólną tematykę ekspozycji poszczególnych hal. O sprzęcie informatycznym mówi się też w cyklu wygłaszanych podczas imprezy odczytów.

Natomiast sama ekspozycja sprzętu informatycznego — podobnie zresztą jak w latach ubiegłych — została rozproszona według oficjalnych danych aż w czterech halach, a mianowicie 12.12, 15, 17 i 18, przy czym jedynie w hali nr 15, gdzie wystawiali gospodarze, ekspozycja ta dominowała wyraźnie nad wyrobami nie związanymi z informatyką. W pozostałych trzech halach sprzęt informatyczny był całkowiecie zdominowany przez wyroby należące do innych branż, jak np. ekspozycja polska w hali nr 18. Przy uważnym zwiedzaniu innych obiektów targowych okazało się, że w rzeczywistości rozproszenie to jest jeszcze większe, ponieważ np. bardzo interesująca wystawa pt. „Komputer w Ameryce", została zlokalizowana w hali nr 16 w ogóle nie ujętej w oficjalnym wykazie 4 hal ekspozycji informatycznej.

OGÓLNA CHARAKTERYSTYKA EKSPOZYCJI

Zgodnie z kilkuletnią już tradycją, dominowała olbrzymia, wielotematyczna i doskonale zaprojektowana ekspozycja kombinatu ROBOTRON. Jak już informowaliśmy w roku ubiegłym, ROBOTRON przejął przedsiębiorstwa kombinatu ZENTRONIK i obejmuje obecnie również branżę maszyn biurowych. Pod względem liczby eksponatów oraz skali oferowanego asortymentu wyrobów Kombinat ROBOTRON okazał się wystawcą większym niż pozostałe kraje RWPG łącznie, co oczywiście nie odzwierciedla rzeczywistego potencjału tych krajów.

Wspomnianą dysproporcję rekompensowały liczne urządzenia peryferyjne produkcji polskiej, radzieckiej, węgierskiej, bułgarskiej i czechosłowackiej, przyłączone do sprzętu NRD, o czym nie we wszystkich jednak przypadkach wspominały napisy informacyjne z parametrami urządzeń. Uwaga ta niestety odnosi się nie tylko do gospodarzy. Tymczasem wydaje się — zwłaszcza, że w bieżącym roku obchodzimy 30-lecie RWPG oraz 10-lecie komputerów Jednolitego Systemu — zarówno wspomniane, jak i wszelkie inne formy kooperacji w zakresie sprzętu informatycznego powinny być bardziej skrupulatnie odnotowywane.

Coraz mniejsze zainteresowanie imprezą wykazują zachodni producenci sprzętu informatycznego. Podstawową przyczyną tego zjawiska jest niewątpliwie wszechstronność oferty NRD. Rozwinięty przez ten kraj przemysł komputerowy zaspokaja obecnie nie tylko potrzeby krajowe, ale może również realizować znaczne zamówienia eksportowe. Dlatego też nawet największe firmy zachodnie nie pojawiają się tu w ogóle lub tylko symbolicznie przypominają o swoim istnieniu (IBM).

W przedstawionej sytuacji relacja z Wiosennych Targów Lipskich 79, będzie poświęcona głównie ekspozycji NRD, którą należy traktować jako interesującą również nas konkretną ofertę handlową tego kraju.

EKSPOZYCJA NRD

Centralnym akcentem ekspozycji NRD był oczywiście komputer drugiej generacji Jednolitego Systemu EC 1055, a właściwie jego jednostka centralna EC 2655, stanowiąca wykładnik osiągniętego poziomu technicznego oraz możliwości przemysłu komputerowego NRD w zakresie systemów dużych.

Producent, który dzięki tej konstrukcji znalazł się w czołówce Jednolitego Systemu, podkreśla, że przy tworzeniu modelu wykorzystano liczne doświadczenia (zwłaszcza w zakresie systemu operacyjnego) z eksploatacji kilkuset egzemplarzy jego poprzednika — komputera RC 1040 (tylko do ZSRR wyeksportowano już 100 egzemplarzy).

Wystawiony egzemplarz EC 1055 demonstrowany był w działaniu na przykładzie systemu rezerwacji miejsc w komunikacji kolejowej. Przyłączone do jednostki centralnej EC 2655 urządzenia peryferyjne ilustrowały wspomnianą już wielostronną współpracę krajów RWPG: pamięci dyskowe EC 5061 (Bułgaria), pamięci taśmowe EC 5017 i czytnik kart EC 6019 (ZSRR), drukarka wierszowa EC 7033 (Polska), perforator kart EC 7014 (CSRS) oraz terminal ekranowy z drukarką mozaikową EC 8564 (Węgry). Konfigurację tę uzupełniało znane już urządzenie do wyprowadzania danych na mikrofilmie EC 7602 produkcji NRD oraz demonstrowany w trybie off-line czechosłowacki pisak x—y Digigraf 1612, zaliczony również do sprzętu Jednolitego Systemu pod symbolem EG 7054.

Kombinat ROBOTRON zademonstrował również działanie produkowanego już od kilku lat systemu minikomputerowego R 4201 w zastosowaniu do rezerwacji miejsc hotelowych w trybie zdalnego przetwarzania danych.

Nowością wśród eksponowanych wyrobów kombinatu ROBOTRON, wskazującą na nadążanie za współczesnym rozwojem mikroelektroniki, były systemy mikrokomputerowe K 1510 i K 1520, które dzięki swej modularnej konstrukcji można dostosowywać do bardzo szerokiego wachlarza potrzeb i warunków różnych użytkowników. O możliwościach zastosowania tych systemów najlepiej świadczą pojemności ich pamięci operacyjnych (RAM) i stałych (PROM), które łącznie wynoszą w tych modelach odpowiednio 16 i 32 K bajtów. Wyposażone w wolniejsze urządzenia peryferyjne (czytnik i perforator taśmy papierowej, drukarka mozaikowa, pamięć magnetyczna kasetowa) mogą być one eksploatowane zarówno we współdziałaniu z większymi systemami komputerowymi, jak i całkowicie niezależnie. Systemy te mają również oprogramowanie podstawowe, stwarzające znaczne możliwości tworzenia oprogramowania użytkowego. Odnosi się to zwłaszcza do modelu K 1520, na potrzeby którego specjalnie skonstruowano urządzenie MRES 20 z własnym oprogramowaniem podstawowym i usługowym, które zapewnia bardzo dogodne, zbliżone do warunków spotykanych w dużych systemach komputerowych, opracowywanie i testowanie programów w trybie konwersacyjnym.

Zgodnie z istniejącymi tendencjami w kombinacie ROBOTRON coraz częściej stosuje się mikroprocesory i pamięci półprzewodnikowe — głównie do

1) budowy programowanych kalkulatorów elektronicznych nazywanych coraz częściej komputerami stołowymi lub biurkowymi

2) rozszerzenia własności funkcjonalnych oraz zwiększenia szybkości i efektywności działania maszyn księgujących i fakturujących oraz urządzeń do przygotowania danych.

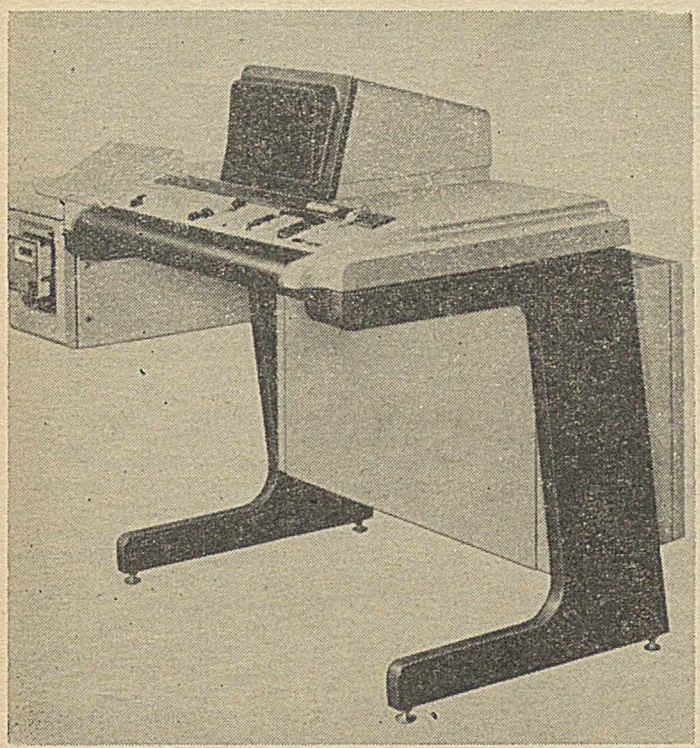
W ramach pierwszego kierunku rozwoju ROBOTRON oferuje 3 modele komputerów stołowych z pamięcią operacyjną 0,25, 0,75 i 1,25 K bajtów: model standardowy K 1001, model K 1002 wyposażony w wymienne karty magnetyczne o pojemności 400 bajtów do wczytywania danych i programów oraz wyprowadzania zawartości pamięci operacyjnej, a wreszcie model K 1003, wyposażony oprócz wspomnianych kart magnetycznych we wbudowaną minidrukarkę.

Drugi kierunek zastosowania mikroprocesorów i pamięci półprzewodnikowych ROBOTRON był reprezentowany na Targach przez całkowicie zmodernizowany asortyment popularnych u nas maszyn tzw. średniej mechanizacji, nazywanej obecnie „średnią automatyzacją". Maszyny te produkowane i sprzedawane były poprzednio przez kombinat ZENTRONIK pod znakiem firmowym „daro", obecnie zastąpionym symbolem „R" (ROBOTRON). Nawiązując do tradycyjnej numeracji poprzednich maszyn tzw. „klasy 170", ROBOTRON oferuje programowane i w pełni konkurencyjne na tle aktualnego poziomu światowego 3 maszyny. Są to:

1) mały automat do fakturowania R 1711, przeznaczony dla małych przedsiębiorstw; oprócz funkcji obliczeniowych maszyny do fakturowania wykonuje on również funkcje programowanych elektrycznych maszyn do pisania (automatów piszących)

2) automat do księgowania i fakturowania R 1720

3) komputer biurowy do księgowania na kontach magnetycznych R 1750 z pamięcią zewnętrzną na dysku elastycznym.



R 1372 — urządzenie klawiaturowe do rejestracji danych na taśmie magnetycznej kasetowej

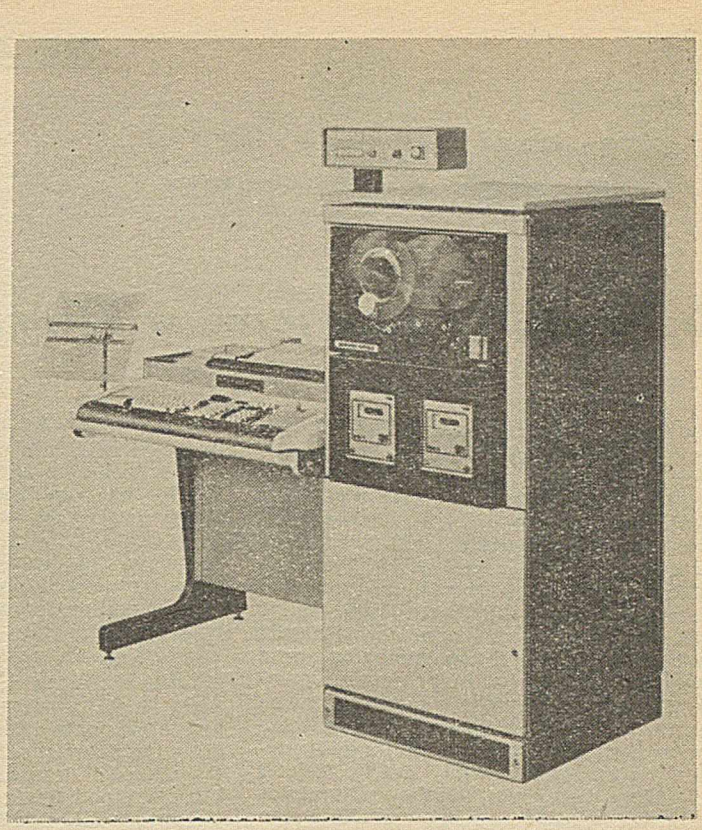
Jeśli chodzi o sprzęt do przygotowania danych, to ROBOTRON oferuje obecnie następujące 3 urządzenia:

1) R 1372 — jednostanowiskowe urządzenie klawiaturowe do rejestracji danych na taśmie magnetycznej kasetowej

2) R 1373 — jak wyżej, lecz z rejestracją na taśmie dziurkowanej

3) R 1375 — czytnik optyczny znaków kreskowych (dokumentów znaczonych) z zapisem danych na taśmie magnetycznej kasetowej.

Ze względu na szeroki zakres stosowania przez ROBOTRON pamięci na taśmie magnetycznej kasetowej (notabene produkcji polskiej — PK-1), oferowany jest również konwerter R 1255. Zapewnia on przenoszenie zapisu z dwóch taśm kasetowych na taśmę półcalową i jest sprzężony z drukarką pozwalającą na sporządzanie protokołów konwersji oraz wypisywanie całkowitej lub częściowej zawartości taśmy magnetycznej.



R 1255 — konwerter zapisu z taśmy kasetowej na taśmę półcalową

Ostatnią wreszcie nowością zaprezentowaną przez ROBOTRON jest drukarka znakowa R 1152, oparta na zasadzie tarczy drukującej. Rozwiązanie to — o szybkości 30 zn./s — jest oczywiście znacznie tańsze od drukarki mozaikowej i powinno w wielu przypadkach zastąpić mniej wydajne elektryczne maszyny do pisania.

INNI WYSTAWCY

Z bogatą i zwartą ekspozycją sprzętu wystąpiły Węgry. Centralnym punktem był tu komputer II generacji Jednolitego Systemu R 11, który zastąpi dotychczas produkowany R 10. Ponadto zaprezentowano specjalizowane systemy mikro- i minikomputerowe oraz terminale ekranowe wspomniane przy omawianiu ekspozycji NRD.

Ekspozycję polską, prawidłowo ukierunkowaną na systemy minikomputerowe i nasze podstawowe eksportowe urządzenia peryferyjne, zlokalizowano jak zwykle w hali 18, gdzie mimo centralnego położenia zginęła w dużym kompleksie innych branż naszego przemysłu.

Ekspozycja bułgarska została rozproszona aż w dwóch halach. Najbardziej interesującym eksponatem było nowe wielostanowiskowe urządzenie do magnetycznej rejestracji danych EC 9003.

Ekspozycja CSRS była jak zwykle bardzo skromna. Prezentowano prawie wyłącznie małe urządzenia peryferyjne i pomocnicze. Nowością była pamięć na dyskach elastycznych ARITMA EC 5075.

Ciekawostką był chiński komputer DJS 131-1, który mimo szybkości 0,5 mln op./s, organizacyjnie i zewnętrznie przypominał maszyny z przełomu lat 1950/60. Jest to wprawdzie znaczny postęp w stosunku do tego, co oglądaliśmy 3 lata temu (I generacja), ale świadczy o olbrzymim zapóźnieniu informatycznym ChRL.

Jakby dla kontrastu zaskoczeniem była ekspozycja Kuby, która zaprezentowała bardzo nowoczesne własne rozwiązania systemu minikomputerowego CID 300 oraz monitora ekranowego CID 72.

Godna odnotowania była także wspomniana wystawa „Komputer w Ameryce". Nie była to oferta handlowa, lecz znakomite przedsięwzięcie popularyzatorskie, adresowane do najszerszych kręgów odbiorców. Warto by pomyśleć o zorganizowaniu podobnej wystawy w Polsce, bardzo często bowiem spotykamy się jeszcze z niedocenianiem lub deformowaniem rzeczywistych funkcji współczesnej informatyki.

Władysław KLEPACZ

WCT/184/K/79

Szkolenie informatyczne w Zjednoczeniu Informatyki

Bardzo szybki rozwój informatyki wymaga nieustającego szkolenia, uzupełniania wiedzy informatycznej, doskonalenia metod pracy. Rozwój bazy technicznej, rozszerzanie konfiguracji systemów komputerowych umożliwia wprowadzanie bardziej wydajnych systemów operacyjnych, to zaś powoduje z kolei pilną konieczność doszkalania zarówno kadry projektowo-programistycznej, jak i technologów oraz operatorów sprzętu. Wprowadzanie oprogramowania wykonanego w nowoczesnej technologii stwarza konieczność nauczania korzystania z niej zarówno kadry własnej ZETO, jak i klientów będących użytkownikami oprogramowania rozpowszechnianego przez ZETO.

ZI jest producentem oprogramowania narzędziowego, za pomocą którego są realizowane konkretne systemy informatyczne, w tym również i poza ZETO. Poważnym zadaniem ZI jest nie tylko dostarczenie narzędzia programistycznego, ale również przekazywanie umiejętności najefektywniejszego jego wykorzystania.

Przykładowo — oferujemy naszym klientom uniwersalny system zarządzania bazą danych RODAN, narzędzie dla tych informatyków, którzy opanowali przynajmniej podstawy systemu operacyjnego OS oraz umiejętność programowania w języku PL/1. Oczywiście przydatność RODAN-u będzie oceniana na podstawie jakości i efektywności opracowanych na jego bazie systemów użytkowych. Wynika stąd konieczność prowadzenia szkoleń na temat projektowania systemów informatycznych w oparciu o RODAN.

ZI jest organizacją, która corocznie zatrudnia znaczną grupę nowych, młodych pracowników, których należy przyuczyć do zawodu lub uzupełnić ich specjalizację informatyczną.

Tak więc Zjednoczenie Informatyki musi prowadzić niezbędne szkolenie informatyczne na wielu poziomach: od elementarnych szkoleń dla nowo przyjmowanych pracowników oraz dla przyszłych użytkowników produktów ZI, poprzez permanentne kursy doskonalące dla własnej kadry oraz klientów, aż do szkolenia najbardziej zaawansowanego przeznaczonego dla współtwórców rozwiązań programowych.

KOORDYNACJA SZKOLEŃ

Na podstawie Zarządzenia Przewodniczącego Komitetu Nauki i Techniki z roku 1971, Ośrodek Badawczo-Rozwojowy Informatyki podjął koordynację działalności szkoleniowej w zakresie informatyki. Obecnie te funkcje OBRI przejęło nowo utworzone Centrum Projektowania i Zastosowań Informatyki ZETO.

Efektem tej koordynacji są porozumienia OBRI z kilkudziesięcioma organizatorami szkoleń informatycznych w kraju, na podstawie których OBRI akceptował programy szkoleń, zatwierdzał skład komisji egzaminacyjnych i zespołów wykładowców, hospitował wykłady, uczestniczył w egzaminach i wydawał zaświadczenia o ukończeniu kursów.

W roku 1975 Minister Nauki, Szkolnictwa Wyższego i Techniki Zarządzeniem nr 19 powierzył Zjednoczeniu Informatyki obowiązki opiniowania programów szkolenia kursowego w kraju w zakresie informatyki. Zadanie to, które na podstawie decyzji Naczelnego Dyrektora ZI, realizowało OBRI, obecnie przejęło również Centrum Projektowania i Zastosowań Informatyki. Z wyżej wymienionego zarządzenia MNSzWiT wynika, że każdy organizator kursu informatycznego ma **prawo** zwrócić się do CPiZI o zaopiniowanie programu, a Centrum ma **obowiązek** taką opinię wydać. Warto zaznaczyć, że wiele resortowych i branżowych ośrodków doskonalenia kadr z tego prawa korzysta.

NIE KURSY LECZ SYSTEM SZKOLENIA

Analizując organizację typowego kursu informatycznego (w innych branżach jest zapewne podobnie) możemy zauważyć kilka ogniw, których wyeliminowanie pozwoliłoby skrócić czas przygotowania kursu, podnieść jego poziom oraz obniżyć koszty.

Z reguły zamawiający kurs (instytucja, przedsiębiorstwo) określa temat, a bezpośredni organizator takiego kursu zamawia ramowy program szkolenia. Po akceptacji takiego programu organizator ten zamawia program szczegółowy, który przesyła do opiniodawców i ostatecznego zatwierdzenia.

Przy braku informacji o istniejących w kraju programach szkoleń wielu organizatorów zleca w tym samym czasie różnym instytucjom bądź indywidualnym osobom opracowanie programów doraźnych kursów o tej samej, bądź bardzo zbliżonej tematyce.

W ten sposób uczestnicy szkoleń w różnych miejscach kraju otrzymują pod tą samą nazwą różną porcję wiedzy, bądź też na różnie nazwanych kursach słuchają wykładów o tych samych zagadnieniach. Przypadkowi wykładowcy, angażowani do przeprowadzenia zajęć na jednym tylko kursie, z uwagi na względnie niskie wynagrodzenie w stosunku do potrzebnego na ten cel nakładu pracy, nie mają dostatecznie silnej motywacji do starannego prowadzenia zajęć, systematycznego analizowania osiągniętych wyników dydaktycznych oraz opracowania specjalnych pomocy dla słuchaczy. Organizator jednorazowego kursu, rozliczając wszystkie poniesione nakłady (plus narzut) na kilkunastu czy dwudziestu kilku uczestników szkolenia, nie jest zwłaszcza w stanie zapewnić im odpowiednich pomocy dydaktycznych.

I tak, mimo ponoszenia znacznych nakładów na tego rodzaju sporadycznych kursach nie można osiągnąć prawidłowych wyników nauczania. Przedstawione niedostatki kursów „jednostkowych" można usunąć poprzez organizowanie szkoleń powtarzalnych stanowiących jednolity system złożony z kursów — elementów (modułów). Programy takich szkoleń powinny być dostępne dla wszystkich zainteresowanych, zaś każdy moduł szkolenia powinien mieć wyraźnie określony zakres oraz bardzo precyzyjne, jednolite wymagania w zakresie przygotowania słuchacza (niezależnie od rodzaju organizatora i miejsca realizacji kursu).

WSSI — WIELOPOZIOMOWY SYSTEM SZKOLENIA INFORMATYCZNEGO

Cechy, o których była mowa wyżej, starano się uwzględnić w Wielopoziomowym Systemie Szkolenia Informatycznego — WSSI. Jest on wynikiem doświadczeń zebranych w czasie wieloletniej działalności szkoleniowej Zjednoczenia Informatyki. W obecnej postaci WSSI funkcjonuje od roku 1976, rezultat pracy badawczo-rozwojowej zapoczątkowanej przez OBRI, a obecnie kontynuowanej i rozwijanej przez CPiZI. Temat tej pracy został sformułowany ze względu na wspomniane na wstępie potrzeby szkoleniowe dając w wyniku pełne uporządkowanie szkolenia przygotowującego do korzystania z oprogramowania rozpowszechnianego przez ZETO.

WSSI obejmuje szeroki zakres zagadnień związanych z oprogramowaniem. Jest on dlatego wielopoziomowy, ponieważ obejmuje zarówno zagadnienia elementarne jak i najbardziej zaawansowane, natomiast określenie „system" uzasadnia fakt, że stanowi on układ wzajemnie powiązanych elementów (modułów szkoleniowych) ukierunkowanych na realizację jednego celu.

Główne cele, które realizuje WSSI, to:

— podniesienie na wyższy poziom szkolenia kursowego realizowanego przez Zjednoczenie Informatyki
— ujednolicenie obowiązujących programów szkolenia (wyeliminowanie przestarzałych, preferowanie najlepszych)

WCT/184/K/79

— ujednolicenie materiałów pomocniczych dla uczestników szkoleń oraz materiałów dydaktycznych dla wykładowców.

Początek funkcjonowania WSSI zbiega się z powołaniem w roku 1976 Ośrodka Szkoleniowego ZETO Łódź, który stał się miejscem pierwszych realizacji większości szkoleń WSSI.

Jak już wspomniano podstawową jednostką WSSI jest moduł szkoleniowy: wyodrębniony, odpowiednio nazwany spójny fragment wiedzy informatycznej którego przyswojenie wymaga około 1 lub 2 tygodni intensywnego szkolenia.

Pełną informację o module szkoleniowym zawiera jego opis składający się z następujących elementów:

symbol — kolejny numer w kartotece P (Programy), uzupełniony numerem wersji

nazwa — kilkuwyrazowe określenie definiujące dokładnie treść modułu

wymagania wstępne — określenie warunków, których znajomość jest niezbędna do zrozumienia treści danego modułu. Może to być wskazanie literatury do wcześniejszego zapoznania się, bądź też — wyliczenie modułów, poprzedzających go tematycznie. Pozwala to jednoznacznie określić położenie danego modułu w zbiorze

program — wyszczególnienie głównych punktów wykładu wraz z rozwinięciem poszczególnych tematów oraz harmonogram ich realizacji

ćwiczenia — opisy ćwiczeń przewidzianych do realizacji i demonstracji komputerowej oraz opisy ćwiczeń do samodzielnego rozwiązywania przez słuchaczy

wykaz literatury — lista publikacji z komentarzem jako rozszerzenie obowiązkowego modułu

materiały szkoleniowe — informacja o pomocach dydaktycznych modułu

sugestie dotyczące dalszej pracy — informacja o następnych modułach szkoleniowych wraz z charakterystyką ich przeznaczenia.

Opisy modułów szkoleniowych udostępniane są przez zakładowe ośrodki INTE sieci ZETO, skutecznie ograniczając zbędne, wielokrotne opracowywanie programów szkoleniowych dla tych samych tematów oraz tych samych zagadnień na różnych kursach.

Dotychczas opracowane opisy modułów WSSI dotyczą głównie systemów operacyjnych dla komputerów Jednolitego Systemu, problematyki baz danych oraz mikroprocesów i mikroprogramowania.

Kartoteka opisów modułów szkoleniowych jest systematycznie zarówno uzupełniana o nowe tematy jak i uaktualniana (nowe wersje wcześniej opracowanych modułów).

Oto niektóre z obecnie oferowanych modułów szkoleniowych:

WSSI-P-01-2 Wprowadzenie w problematykę systemów operacyjnych

WSSI-P-02-1 Dyskowy system operacyjny DOS dla programistów i projektantów

WSSI-P-03-2 System sterowania produkcją STEP

WSSI-P-06-1 Język programowania PL/1

WSSI-P-07-1 Uniwersalny system zarządzania bazą danych RODAN

WSSI-P-11-2 Wprowadzenie do problematyki baz danych

WSSI-P-12-1 System operacyjny OS dla programistów i projektantów

WSSI-P-13-1 System operacyjny OS dla programisty systemowego

WSSI-P-16-1 Programowanie w języku ASSEMBLER/DOS

WSSI-P-18-2 Programowanie w języku PL/1/DOS

WSSI-P-19-1 Programowanie w języku PL/1/OS

WSSI-P-20-1 System operacyjny DOS dla programisty systemowego

WSSI-P-25-1 Wprowadzenie do problematyki komputerów JS

WSSI-P-26-1 Wprowadzenie do wspomaganego komputerem projektowania systemów informatycznych

WSSI-P-27-1 Organizacyjne podstawy działania współczesnych systemów sterowania produkcją

WSSI-P-32-1 Projektowanie systemów informatycznych

WSSI-P-39-1 Programowanie w języku ASSEMBLER/OS

WSSI-P-40-1 Wprowadzenie w problematykę systemów mikroprocesorowych

WSSI-P-77-1 Ewaluacja systemów.

WYNIKI DZIAŁALNOŚCI SZKOLENIOWEJ W 1978 R.

W roku 1978 na 123 kursach zorganizowanych w Zjednoczeniu Informatyki przeszkolono łącznie 2617 osób, z czego 897 stanowili pracownicy ZETO. Najwięcej, bo 1001 słuchaczy, przeszkolono we wspomnianym Ośrodku Szkoleniowym ZETO Łódź. Na zorganizowanych tam 38 szkoleń — 15 dotyczyło produktów ZETO (339 słuchaczy), 11 — języków i technik programowania (316 słuchaczy), 5 — systemów operacyjnych (135 słuchaczy) a 4 seminaria — problematyki baz danych (145 słuchaczy).

Zajęcia w ośrodku łódzkim prowadziło 65 wykładowców, którzy przepracowali łącznie 1968 godzin. Wśród wykładowców 15 było pracowników OBRI, 15 — ZETO Łódź, 15 — innych przedsiębiorstw ZETO, a 20 pochodziło spoza ZETO.

Warto podkreślić, że moduły szkoleniowe WSSI były również podstawą szkoleń prowadzonych przez organizatorów spoza ZI: w Szczecinie, Koszalinie, Poznaniu i innych miastach. Organizatorzy ci stwierdzili, że WSSI pozwala precyzyjnie zaplanować z dużym wyprzedzeniem szkolenie zgodnie z potrzebami różnych kategorii pracowników.

ZAMIERZENIA

Rosnące wciąż potrzeby szkoleniowe zmuszają do zwiększenia liczby bezpośrednich organizatorów szkoleń w ZI. Zarysowująca się specjalizacja ośrodków wymaga uwzględnienia tego faktu w planach rozwojowych tego szkolenia.

Dotychczasowy główny realizator WSSI — Ośrodek Szkoleniowy ZETO Łódź, de facto Centralny Ośrodek Szkoleniowy ZI, będzie musiał rozszerzyć swą działalność o szkolenie wykładowców, ponieważ szkolenia szczebla podstawowego będą stopniowo przesuwane do poszczególnych Zakładów ZETO, co będzie wymagało przygotowania licznych wykładowców.

Aby osiągnąć jeszcze większe korzyści z wdrażania WSSI, należy rozwiązać jeszcze szereg dodatkowych problemów. Najważniejszymi z nich są:

- zapewnienie współpracy wszystkich krajowych organizatorów szkoleń z CPiZI. Kontakt z CPiZI powinien być nawiązywany w momencie przygotowywania się do kursu. Należy wówczas wyjaśnić, czy w przypadku proponowanego kursu, jego tematyka nie obejmuje istniejących już modułów WSSI, lub czy inny organizator nie podjął już tej tematyki. Jeśli w tym ostatnim przypadku to nie nastąpiło, wówczas należy uzgodnić zakres i zlecić wykonanie nowego modułu szkoleniowego
- zorganizowanie kompetentnego zespołu opiniującego opisy modułów szkoleniowych
- zorganizowanie stałego „zasilania" WSSI w nowe tematy
- zorganizowanie systematycznej pracy z wykładowcami realizującymi moduły WSSI.

Wacław PANKIEWICZ
Centrum Projektowania i Zastosowań Informatyki ZETO
Warszawa

I Ogólnopolska Giełda Pomysłów Racjonalizatorskich dla komputerów ODRA 1305

Wśród użytkowanej w kraju grupy średnich i dużych komputerów prawie połowę stanowią maszyny serii ODRA 1305. Wieloletnie doświadczenia w użytkowaniu tych maszyn, a szczególnie najpopularniejszego (również w regionowanego komputera ODRA 1305, dopronie poznańskim) i aktualnie produkowadziły do licznych pomysłów racjonalizujących pracę tego sprzętu i jego oprogramowania podstawowego.

W czasie środowiskowych spotkań informatyków, organizowanych przez Sekcję Maszyn i Systemów Cyfrowych SEP w ośrodkach obliczeniowych Poznania, wielokrotnie podkreślano znaczenie tych usprawnień i mówiono o niedostatecznej informacji o dokonaniach poszczególnych ośrodków.

Wychodząc naprzeciw tym postulatom SMiSC SEP w Poznaniu przy współudziale Komitetu ds. Informatyki OW NOT oraz Środowiskowego Ośrodka Informatyki Politechniki Poznańskiej zorganizowała, w roku obchodów 50-lecia SEP w Poznaniu i 15-lecia informatyki w Wielkopolsce, I Ogólnopolską Giełdę Pomysłów Racjonalizatorskich dla Komputerów ODRA 1305. Giełda odbyła się w Poznaniu w dniu 24 listopada 1978 r.

Celem Giełdy była prezentacja i wymiana pomysłów powstałych w krajowych ośrodkach eksploatujących tę maszynę, a także poszerzenie wymiany nowatorskiej myśli technicznej pomiędzy użytkownikami a producentami urządzeń informatyki.

Dla realizacji tych celów organizatorzy wysłali informacje o Giełdzie do 192 instytucji użytkujących komputery ODRA 1305. W odpowiedzi nadesłano i zgłoszono na Giełdę 144 pomysły, opracowane przez 120 autorów z 32 ośrodków (16,7% zawiadomionych). Wśród zgłoszeń 103 pomysły dotyczyły sprzętu (71,5% ogółu pomysłów), a tylko 41 oprogramowania technicznego i podstawowego (28,5%).

Zestawienie pomysłów w podziale wg rodzaju i oprogramowania zawiera tabela 1.

133 zgłoszone wcześniej pomysły zebrane zostały zgodnie z przyjętym podziałem w katalogu, który został dołączony do materiałów IV konferencji środowiskowej „Informatyka w rozwoju gospodarki Wielkopolski".

Usprawnienia przedstawione na Giełdzie dotyczyły głównie następujących zagadnień:

- poprawiania błędów występujących w pracy sprzętu
- zastępowania dotychczas importowanych elementów urządzeń elementami krajowymi
- ułatwień w obsłudze operatorskiej urządzeń
- oszczędności w zużyciu określonych elementów
- zmian koncepcji istniejących rozwiązań pozwalających uzyskać urządzenie o nowych możliwościach funkcjonalnych
- konstruowania nowych urządzeń.

Pomysły w podziale wg rodzaju urządzeń i oprogramowania

	Miejsce zastosowania pomysłu (rodzaj urządzenia lub oprogramowania)	Liczba pomysłów
Sprzęt	Jednostka centralna ODRA 1305	12
	Jednostka centralna ODRA 1325	2
	Jednostka centralna ODRA 1304	2
	Przełącznica SI	2
	Kontrola operatorska	3
	Pamięć dyskowa	2
	Pamięć taśmowa	21
	Czytnik kart	11
	Czytnik/perforator taśmy	7
	Drukarka wierszowa	19
	Lokalne monitory ekranowe	2
	Urządzenia transmisji danych	5
	Urządzenia przygotowania danych	5
	Urządzenia testujące	8
	Urządzenia klimatyzacyjne	2
Oprogramowanie	Oprogramowanie techniczne	7
	Modyfikacje egzekutorów	6
	Systemy zarządzające	2
	Programy organizacyjne i obsługi	21
	Programy sortowania	2
	Oprogramowanie inne	3

W Giełdzie wzięło udział 89 osób reprezentujących 49 ośrodków obliczeniowych, a także 5 przedstawicieli producentów sprzętu (mgr Fajwel Klejn — BOT ELWRO SERVICE, mgr inż. Kazimierz Mazurkiewicz — BOT ELWRO SERVICE, inż. Mirosław Pasterak — OBRUI MERAMAT, mgr inż. Lech Mirgos — OBRUI MERAMAT, mgr inż. Teresa Kołodziej — ZMP MERA-BŁONIE), oraz przedstawiciel przedsiębiorstwa POSTEOR — Oddział w Poznaniu — mgr inż. Zbigniew Kraśnicki.

W czasie Giełdy nastąpiła konfrontacja twórców pomysłów z producentami sprzętu. Producenci byli zaskoczeni tym, że jest aż tak dużo potrzeb w dziedzinie usprawnień sprzętu. Niektóre z przedstawionych pomysłów zainteresowały producentów szczególnie, jak np. ograniczenie błędów w drukarce DW 325 powstających od wyłączeń i przełączeń kierunku biegu silnika taśmy barwiącej, sygnalizator przesunięcia taśmy barwiącej w drukarce, rozładowanie taśmy magnetycznej w PT 3 i PT 3M oraz wiele innych. Część z prezentowanych pomysłów została w międzyczasie już rozwiązana i zastosowana przez producentów. Niektóre z propozycji dotyczących wykonania nowych urządzeń np. urządzenia do czyszczenia i testowania taśm magnetycznych zdobyły duże uznanie zebranych specjalistów. Pomysłami tymi zainteresował się POSTEOR i po dopracowaniu rozwiązań i wykonaniu prototypów można będzie poszukać producentów tych urządzeń. Wiele pomysłów wzbudziło bezpośrednie zainteresowanie ośrodków obliczeniowych, stając się przedmiotem wymiany już w trakcie Giełdy.

Mimo że Giełda informatycznych pomysłów racjonalizatorskich była pierwszą tego typu imprezą, zarówno jej eksperymentalna forma, jak i zakres rzeczowy, zdobyły sobie uznanie uczestników, którzy podkreślali roboczy charakter Giełdy oraz jej sprawne przeprowadzenie.

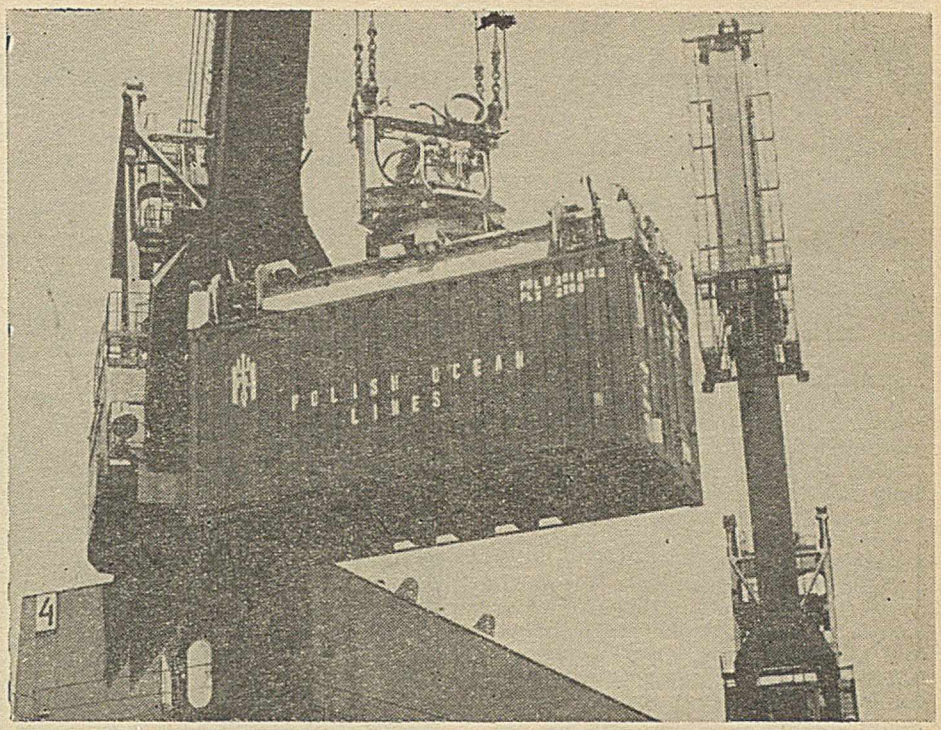
Giełda miała niewątpliwie duże znaczenie gospodarcze, gdyż większość z proponowanych pomysłów przynosi konkretne wymierne korzyści ekonomiczne.

Ponieważ uczestnicy ocenili ją jednomyślnie jako imprezę bardzo potrzebną — należy mieć nadzieję, że inicjatywy tego typu będą kontynuowane.

Inż. Maciej STROIŃSKI
(przewodniczący Kolegium SMiSC SEP w Poznaniu)

Mgr inż. Michał SAJKOWSKI
(sekretarz Giełdy)

Minikomputery wychodzą w morze



Zdj. CAF — Uklejewski

Sporym kosztem ale dość szybko potrafiliśmy zagospodarować pięćsetkilometrowe wybrzeże Bałtyku, jakie uzyskaliśmy po ostatniej wojnie. Mamy dziś parę liczących się w Europie portów i sporą flotę handlową. Flota ta staje się nie tylko coraz większa ale i coraz nowocześniejsza. Najnowsze zaś jednostki pływające dorównują najlepszym na świecie. Wśród nich sporą już grupę stanowią kontenerowce i semikontenerowce. Różnią się one od tradycyjnych statków przede wszystkim tym, że przewożą ładunki w ogromnych metalowych skrzyniach-kontenerach.

Te nowoczesne statki budują dla jednego z dwóch polskich armatorów — Polskich Linii Oceanicznych — stocznie zachodnie. Są to statki drogie. Aby były rentowne — muszą jak najszybciej przemierzać zadane trasy i przewozić jak najwięcej ładunków. Wyliczono, że dzienny koszt utrzymania takiego statku oscyluje wokół połowy miliona złotych. Każdy dzień zbędnego postoju kosztuje więc gospodarkę niemałe pieniądze.

W celu poprawy efektywności, na statkach tego rodzaju instaluje się minikomputery. Mają one znaczny udział w skracaniu rejsów. W transporcie morskim czas trwania rejsu jest najbardziej uzależniony od tempa załadunku i wyładunku towarów. W przypadku kontenerowców czas ten jest dodatkowo zdeterminowany sposobem rozmieszczenia kontenerów. Nie może ono być dowolne. Po pierwsze z uwagi na wymóg bezpieczeństwa, które w głównej mierze wynika z zagwarantowania właściwej stateczności statku. Po drugie, ponieważ statki przybijają do portów pośrednich, kontenery winny być tak rozmieszczone, aby te które winny być wyładowane we właściwym porcie znajdowały się zawsze na „wierzchu".

Do niedawna obliczenia związane z właściwym rozmieszczeniem kontenerów zgodnym zarówno z warunkiem zachowania prawidłowej stateczności statku, jak i ich łatwą dostępnością, wykonywane były bardzo pracochłonnymi metodami „ręcznymi". W ten sposób statek przedłużał swoje pobyty w portach nie tyle wskutek samych operacji załadunku i wyładunku, co w oczekiwaniu na każdorazowe sporządzanie i korygowanie planu rozmieszczenia kontenerów.

Obecnie funkcje sporządzenia takiego planu przejął minikomputer, a w konsekwencji udało się wydatnie skrócić okresy postojów statków w portach.

System zwany „sztauerskim" opracowany został w Ośrodku Badawczo-Rozwojowym Polskich Linii Oceanicznych w Gdyni. Pozwala on optymalizować procesy załadunku i wyładunku tak w portach początkowych, jak i pośrednich. Nie jest to zadanie proste, zważywszy, że występuje w nim sporo zmiennych (ciężary jednostkowe każdego z kontenerów, parametry stateczności, wielkość i kształt ładowni, częstość i wielkość wyładunków w portach pośrednich itd.).

Budowane obecnie w zagranicznych stoczniach kontenerowce wyposażone są w minikomputery WANG, na które ostatecznie zdecydowało się PLO. Na decyzję tę wpłynęły właściwe parametry techniczne i użytkowe tych maszyn, a także istniejący już w kraju serwis firmowy oraz wyposażenie w łatwy do opanowania i użytkowania język programowania BASIC. Zadecydowała również stosunkowo niska cena sprzętu ponieważ instalowane na kontenerowcach jako część ich wyposażenia konfiguracje modelu WANG 2200 kosztują zaledwie ok. 20 tys. $. Nie bez znaczenia jest fakt, że istniejąca możliwość pracy konwersacyjnej czyni WANGA narzędziem stosunkowo łatwym do opanowania przez późniejszych bezpośrednich użytkowników — kadrę oficerską kontenerowców obytą już jako tako ze sprzętem komputerowym w trakcie studiów na Wyższej Szkole Morskiej.

Należy przy okazji zaznaczyć, że system sztauerski był opracowywany w OBR PLO przy czynnym udziale potencjalnych użytkowników.

Przydatność WANGA dla statku nie kończy się na systemie „sztauerskim". Już obecnie wspomniany OBR przy współpracy z zakładami RAWAR opracowuje system NAWIGATOR, którego zadaniem będzie eliminacja zdarzających się nadal kolizji z innymi jednostkami na morzu. System ten umożliwi optymalizację kursu statku na podstawie dostarczanych przez radar pokładowy danych o położeniu statku względem innych płynących jednostek.

W dalszych planach OBR przewiduje się opracowanie systemu wspomagania autopilota oraz systemu związanego z energetyką statku, systemu który pozwoli na uzyskanie oszczędności w zużyciu paliwa.

Te, jak i dalsze plany zastosowania komputerów we flocie handlowej będą możliwe do zrealizowania tym szybciej, im szybciej wzmocni się kadrowo Ośrodek Badawczo-Rozwojowy PLO, w którym obecnie na dobrą sprawę z 35-osobowej kadry nie więcej niż 10% stanowią informatycy.

Uzupełnienia wymaga też sprzęt i to nie tylko informatyczny. Na przykład trudno jest opracowywać system oparty na współdziałaniu WANGA z radarem, gdy w OBR radar jest jak na razie tylko przedmiotem marzeń.

Mimo wymienionych trudności informatycy z Ośrodka Badawczo-Rozwojowego wykonali już sporo nie tylko „dla morza" ale i dla lądu. Na przykład dla potrzeb Ministerstwa Żeglugi opracowano system pozwalający analizować celowość budowy nowych jednostek pływających w kontekście aktualnych i prognozowanych koniunktur w handlu międzynarodowym. W ten sposób armator może w perspektywie 15-letniej określać wielkość, tonaż i rodzaj statków potrzebnych polskiej żegludze, a także analizować oferty stoczni zagranicznych na budowę statków dla Polski.

Obecnie Ośrodek Badawczo-Rozwojowy PLO stanął wobec konieczności realizacji zdalnej transmisji danych i wyników od i do agentów PLO w obecnych portach. Początkowo ma to być transmisja z wykorzystaniem dalekopisów, a w dalszej przyszłości z wykorzystaniem urządzeń końcowych.

Aktywne zainteresowanie dyrekcji Polskich Linii Oceanicznych poczynaniami informatyków z OBR PLO pozwala żywić nadzieję, że ambitne plany zostaną zrealizowane chociaż i to, co już osiągnięto — zasługuje na duże uznanie.

Krystyn BERNATOWICZ

IV Ogólnopolskie Seminarium Metodyczne Informatyki

W dniach 1—8 kwietnia br. odbyło się w Zawoi IV Ogólnopolskie Seminarium Metodyczne Informatyki zorganizowane z inicjatywy Departamentu Informatyki MNSzWiT przez Instytut Informatyki Uniwersytetu Jagiellońskiego w Krakowie. Zgodnie z tradycją poprzednich trzech seminariów, tegoroczne spotkanie poświęcone było głównie zagadnieniom dydaktyki na kierunku „Informatyka" w politechnikach i uniwersytetach. Ponadto tematykę obrad stanowiły zagdnienia kierunków rozwoju prac badawczych prowadzonych w uczelniach, problematyka zaopatrzenia rynku w skrypty i podręczniki z dziedziny informatyki oraz analiza relacji między potrzebami przemysłu informatycznego a modelem absolwenta studiów informatycznych.

Otwarcia seminarium dokonał w imieniu organizatorów dr inż. Jerzy Król (Instytut Informatyki UJ) nawiązując do genezy spotkań wywodzącej się z wyodrębnienia kierunku „Informatyka" w samodzielny kierunek studiów. Organizowane coroczne Seminaria Metodyczne są ważnym forum dyskusji, wymiany doświadczeń i kształtowania perspektyw rozwoju informatyki w środowisku akademickim całego kraju.

Obrady seminarium były prowadzone w pięciu sesjach tematycznych. Ich przebieg był następujący:

Dr Tadeusz Jeleniewski (Politechnika Wrocławska) wygłasza referat „Wykorzystanie środków informatyki do nauczania metodologii projektowania—doświadczenia i perspektywy"
Zdj. Krzysztof Kura

Sesja 1. „Nowe koncepcje programowe i metodologiczne w dydaktyce informatyki"

Obradom pierwszej części sesji przewodniczył prof. dr inż. Jerzy Bromirski (Politechnika Wrocławska). W toku tej części obrad wygłoszono 4 referaty poświęcone organizacji i metodologii kształcenia w dziedzinie informatyki podkreślając znaczenie odpowiedniego doboru tematycznego i chronologicznego przedmiotów w toku studiów, a także omówiono zagadnienia związane z organizacją zespołowej pracy studentów, projektowaniem programów i ich dokumentowaniem itp. Obradom części drugiej przewodniczył prof. dr inż. Adam Sielicki (Politechnika Wrocławska). W części tej wygłoszono 8 referatów poświęconych zagadnieniom komputerowo-wspomaganego nauczania oraz problematyce nauczania przedmiotów związanych z metodami translacji, oprogramowaniem podstawowym i symulacją komputerową.

Sesja 2. „Problemy zaopatrzenia w skrypty i podręczniki"

Obradom przewodniczył red. mgr Feliks Ptasiński (PWN — Warszawa), który poinformował zebranych o aktualnie wydawanych i przewidywanych do druku pozycjach z dziedziny informatyki. W toku dyskusji poruszono m. in. problem niedostatecznych ilościowo nakładów niektórych tytułów, jak też proponowano wydanie tłumaczeń kilku znanych pozycji literatury światowej.

Sesja 3. „Problemy nauczania programowania"

Obradom przewodniczył dr Jan Madey (Uniwersytet Warszawski). Przedstawiono 8 referatów dotyczących problematyki metodologii i organizacji nauczania programowania. Dyskutowano m.in. celowość wyboru określonych języków programowania jako języków podstawowych, jak też zagadnienia związane z eksploatacją systemów komputerowych w nauczaniu programowania. Przedstawiono również pewne nowe, niekonwencjonalne koncepcje organizacji ćwiczeń z przedmiotów związanych z nauczaniem programowania.

Sesja 4. „Prace badawcze i kierunki ich rozwoju"

Obradom przewodniczył doc. dr hab. Stanisław Waligórski (Uniwersytet Warszawski). Wygłoszono 6 referatów o tematyce z pogranicza dydaktyki i prac badawczych. Problematyka referatów dotyczyła języków programowania b. wysokiego rzędu (PROLOG), przetwarzania sygnałów cyfrowych, zagadnień związanych z implementacją systemów cyfrowych oraz budowy metakompilatorów.

Sesja 5. „Potrzeby przemysłu informatycznego a model absolwenta"

Obrady sesji odbywały się pod przewodnictwem doc. dra Zbigniewa Kierzkowskiego (Politechnika Poznańska). Sesję rozpoczęto referatem wprowadzającym w zagadnienia metodologii projektowania systemów informatycznych w warunkach przemysłowych. Następnie odbyła się dyskusja panelowa, która miała na celu przygotowanie wniosków i postulatów dotyczących rozwoju informatyki w uczelniach i gospodarce, w kontekście:

- modelu absolwenta studiów informatycznych

● analizy rozwoju informatyki w warunkach ograniczeń w zatrudnieniu.

Wprowadzenie do dyskusji przygotowali: dr Jan Bobrowski (Dep. Informatyki MNSzWiT), prof. dr Jerzy Bromirski (Politechnika Wrocławska), doc. dr hab. Andrzej Gościński (AGH Kraków), doc. dr Zbigniew Kierzkowski (Politechnika Poznańska), dr Zygmunt Ryznar (ZETO Kraków), dr Stanisław Szczepkowicz (Uniwersytet Wrocławski), doc. dr hab. Stanisław Waligórski (Uniwersytet Warszawski) i dr hab. Jan Zabrodzki (Politechnika Warszawska). W dyskusji poruszono szereg problemów związanych z kształtowaniem profesjonalnych umiejętności absolwentów studiów informatycznych oraz z właściwym ich zatrudnianiem. Zasadnicze wnioski z dyskusji można sformułować następująco:

● absolwent informatyki powinien być dobrze przygotowany do praktycznego wykonywania zawodu i pracy zespołowej

● nie należy rezygnować z kształcenia specjalistów w zakresie teoretycznych podstaw informatyki, zwłaszcza w uniwersytetach

● absolwenci powinni być lepiej niż obecnie przygotowani w zakresie nowych istotnych działów informatyki (np. baz danych)

● w warunkach ograniczeń w zatrudnianiu stwierdza się obserwowany powszechnie brak rotacji kadry na stanowiskach informatycznych, często zajmowanych przez osoby bez odpowiednich kwalifikacji zawodowych.

Zamknięcia seminarium dokonał, w imieniu Departamentu Informatyki MNSzWiT, dr Jan Bobrowski, oświadczając, że Departament z zainteresowaniem i zadowoleniem śledzi czwarte już tego rodzaju spotkanie ogólnopolskie, które spełnia ważną rolę integrowania środowiska informatycznego w ośrodkach akademickich. Mówca poinformował zebranych, że organizacji przyszłorocznego, piątego seminarium podejmuje się Akademia Górniczo-Hutnicza w Krakowie.

W imieniu organizatorów podsumowania seminarium dokonał dr inż. Jerzy Król zwracając m.in. uwagę na zaznaczającą się już wyraźnie linię tematycznej ewolucji spotkań, na których przeważające początkowo zagadnienia dydaktyczno-programowe uzupełniane są w coraz większym stopniu problematyką badawczą.

W obradach IV Seminarium wzięło udział ponad 80 uczestników z 20 krajowych ośrodków informatyki (12 ośrodków uczelnianych oraz 8 pozauczelnianych).

mgr inż. Lech PROFELSKI
Instytut Informatyki UJ

Teletransmisja dla fabryki domów

Łódzki Kombinat Budowy Domów (obecnie Łódzki Kombinat Budowlany „Zachód") stosunkowo wcześnie, bo już w 1975 r. zaangażował się w nowoczesne korzystanie z informatyki. Nowoczesność polegała tu na stosowaniu informatyki na co dzień, a nie w dość długim cyklu przetwarzania tradycyjnego, jaki wynika na ogół z aktualnych możliwości usługowych ośrodków ETOB czy ZETO, warunkowanych okresowym dostarczaniem dokumentów źródłowych do ośrodka obliczeniowego.

Trudno teraz osądzać, na ile szczęśliwy przypadek, a na ile zapobiegliwość zespołu informatyków w dyrekcji kombinatu spowodowały, że na ich terenie został zainstalowany minikomputer ICL 2903, z dyskami i monitorowymi urządzeniami do zbierania i konwersacyjnego przetwarzania danych. Komputer ten był własnością „Skórimpexu", który po prostu nie miał na czas gotowego pomieszczenia do zainstalowania i eksploatacji tego sprzętu. Kombinat budowlany takie pomieszczenie zaoferował na zasadach wspólnego korzystania z komputera.

Grupa entuzjastów sprawy — wciąż jeszcze, niestety, takimi kategoriami określeń wypada operować — kierowana przez mgr. Edwarda Bogulewskiego, szybko i skutecznie wykorzystała możliwości i oprogramowanie podstawowe komputera, przysposabiając go do operatywnego kierowania produkcją wytwórni prefabrykatów budowlanych, zwanej potocznie fabryką domów. Ponieważ jednak kombinat nie tylko wytwarza elementy, ale głównie buduje domy, więc systemem informatycznym objęto także codzienną kontrolę stanu realizacji budynków oraz programowanie dostaw elementów na poszczególne budowy.

Uśmiech losu trwał jednak niewiele dłużej niż dwa lata. Jakiś-tam „gentleman-agreement" został przez budownictwo dotrzymany i korzystający dotąd ze wspomnianej gościny Kombinatu „Skórimpex" dorobił się w 1978 r. własnego pomieszczenia, po czym po prostu zabrał swój komputer. Wprawdzie kombinat dalej z niego korzystał, ale już na zasadach raczej tradycyjnych, tzn. bez teletransmisji, a więc ze wszystkimi utrudnieniami powodowanymi oddaleniem sprzętu od przedsiębiorstwa.

Dyrekcje Kombinatu oraz Łódzkiego Zjednoczenia Budownictwa podjęły wówczas nader energiczną próbę zakupu podobnego sprzętu na własność, co jest godne podkreślenia, choćby dla pokrzepienia niepoprawnych fanatyków złotówkowego obliczania tego wszystkiego, co efektywnością w gospodarce narodowej nazywają. Trudno bowiem przypuścić, że łódzki kombinat i łódzkie zjednoczenie kalkulacji owej „efektywności" nie przeprowadziły. Inicjatywa ta została jednak zważona w bardziej brutalnym ujęciu globalnym.

Koszty komputera ICL 2903 w konfiguracji wypróbowanej przez Łódzki Kombinat są rzędu setek tysięcy dolarów. Fabryk domów jest w Polsce ponad sto pięćdziesiąt, a dąży się przecież do informatycznych rozwiązań powtarzalnych. I z tych właśnie względów reakcja kierownictwa resortu i Centrum ETOB na konstruktywne nawet łódzkie apetyty nie mogła być inna niż negatywna.

Niemniej jednak należało w miarę możliwości wykorzystać istniejący dorobek kombinatu, a zwłaszcza zdobyte już umiejętności i doświadczenie jego informatyków. Centrum ETOB nadało więc priorytet takiemu wyposażeniu łódzkiego przedsiębiorstwa ETOB, aby kombinat można było obsługiwać ODRĄ 1305 w trybie codziennego, zdalnego przetwarzania wsadowego. Istniało ku temu ułatwienie w postaci zgodności oprogramowania komputerów ICL 2903 i ODRA 1305. W Kombinacie zainstalowano na razie dwa urządzenia końcowe typu DZM 180 KSRE, natomiast w ETOBIE — multipleksor MPX-325 z odpowiednimi urządzeniami pośredniczącymi dla teletransmisji poprzez łącza komutowane.

Zdalna, użytkowa eksploatacja systemu dyspozytorskiego dla fabryki domów stała się faktem z początkiem maja br. Nie jest to jeszcze pełne przeniesienie tego co poprzednio już zrobiono, a zwłaszcza tego, co zamierzano zrealizować na własnym ICL 2903. Niełatwo bowiem przechodzi się ze sprzętu za setki tysięcy dolarów na sprzęt za setki tysięcy złotych, zwłaszcza gdy dostępność tego ostatniego jest w znacznym stopniu reglamentowana. Niemniej jednak, już w obecnej skromnej konfiguracji odbywa się codzienne zdalne wprowadzanie do ODRY danych źródłowych równoważnych pojemności informatycznej 500—700 kart dziurkowanych, oraz po 2—3 godzinach — zdalna retransmisja wydruków przetwarzania, dotyczących:

- stanu zapasów prefabrykatów w magazynach fabryki domów
- raportów produkcyjnych za ostatnią dobę
- raportów o ilości i asortymencie prefabrykatów dostarczonych na poszczególne budowy i wbudowanych w poszczególnych obiektach
- dyspozycje dostaw prefabrykatów na budowy.

Przekazywanie danych z kombinatu do ośrodka obliczeniowego odbywa się w godz. 8.00—9.00, natomiast przesyłanie wyników przetwarzania do kombinatu w godz. 10.00—11.00, na razie z szybkością 200 bodów. Wkrótce ma być uruchomiona transmisja 600-bodowa. Już obecnie, w wyniku odpowiednich zabiegów programowych, uzyskano około trzykrotnie większą szybkość przesyłania tabulogramów, niż przewidują to rozwiązania standardowe.

W opisywanym tu skrótowo przedsięwzięciu, polegającym na zastosowaniu krajowych substytutów sprzętowych systemu ICL 2903, a jednocześnie opracowaniu rozwiązań powielarnych dla zarządzania fabrykami domów, szczególny udział mają: zespół ETOBU Łódź, a zwłaszcza Grzegorz Janicki (który swego zaangażowania nie ograniczał do funkcji zastępcy dyrektora przedsiębiorstwa), Roman Ferenc i Jerzy Ignaczak, oraz wspomniana grupa informatyków kombinatu, z której należy wymienić jej kierownika Edwarda Bogulewskiego i Janinę Zdziubany — najaktywniejszą zawodowo kobietę w tym przedsięwzięciu.

Odrowski multipleksor umożliwia oczywiście dalszy rozwój zdalnego trybu przetwarzania danych na sprzęcie krajowym, najbardziej w sieci ETOBU rozpowszechnionym. Najbliższe konkretne zamierzenia dotyczą powtórzenia rozwiązań użytych w kombinacie łódzkim na rzecz podobnego kombinatu w Piotrkowie Trybunalskim oraz zdalnej obsługi dwóch przedsiębiorstw Zjednoczenia „Elektromontaż" (w Łodzi i Nowej Hucie), przodującego w resorcie budownictwa pod względem zastosowań informatyki, a obsługiwanego właśnie przez ETOB w Łodzi.

Obecnie dla fabryk domów opracowuje się znacznie wzbogacone repertuaru operatywnych usług informatycznych, zwłaszcza w dziedzinie technicznego przygotowania produkcji oraz powiązania systemu dyspozytorskiego z gospodarką środkami produkcji.

Przewiduje się, że w jednym z najbliższych numerów „INFORMATYKI" zostaną opisane bardziej szczegółowo wyniki prac i praktycznych doświadczeń związanych z przedstawionymi tu systemami, a zwłaszcza te ich aspekty, które mogą zainteresować szerszy krąg informatyków.

Z okazji święta Odrodzenia została wręczona doroczna nagroda (II stopnia) Ministerstwa Budownictwa i PMB za opracowanie i wdrożenie zinformatyzowanego systemu dyspozytorskiego w Łódzkim Kombinacie Budowy Domów (jeszcze na etapie poprzedzającym teletransmisję). Otrzymał ją zespół w składzie: Jerzy Tomaszewski, Edward Bogulewski, Andrzej Gutowski, Janina Zdziubany, Andrzej Janik, Ryszard Wiśniewski, Bogdan Gadek, Ryszard Kurzyjamski, Roman Ferenc. System został zrealizowany w Łódzkim Kombinacie Budowlanym „Zachód", przy współpracy Instytutu Organizacji i Zarządzania Uniwersytetu Łódzkiego, Przedsiębiorstwa Handlu Zagranicznego „SKÓRIMPEX", Łódzkiego Przedsiębiorstwa Informatyki Przemysłu Budowlanego ETOB oraz Poznańskiego Kombinatu Budowlanego. (wład)

ETOB Lublin – bliżej samodzielności

W grudniu 1974 roku Centrum ETOB zawarło z Lubelskim Zjednoczeniem Budownictwa porozumienie o utworzeniu w Lublinie ośrodka obliczeniowego", który docelowo przekształcony zostanie w Lubelskie Przedsiębiorstwo Informatyki Przemysłu Budowlanego ETOB", a do tego czasu będzie filią ETOBU w Warszawie. Lubelskie Zjednoczenie zobowiązywało się w porozumieniu, że do końca 1975 r. udostępni dla ośrodka odpowiednie pomieszczenia w budowanym właśnie dużym biurowcu dla „Miastoprojektu", natomiast Centrum ETOB wyposaży ten ośrodek w sprzęt informatyczny. Zwiększany z roku na rok priorytet budownictwa mieszkaniowego, za które LZB jest odpowiedzialne przede wszystkim, spowodował, że lokalowy happy-end dla ośrodka ETOB nastąpił dopiero na początku 1979 roku.

Przyznać jednak trzeba, że mimo niezwykle trudnych warunków lokalowych opóźnienie to miało dla ETOBU także pewne zalety. Spokojniej przebiegał wydłużony etap „raczkowania" ośrodka, krzepła jego kadra, a przedsiębiorstwa budowlane miały więcej czasu na stopniowe oswajanie się z zastosowaniami informatyki. Zwłaszcza tej ostatniej sprawie kierownik ośrodka, mgr Ryszard Burski i jego zespół poświęcali szczególnie dużo uwagi.

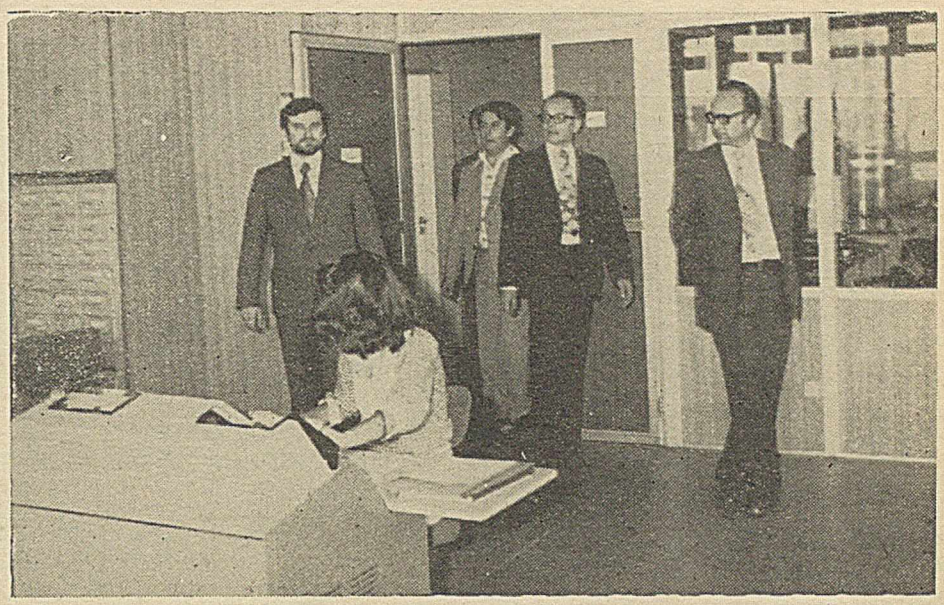
Organizowano liczne szkolenia i kursy, wykorzystując do tego doświadczenia i możliwości macierzystego przedsiębiorstwa ETOB w Warszawie, którego dyrekcja od samego początku nową filię otaczała należytą opieką.

W 1975 roku uruchomiono w Lublinie kilkunastoosobową stację przygotowania danych, co miało następujące trzy istotne zalety:

- zaznaczało obecność informatyki w lubelskim budownictwie
- ułatwiło bezpośredni kontakt informatyków z użytkownikami usług informatycznych sieci ETOB
- odciążało macierzysty ETOB w Warszawie od tworzenia maszynowych nośników danych, umożliwiając dzięki temu rozszerzanie usług informatycznych na Lubelszczyźnie.

W tym okresie lubelski ośrodek był więc głównie pośrednikiem pomiędzy użytkownikami z własnego terenu, a przetwarzającym ich dane ETOBEM w Warszawie.

Zwiększyła się również liczba użytkowników informatyki. Pod koniec 1978 r. ośrodek pośredniczy już w obsłudze 37 przedsiębiorstw w zakresie gospodarki materiałowej, a 11 — w ewidencji i rozliczania środków trwałych. Zjednoczenie Budownictwa Przemysłowego WSCHÓD w Lublinie jest jednym z głównych użytkowników systemu BAZA, wspierającego planowanie, nadzorowanie i rozliczanie produkcji budowlano-montażowej. Siedemnaście przedsiębiorstw regionu lubelskiego korzysta z informatyki dla ewidencji i rozliczeń pracy maszyn budowlanych. Pojedyncze przedsiębiorstwa są użytkownikami innych systemów informatycznych.



Lubelski ośrodek ETOB zwiedzają w dniu jego otwarcia (od prawej): dyr. M. Grochowski, dyr. M. Mazurski oraz mgr M. Woliński z KW PZPR — w towarzystwie kierownika ośrodka, mgr. R. Burskiego

Warszawskiemu ETOBOWI przybywało w tym czasie klientów również z innych regionów, co stworzyło trudności w pełnym zaspokajaniu potrzeb ośrodka lubelskiego (w przeliczeniu na karty dziurkowane, ilość tych danych w ciągu ostatnich trzech lat wzrosła dwunastokrotnie).

W początku bieżącego roku niebezpieczeństwo to zostało szczęśliwie zażegnane oddaniem do użytku ośrodka lokalu z prawdziwego zdarzenia oraz zainstalowaniem komputera ODRA 1305, niestety na razie w dość skromnym zestawie, zwanym przez producenta — elegancko — konfiguracją podstawową.

W dniu 17 maja 1979 r. odbyło się oficjalne otwarcie nowego etapu działalności lubelskiego ośrodka ETOB, w nowej, dostosowanej w pełni do potrzeb informatyki siedzibie przy ulicy Wieniawskiej 14. W skromnej uroczystości otwarcia uczestniczyli przedstawiciele lubelskich władz administracyjnych i partyjnych, dyrektor Departamentu Ekonomiki i Finansów Ministerstwa Budownictwa i PMB (sprawującego ministerialną pieczę nad Centrum ETOB) mgr Marian Mazurski, naczelny dyrektor Centrum ETOB doc. dr inż. Marek Grochowski ze swym zastępcą ds. projektowania i rozwoju mgr. inż. Tadeuszem Niemunisem, a ze strony Lubelskiego Zjednoczenia Budownictwa — jego dyrektor ds. ekonomicznych — mgr Wacław Achler, skuteczny orędownik rozwoju zastosowań informatyki w swym Zjednoczeniu oraz główny księgowy Zjednoczenia WSCHÓD — mgr Aleksander Gromaszek. Obowiązki gospodarzy pełnili: dyrektor warszawskiego ETOBU mgr Marian Uraz, jego zastępca ds. eksploatacji mgr inż. Wojciech Iwanowski oraz kierownik lubelskiego ośrodka mgr Ryszard Burski.

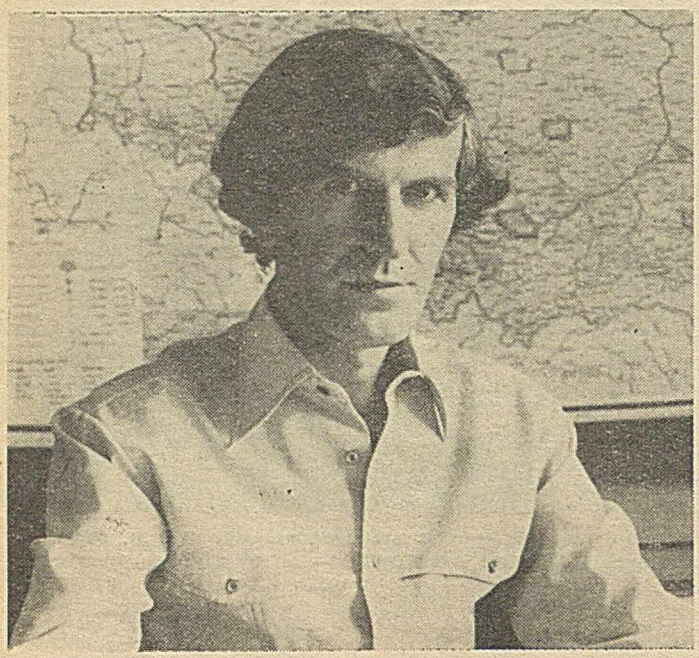
Informatykom oddano do dyspozycji dwie kondygnacje biurowca, o powierzchni ok. 1300 m², w tym sala komputerowa liczy sobie 350 m², a łączna powierzchnia technologiczna wynosi 800 m². Całość, łącznie z wyposażeniem informatycznym i biurowym, kosztowała ok. 50 mln złotych, z czego 32 miliony przypada na sprzęt do przetwarzania danych.

W nowoczesnej scenerii wnętrz sprzęt komputerowy prezentuje się raczej skromnie: ODRA 1305 z pamięcią operacyjną 64 K, 10 jednostek pamięci taśmowej PT 3, drukarka wierszowa wspierana drukarką mozaikową DZM 180. Oczywiście od zaraz przydałaby się druga drukarka wierszowa, większa pamięć operacyjna i pamięci dyskowe.

Na to trzeba jednak poczekać, oby tylko nie długo. Niemniej jednak uzyskanie własnego komputera stanowi dla lubelskiego ośrodka milowy krok w kierunku wzrostu samodzielności działania, zwiększenia operatywności świadczenia usług informatycznych dla budownictwa na terenie województw lubelskiego, chełmskiego, zamojskiego i bialsko-podlaskiego (taki jest „przydział" rejonizacyjny dla ETOBU w Lublinie) oraz — nie ma co ukrywać — przynosi wyraźną ulgę dla macierzystego przedsiębiorstwa w Warszawie. Natychmiast po uruchomieniu komputera przystąpiono do przejmowania z Warszawy eksploatacji systemów informatycznych, głównie dla przedsiębiorstw podległych dwom lubelskim zjednoczeniom: budownictwa ogólnego oraz budownictwa przemysłowego. Stopniowo usługi informatyczne będą rozszerzane również na inne przedsiębiorstwa resortu budownictwa i przemysłu materiałów budowlanych, m. in. na cementownie regionu lubelskiego.

Krystyn BERNATOWICZ

Jan Stepaniec



Wiele osób uważa mgr. inż. Jana Stepańca za jedną z najciekawszych postaci w 25-letniej historii ETOB-u.

On sam utrzymuje, że informatyka jest mu nie tylko zawodem, ale i jedynym prawdziwym hobby. Nawet żonę ma informatyczkę. Traktuje ją (informatykę, nie żonę) jako pasję bardziej intelektualną niż techniczną. Chciałby oczywiście, aby wyniki własnych prac — zwłaszcza tych jeszcze nie dokonanych — zostały należycie docenione i wykorzystane, ale bodaj wyżej jest skłonny cenić osobistą, wewnętrzną satysfakcję z tego co robi.

Urodził się w 1939 roku w Żółkwi k. Lwowa. W 1945 r. został z rodziną repatriowany do Wolsztyna na Wielkopolsce. Tam liceum ogólnokształcące, ukończone z dyplomem olimpiady matematycznej. Zdecydował się na Wydział Łączności Politechniki Wrocławskiej. Po roku został namówiony do kontynuowania edukacji w bogatej tradycjami Wyższej Szkole Technicznej im. Baumana w Moskwie, na Wydziale Konstrukcji Przyrządów, gdzie potem wybrał specjalność budowy i obsługi urządzeń i maszyn cyfrowych.

Po powrocie do Polski w 1962 r. podejmuje pracę w Biurze Rozliczeń Budownictwa (obecnie ETOB). Wprawdzie w BRB nie było jeszcze komputera, tylko maszyny licząco-analityczne, ale elektroniczne przystawki kalkulatorów zawierały sporo ponad tysiąc lamp (potem tranzystorów), co pracę konserwatora tych urządzeń — zwłaszcza poprzedzoną praktyką we Francji — czyniło dostatecznie atrakcyjną dla początkującego „hardware'owca".

Śledził też J. Stepaniec to wszystko, co w jego dziedzinie działo się w Polsce u innych. Pojawiły się ZAM-y, ale na dobre zainteresował się komputerem ICT-1300, zakupionym przez ówczesny Ośrodek Doskonalenia Kadr Kierowniczych. Było to pierwsze w kraju urządzenie autentycznie zorientowane na masowe przetwarzanie danych. Ośrodek wyraził zgodę na korzystanie z maszyny przez BRB, ale trzeba było przygotować oprogramowanie użytkowe, ucząc się niejako „w biegu" umiejętności programowania. Wspólnie z J. Kochanowskim (dotąd pracuje w ETOB-ie) opracowuje pierwszą „materiałówkę" dla budownictwa, a razem z I. Linkowską — pierwszy system zatrudnienia i płac.

A opracowały je w 1966 roku trzy w zasadzie osoby w ciągu ośmiu zaledwie miesięcy, programować zaś trzeba było w języku wewnętrznym maszyny. Nikt tych ludzi nie poganiał, ani się specjalnie nie zapracowywali. Co najwyżej nie chcieli dać się wyprzedzić zespołom z innych resortów. Obecnie, przy istotnych ułatwieniach w technice programowania, bywa, że podobne opracowania — wykonywane przez parokrotnie liczniejsze zespoły — powstają w czasie parokrotnie dłuższym. Warto się nad tym zadumać i wyciągnąć wnioski.

Dostawy komputerów do BRB odwlekały się jednak, a nasz bohater coraz pewniejszym i bardziej aktywnym stawał się w programowaniu. Na ICT-1300 opracował m. in. programy sortowania szybsze od firmowych. Kiedy więc w 1967 r. firma ICL szuka w Polsce kandydatów na szkolenie, J. Stepaniec ma pewne miejsce w nielicznej grupce wybranych do rocznego stażu. W Anglii kontynuuje prace nad programami sortowania, najpierw na wielkogabarytowych kasetach magnetycznych, potem na dyskach. Oswaja się z systemami operacyjnymi GEORGE. Po latach powie jednak, że musiał bardzo długo czekać na możliwości praktycznego wykorzystania tej wiedzy, a kiedy to wreszcie nadeszło, trzeba było gruntownie „odkurzać" zasoby pamięci. Tej ludzkiej, nie komputerowej.

Rok 1970/71 był znamienny tym, że BRB otrzymało pierwsze MIŃSKI 32. Z tego okresu mgr Stepaniec pamięta zwłaszcza dwa fakty: że w mieście Mińsk chodziło się tyłem do przodu z powodu silnych wiatrów przy 30-stopniowym mrozie i że w komputerach MIŃSK trzeba było szybko „leczyć" fabryczne oprogramowanie użytkowe z licznych braków i błędów utrudniających przetwarzanie danych na szeroką skalę.

W tym mniej więcej okresie rozpoczynają się dla Jana Stepańca wieloletnie zmagania z czymś takim, co P. Lawrance sformułował jako „Prawo Petera". Mnożą się propozycje obejmowania coraz to bardziej eksponowanych funkcji kierowniczych, z typową argumentacją: — któż lepiej do tego się nadaje? Zwykle te funkcje w końcu przyjmował, choć twierdzi, że często zżymał się, iż pochłaniają mu czas, w którym można by było wiele bardziej pożytecznych i dających więcej satysfakcji rzeczy zrobić.

Zaczęło się od obowiązków koordynatora rozwoju informatyki w resorcie budownictwa. Potem był kierownikiem grupy zespołów projektowo-programowych, różnie z biegiem czasu kształtowanych i lokowanych organizacyjnie. Był też zastępcą kierownika popularnej swego czasu, ponad 200-osobowej pracowni „ETOBSYSTEM". W 1973 roku odbył trzymiesięczny staż w W. Brytanii, w dziedzinie banków danych.

Od około czterech lat pracuje nad zagadnieniami obsługi informatycznej centrali resortu (obecnie na stanowisku głównego specjalisty w warszawskim ETOB-ie). Już pierwsze zastosowania informatyki przyniosły odczuwalne efekty i zostały wysoko ocenione. Dotyczyły corocznej pracy nad planem budownictwa, a ściślej — bilansowania zadań i mocy wykonawczej tego resortu. Drugą domeną wsparcia informatycznego jest kontrola wykonywania inwestycji szczególnie ważnych w budownictwie przemysłowym.

Jan Stepaniec twierdzi, że ETOB osiągnął już w zasadzie kres swych możliwości usługodawczych dla centrali resortu przy obecnym wyposażeniu informatycznym. Istotny postęp może być dokonany tylko przez zastosowanie sprawnego systemu, zapewniającego odpowiedzi na formułowane przez Ministerstwo pytania już po kilku godzinach. Język zapytań nie musi być łatwy i prosty — końcówkę w Ministerstwie obsługiwaliby informatycy. Nie da się tego osiągnąć na ODRZE 1305, ani tym bardziej na R 32. Może więc R 60?

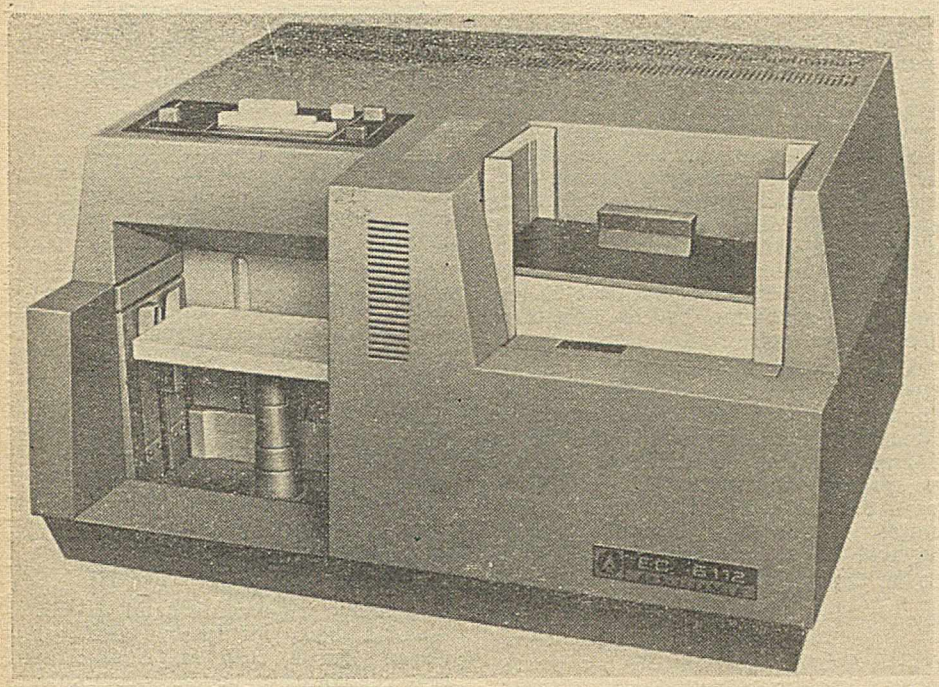
Doc. dr Marek Grochowski, Naczelny Dyrektor Centrum ETOB, wpadł na pomysł, aby kierunki rozwoju informatyki w budownictwie na lata 1981—85 zostały opracowane w oparciu o dwie, niedawno zadane, prace doktorskie. Jednym z doktorantów jest mgr inż. Jan Stepaniec. Teza wiodąca jego pracy polega na skonstruowaniu i analizie zamierzonego do wprowadzenia w budownictwie modelu zarządzania o charakterze dyrektywnym, tak, aby w jak największym stopniu wyeliminować sprzeczności interesów przedsiębiorstw, zjednoczeń i resortu. Założono iż skuteczność, a nawet realność funkcjonowania takiego modelu wymaga odpowiednio zorganizowanego wsparcia informatycznego. I właśnie koncepcja realizacji tego wsparcia jest głównym celem pracy doktorskiej.

Najprawdziwszą i najbliższą sercu pasją zawodową naszego bohatera jest jednak warsztat programistów. Zapytany, czy nie marzy mu się czasem zmiana profesji, odpowiedział bez wahania: — chciałbym umrzeć programistą, ale wtedy dopiero, gdy inni programiści będą powszechnie stosowali co najmniej dziesięciokrotnie wydajniejsze metody pracy, a ja będę widział swój znaczący udział w dostarczeniu im tych metod.

Niewątpliwą J. Stepańca zaletą jest jego otwartość na udostępnianie swej wiedzy innym. Wydaje się nawet, że on to po prostu lubi, ale bez skłonności do epatowania bliźnich. Wśród tych bliźnich są wprawdzie i przekonani o tym, że Stepaniec precyzyjnie wyważa co i komu „sprzedać", a co zachować dla siebie. Czy to jednak zarzut? Faktem jest, że od ośmiu lat jest wykładowcą i kierownikiem prac dyplomowych na Studium Podyplomowym Politechniki Warszawskiej o nazwie „Przetwarzanie danych i zarządzanie w budownictwie". Faktem jest, że w sieci ETOB już istnieje wiele trwałych śladów jego działalności. (wład)

Nowe czechosłowackie urządzenia peryferyjne

Czytnik kart dziurkowanych A 2050 (EC 6112)



Czytnik A 2050 jest małym urządzeniem peryferyjnym w wykonaniu stołowym, dostosowanym do wprowadzania danych z 80- lub 90-kolumnowych, a także 51- lub 58-kolumnowych kart dziurkowanych. Pod symbolem EC 6112 wchodzi on do wykazu urządzeń peryferyjnych Jednolitego Systemu EMC (RIAD).

Urządzenie to może odczytywać 256 kombinacji kodu KPK 12, które przekształca na znaki kodów DKOI lub KOI-8, a także przesyłać dane binarnie w zapisie sześciobitowym. Odczytywane karty mogą być wydziurkowane lub znaczone ołówkiem w formie pionowych kresek.

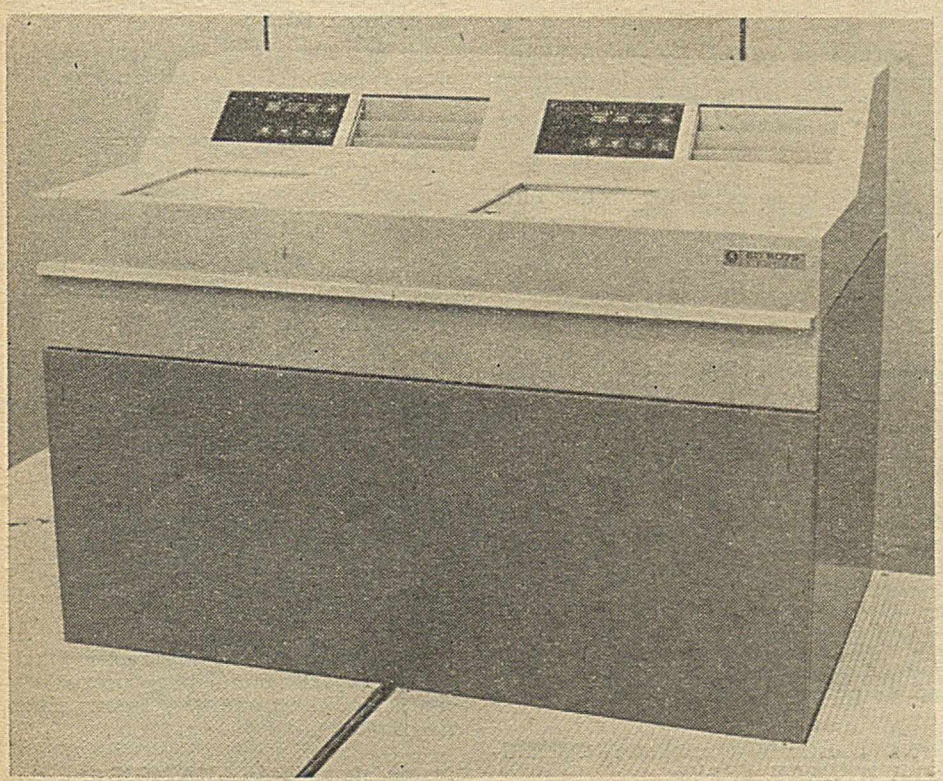
Zmiana kodu polega wyłącznie na wymianie płytki dekodera w elektronice czytnika. Pojemność magazynu podającego i odkładającego wynosi 600 kart, natomiast szybkość odczytu — 300 kart/min. Czytnik dostosowany jest do wymagań typowego interfejsu JS EMC.

W roku ubiegłym przeprowadzono pomyślnie próby współpracy czytnika z minikomputerem MERA 305 i MERA 400 w GUS oraz MERA-ZSM w Warszawie.

Producentem obu urządzeń jest czechosłowackie przedsiębiorstwo Aritma natomiast eksporterem PHZ KOVO w Pradze.

Jednostka pamięci na dyskach elastycznych EC 5075

Jednostka pamięci na dyskach elastycznych EC 5075 jest przeznaczona do bezpośredniej współpracy z komputerem. Odpowiada ona podobnej jednostce pamięci IBM 3540 model B2. Sposób oraz format zapisu danych są zgodne z projektem normy ISO oraz standardu IBM GA 21-9190. Jednostka ta jest urządzeniem peryferyjnym przeznaczonym dla drugiej generacji komputerów Jednolitego Systemu EMC (RIAD 2). Została ona skonstruowana na układach scalonych i wyposażona w pamięć typu PROM.



Jednostkę pamięci EC 5075 można przyłączyć do kanałów: standardowego multipleksorowego, selektorowego lub multipleksorowego blokowego. Jest wyposażona we własną jednostkę sterującą, zapewniającą niezależną obsługę dwóch zespołów pamięci oraz dwóch urządzeń automatycznej wymiany dysków elastycznych.

W jednym urządzeniu automatycznej wymiany można umieścić maksymalnie 20 dysków elastycznych, co stanowi równoważność 38 000 kart dziurkowanych. Jednostka sterująca umożliwia pracę urządzenia w dowolnym trybie tzn. jako urządzenie wejścia-wyjścia, dwa niezależne urządzenia wejścia albo równoległe urządzenia wyjścia.

Dalszą zaletą EC 5075 jest znaczna szybkość wprowadzania i wyprowadzania danych. W porównaniu z czytnikiem kart EC 6016 szybkość ta jest 3,6 raza większa, a w porównaniu z dziurkarką EC 7014 nawet 36 razy większa.

Należy spodziewać się, że EC 5075 okaże się bardzo potrzebnym urządzeniem peryferyjnym, zwłaszcza po rozpoczęciu produkcji programowanej stacji rejestracji danych na dyskach elastycznych PSPD 90 w zakładach MERA KFAP Kraków.

Opracował na podstawie materiałów firmowych W. Klepacz

IBM System 38

System 38 zastąpić ma wprowadzony na rynek 10 lat temu komputer biurowy System 3. Firma IBM wyprodukowała i zainstalowała w ciągu tego okresu ok. 40 tys. egzemplarzy tego modelu.

System 38 charakteryzują istotne zmiany w architekturze i systemie operacyjnym. Architektura jest jeszcze otoczona tajemnicą firmową, ale wiadomo, że zastosowano dwa typy mikrokodów, obydwa zapamiętane w pamięciach o bezpośrednim dostępie (RAM).

System 38 nie jest szczególnie interesujący w tradycyjnych porównaniach cenowych, ale jego wysoki koszt tłumaczą znacznie rozszerzone możliwości funkcjonalne, zwłaszcza zaś możliwości wirtualnej adresacji 280×10^{12} bajtów. Większość funkcji systemu operacyjnego realizowana jest w formie mikrokodu.

Podstawowe dane techniczne:

	Model 3	Model 5
Pamięć główna		
pojemność	512 KB÷1 MB	512 KB÷1,5 MB
cykl	1100 ns	600 ns
moduły	RAM á 64 K bitów	RAM á 32 K bitów
Szybka pamięć sterująca mikroinstrukcjami		
pojemność	4000 słów á 32 bity	8000 słów á 32 bity
cykl	400 ns	200 ns
moduły	RAM á 12 K bitów	RAM á 1096 bitów
Cena konfiguracji podstawowej	ok. 70 + 89 tys. dol.	ok. 100 + 136 tys. dol.

Koszt programu konwersji i rekompilacji z systemu 3 wynosi jednorazowo 1300 dol.

Dla orientacji o poziomie kosztów podano, że mała konfiguracja modelu 3 z pamięcią 512 KB, pamięcią dyskową 64, 5 MB, drukarką i dwoma ekranowymi urządzeniami końcowymi kosztować będzie ok. 115 tys. dol., a w przypadku dzierżawy 3 444 dol. miesięcznie. Model 5 z pełną pamięcią główną (1,5 MB), pamięcią dyskową o pojemności 387,1 MB, dwoma drukarkami, 32 terminalami ekranowymi, 8 drukarkami pomocniczymi oraz 60 terminalami zdalnymi będzie kosztować ok. 760 tys. dol.

Cena modułów pamięci jest zaskakująco niska: 20 000 dol. za 1 MB w modelu 3 oraz 28 000 dol. w szybszym modelu 5. Dla porównania 1 MB w tej samej technologii 64 K RAM kosztuje w systemie 8100 18 000 dol., a 1 MB w serii 370 110 tys. dol. Cena 1 MB pamięci w systemach Hewlett Packard wynosi 32 000 dol. (wszystkie ceny dotyczą rynku USA).

Podobnie jak w systemie 8100, system operacyjny oraz całe oprogramowanie firmowe są odpłatne. Przykładowo koszt systemu operacyjnego CPF wynosi 400 dol. miesięcznie, języka programowania RPG III — 60 dol. miesięcznie, programów usługowych zarządzania bazą danych — 30 dol. miesięcznie.

System 38 będzie na rynku konkurował przede wszystkim z odpowiednimi modelami firm DEC, Hewlett Packard i Prime Computer Co. **(T. J.)**

Na podstawie czasopisma ELECTRONICS, nr 24/78

HONEYWELL w Jugosławii

Kierownictwo firmy HONEYWELL zatwierdziło utworzenie nowego przedsiębiorstwa w kooperacji z Jugosławią pod nazwą EI-HONEYWELL INFORMACIONI SISTEMI, w którym HONEYWELL będzie miał 30-procentowy udział.

Proponowane przedsięwzięcie znajduje się obecnie w fazie zatwierdzania przez rządy USA i Jugosławii oraz przez kierownictwa przedsiębiorstw jugosłowiańskich, które porozumienie to będą realizowały. Przedsiębiorstwami tymi są: ELEKTRONSKA INDUSTRIJA (EI), koncern produkcji wyrobów elektrycznych i elektronicznych, który będzie miał 63% udziału, oraz PROGRES, dotychczasowe przedsiębiorstwo sprzedaży wyrobów firmy HONEYWELL na Jugosławię, z udziałem 7-procentowym.

Biura nowego przedsiębiorstwa będą zlokalizowane w Belgradzie i Niszu, natomiast zakład produkcyjny zostanie uruchomiony w Niszu. Zatrudnienie wyniesie ok. 200 osób, przy czym reprezentantami HONEYWELLA będą mogły być osoby należące do 4 różnych narodowości. Przedsiębiorstwo będzie produkowało minikomputery serii LEVEL 6 (modele 33, 37, 43, 47, 53 i 57), jak również niektóre modele drukarek znakowych i monitorów ekranowych tej firmy.

Porozumienie to, uzależnione nadal od ostatecznych formalnych zatwierdzeń, jest kontynuacją długoletniej współpracy i poprzednich porozumień pomiędzy firmami EI, PROGRES i HONEYWELL związanych z rozwojem zastosowań komputerów w instytucjach rządowych i przemyśle Jugosławii. **(W.K.)**

Odpowiedź SIEMENSA na System IBM 4331

Odpowiedzią SIEMENSA na IBM-owski System 38 i 4331 jest nowy model serii 7000, mieszczący się poniżej parametrów ogłoszonych w ubiegłym roku modeli 7.708 i 7.718. Model ten nazwano 7.706, a jego minimalna konfiguracja charakteryzuje się pamięcią operacyjną o pojemności 384 KB i obejmuje dwie jednostki pamięci dyskowej po 60 MB, jednostkę pamięci na dysku elastycznym oraz drukarkę o szybkości 300 wierszy/min. Koszt takiego zestawu, wyposażonego w system operacyjny BS 200 wynosi ok. 265 000 DM (ok. 1 400 000 dol.).

Zestaw powyższy może być na miejscu u użytkownika rozbudowany do typowych konfiguracji modeli 7.708 i 7.718. Rozbudowa modelu może również następować na odcinku oprogramowania w drodze rozszerzenia do systemu operacyjnego BS 2000 o elementy umożliwiające pracę w trybie konwersacyjnym.

Dostawy nowego modelu przewidziano na początek 1980 r. **(T. J.)**

Na podstawie czasopisma COMPUTER WEEKLY, nr 6/79

CYBER dla Chin

Chiny zakupiły od firmy Control Data trzy systemy komputerowe z serii CYBER za kwotę ok. 69 mln dolarów. Jest to największy jednorazowy kontrakt zawarty z krajem nie kapitalistycznym i dotyczy m.in. komputera CYBER 175 o mocy obliczeniowej ośmiokrotnie większej niż jakikolwiek komputer sprzedany dotąd do kraju socjalistycznego.

Zakupione przez Chiny systemy mają być wykorzystane przy poszukiwaniach złóż ropy naftowej i gazu ziemnego. **(I.S.)**

Na podstawie DATAMATION nr 2/1979

Sytuacja po 10 latach

Mało firm komputerowych ma lepszy powód do uroczystego obchodzenia 10-lecia swego istnienia niż brytyjski ICL. W przemyśle, w którym obserwuje się likwidacje nawet największych firm o zasięgu światowym, ICL, ktory osiągnął ostatnio roczną kwotę sprzedaży rzędu 1 mld dol., może poszczycić się większym sukcesem niż samo tylko przetrwanie 10 lat, jakie upłynęły od czasu utworzenia firmy.

Nie dawniej jak dwa lub trzy lata temu tylko niewielu ludzi dawało ICL-owi szanse na uzyskanie dzisiejszej, bardzo wysokiej pozycji rynkowej. Podstawą takiego pesymizmu były kłopoty związane z odziedziczeniem klientów firmy English Electric Computers Ltd., która w 1968 r. połączyła się z ICL. Decyzja podtrzymywania zarówno ICL-owskiej serii 1900, jak i Systemu 4 EEC niemal nie stała się katastrofą dla nowo utworzonej firmy.

„Uważam, że połączenie to było błędem, a Systemu 4 trzeba było się z czasem pozbyć" — powiedział szef amerykańskiej filii ICL Tom Hudson. Decyzja ta nie tylko wywołała problemy podtrzymywania obu serii, ale dodała kłopoty związane z rozwojem nowej linii stanowiącej pomost między poprzednimi seriami. Upłynęło 6 lat, wydano 80 mln. dol. z kasy państwowej i szeroko wprowadzono pionierskie metody mikroprogramowania — zanim w 1974 r., uruchomiono pierwsze modele nowej serii — komputery 2980 i 2970. Ale nawet wtedy natknięto się na trudności z systemem operacyjnym, a złą opinię na temat tego systemu dopiero ostatnio udało się zatrzeć.

Obecny dyrektor ICL, Christopher Wilson, niezachwianie wierzy, że decyzja opracowania nowej serii była słuszna i ustawi firmę w dobrej pozycji w latach 80-tych. Twierdzi on, że większość nowych serii przechodzi kilkuletni okres ząbkowania (czego najlepszym przykładem jest historia serii IBM 360) i że „gdybyśmy zatrzymali się na architekturze serii 1900, stalibyśmy dotąd na bocznicy bez pary".

Firma zaprojektowała nowy system od podstaw, włączając tak nowoczesne rozwiązania, jak koncepcję maszyny wirtualnej, wielopoziomowe adresowanie pamięci wirtualnej do pojemności 4 mld bajtów oraz przetwarzanie wg zasady „push down stack". Ostatnio ogłoszone podjęcie produkcji modeli pośrednich 2940 i 2950 stało się znacznym sukcesem firmy.

Jednym z twórców obecnej pozycji ICL jest Geoffrey Cross. Wraz z Crossem do ICL przeszła grupa kierownicza, ktora poprzednio pracowała z nim w firmie Univac. Cross pełniący do 1977 r. funkcję dyrektora ICL, wprowadził nowe zasady dyscypliny finansowej i odpowiedni układ priorytetów oraz udaremnił rozgrywki frakcyjne między rywalizującymi grupami z ICL i English Electric. Cross jest również autorem bardzo ważnej dla ICL decyzji, jaką był zakup w 1976 r. firm Singer Business Machines i Singer's Cogar Corp.

Transakcja ta rozszerzyła zasięg ICL-u w trzech podstawowych obszarach. Po pierwsze, wprowadziła firmę na szybko rozwijający się rynek małych systemów biurowych. Po drugie — stworzyła ICL-owi bazę produkcyjną w USA — umożliwiając kontakt z technologią amerykańską. I po trzecie — zapewniła ICL-owi rzeczywistą obecność na rynku europejskim. Nastąpiła również pewna subtelna zmiana psychologiczna — „Firma zaczęła myśleć kategoriami międzynarodowymi" — jak to skomentował jeden z członków kierownictwa.

W jakich obszarach zamierza się ICL uplasować w pierwszej połowie lat 80-tych, gdy wpływy ze sprzedaży osiągną rząd 2 mld dol.? Wydaje się, że częściowo w Starym, a częściowo w Nowym Świecie. Już obecnie z Europy kontynentalnej płyną zyski rzędu 260 mln dol. ze sprzedaży małych systemów biurowych. Potrzebny będzie teraz wzrost i osiągnięcie minimalnej bazy klientów modeli pośrednich serii 2900.

W styczniu 1977 r., po zaprzestaniu działalności niektórych biur ICL w USA, powołano nową firmę ICL Inc., której zadaniem jest umocnienie obecności ICL w Stanach Zjednoczonych.

(T. J.)

Na podstawie czasopisma ELECTRONICS, nr 24/78

CII Honeywell—Bull fundatorem nagród

Kierownictwo firmy CII Honeywell-Bull ustanowiło nagrodę w wysokości 50 000 franków francuskich za wybitne osiągnięcia w dziedzinie informatyki. Nagroda ta będzie wypłacana co 2 lata osobie lub zespołowi za innowacje na poziomie europejskim, zarówno w działalności praktycznej, jak i w pracach teoretycznych. Pierwsza tego rodzaju nagroda została przyznana we wrześniu br. **(W.K.)**

Na podstawie czasopisma ONLINE nr 3/79

MERA·ELWRO

Dwa medale na SOFTARGU'79

Na katowickiej imprezie SOFTARG '79 zostały wyróżnione przyznaniem medali 2 firmowe pakiety oprogramowania MERA-ELWRO: „Biblioteka modułów matematycznych" oraz „Konwersacyjny system wyszukiwania informacji naukowo-technicznej WINT".

Wyróżniony złotym medalem pakiet „Biblioteka modułów matematycznych" zawiera około 450 podprogramów i funkcji (w znaczeniu języka FORTRAN) umożliwiających łatwe wykonywanie standardowych obliczeń numerycznych z następujących dziedzin matematyki:

— teoria liczb, kombinatoryka
— działania na wielomianach, ciągach, szeregach nieskończonych, ułamkach łańcuchowych
— szacowanie i obliczanie zer wielomianów
— rozwiązywanie równań nieliniowych
— rozwiązywanie układów równań nieliniowych
— obliczanie ekstremów funkcji jednej i wielu zmiennych
— interpolacja
— aproksymacja średniokwadratowa i innych rodzajów oprócz jednostajnej
— algebra liniowa
— całkowanie i różniczkowanie numeryczne
— rozwiązywanie równań różniczkowych zwyczajnych i ich układów
— rozwiązywanie równań całkowych i całkowo-różniczkowych
— obliczanie wartości funkcji elementarnych i specjalnych
— obliczania probabilistyczne i statystyczne
— porządkowanie, sortowanie
— badania operacyjne.

Pakiet ten dostarczany jest w dwóch wersjach, dostosowanych do systemów DOS i OS.

Wyróżniony srebrnym medalem „Konwersacyjny system wyszukiwania informacji naukowo-technicznej WINT" umożliwia:

— przechowywanie i aktualizację informacji o dowolnej tematyce
— redagowanie biuletynów informacyjnych wraz z indeksami
— konwersacyjne wyszukiwanie informacji z wykorzystaniem monitorów ekranowych.

WINT może być eksploatowany pod kontrolą systemu operacyjnego OS. Dokładniejsze informacje na temat wyżej wymienionych pakietów można uzyskać w Dziale Serwisu Oprogramowania BOT Elwro-Serwis, Wrocław, ul. Ostrowskiego 32 tel. 44-35-23.

Przemysław NOWAK

System edycji programów w języku SAWIK

W Instytucie Łączności prowadzone są m.in. prace nad rozwojem oprogramowania minikomputera MERA 305, umożliwiającego współpracę z apraturą automatycznych pomiarów łączy międzymiastowych. W pracach tych zachodziła konieczność pisania i uruchamiania wielu rozbudowanych programów w języku SAWIK. Celowe stało się więc opracowanie systemu ułatwiającego i przyspieszającego wykonanie postawionych zadań, z pełnym wykorzystaniem możliwości stwarzanych przez konfigurację zainstalowanego w Instytucie sprzętu.

System taki, który nazwano ELBE, umożliwia m.in.:
- pisanie programów za pomocą monitora ekranowego
- wyświetlanie na monitorze dowolnego fragmentu napisanego programu
- poprawianie dowolnego fragmentu programu
- przechowywanie programów źródłowych w zbiorze bibliotecznym zarejestrowanym w pamięci dyskowej.

Do systemu włączono translator języka SAWIK, w którym dokonano niewielkich zmian, np. umożliwiających czytanie programu z pamięci dyskowej. Główny nacisk położono na prostotę eksploatacji. Zrezygnowano z tworzenia „języka" dyrektyw, zastępując to stawianiem alternatywnych pytań wyświetlanych na ekranie monitora w momentach wymagających podjęcia decyzji. Dodatkowym ułatwieniem jest automatyczne ustawianie się znacznika (kursora) na żądanej lub spodziewanej pozycji. Do minimum ograniczono korzystanie z taśmy papierowej, którą wykorzystuje się jedynie na końcu translacji tzn. w momencie gdy wyprowadzana jest postać binarna programu.

Mówiąc ogólnie, system pozwala na manipulowanie dowolnymi tekstami, które traktowane są jako programy dopiero w trakcie translacji. Dlatego też dalej używać będziemy szerszego określenia „tekst", rozumiejąc przez to także program zakodowany w języku SAWIK.

WŁASNOŚCI SYSTEMU

System wymaga następującej konfiguracji sprzętu:
- jednostka centralna MERA 305
- jednostka pamięci dyskowej
- alfanumeryczny monitor ekranowy (VIDEOTON)
- drukarka znakowo-mozaikowa DZM 180
- dziurkarka taśmy papierowej
- czytnik taśmy papierowej.

Dla przygotowywanego tekstu system rezerwuje około 46 K bajtów w pamięci dyskowej, co pozwala na zapisanie około 46 000 znaków. Można oszacować, że umożliwia to zapis około 7000 instrukcji języka SAWIK. Oprócz obszaru roboczego, w pamięci dyskowej wyróżniono obszar pośredni o tej samej wielkości. Pełni on funkcje pomocnicze, służąc do chwilowego przechowywania tekstów lub ich fragmentów. Na potrzeby biblioteki rezerwowanych jest na dysku 120 ścieżek, co pozwala przechowywać do 15 programów jednocześnie. Pozostała część obszaru dyskowego pozostaje do dyspozycji programisty.

Pisane teksty nie mogą zawierać więcej niż 40 znaków w wierszu. Ograniczenie to wynika z szerokości ekranu monitora.

OPERACJE SYSTEMU

Kolejność opisu operacji systemu odpowiada kolejności czynności wykonywanych na ogół podczas przygotowywania programu.

Wybór rodzaju pracy

Po zainicjowaniu działania systemu, na ekranie monitora wyświetlany jest tekst oferujący różne rodzaje pracy (rys. 1). Wybór polega na ustawieniu znacznika przed gwiazdką poprzedzającą wybrany rodzaj pracy. Powoduje to zniknięcie wszystkich napisów i rozpoczęcie realizacji odpowiedniego trybu pracy.

WYBIERZ RODZAJ PRACY

* PRZEGLĄDANIE	DYSK→MON
* POPRAWKI	DYSK→MON
* LISTOWANIE	DYSK→DZM
* TRANSLACJA	DYSK→DZM
* SKŁADOWANIE	DYSK→BIBL
* POBIERANIE	BIBL→DYSK
* DZIURKOWANIE	DYSK→TP
* WCZYTYWANIE	TP→DYSK
* PISANIE	MON→DYSK
* CZYTANIE BINARNE	TP→PAO

Rys. 1. Tekst na ekranie monitora pozwalający na wybór żądanego trybu pracy (wybrano translację)

Pisanie

Podczas realizacji tego trybu pracy, na ekranie można pisać dowolne teksty (nie zawierające znaku „#"), korzystając ze wszystkich możliwości jakie stwarza monitor ekranowy. Piszący nie jest obarczony koniecznością stawiania specjalnych znaków nowej linii, ponieważ tekst zapamiętywany jest w postaci „optycznej", tzn. odpowiadającej dokładnie temu, co jest wyświetlone na ekranie.

Zapisanie zawartości ekranu w obszarze pośrednim pamięci dyskowej następuje po wysłaniu przerwania (specjalny klawisz monitora). Zapamiętany zostaje wówczas fragment tekstu zawarty pomiędzy znacznikiem, a znakiem końca transmisji. Należy dodać, że spacje, za którymi nie następuje żaden inny znak w wierszu, nie są zapamiętywane. Po zapamiętaniu tekstu można ponownie wpisywać dowolne ciągi znaków, zapamiętywać je, kasować zawartość ekranu itp.

Wykrycie przez system znaku „#" w tekście oznacza zakończenie czynności pisania. Cały napisany tekst jest wówczas przepisywany z obszaru pośredniego do obszaru roboczego pamięci dyskowej i sterowanie systemu przechodzi do początku (wybór rodzaju pracy).

Przeglądanie

Po wybraniu tego rodzaju pracy, na ekranie wyświetlonych zostaje 20 ponumerowanych wierszy tekstu przechowywanego w obszarze roboczym pamięci dyskowej. Wysłanie przerwania powoduje skasowanie zawartości ekranu i wyświetlenie kolejnych 20 wierszy. Po wyświetleniu całego tekstu z obszaru roboczego sterowanie przekazywane jest do początku (wybór rodzaju pracy).

Listowanie

W tym trybie pracy, na drukarce wypisywany jest tekst znajdujący się w obszarze roboczym pamięci dyskowej, z numerowaniem kolejnych wierszy. Po zakończeniu listowania, sterowanie przechodzi do początku (wybór rodzaju pracy).

Poprawki

Wybranie tego rodzaju pracy powoduje, że na monitorze ukazuje się żądanie podania numeru wiersza, który ma być wyświetlony. Po określeniu numeru (i wysłaniu przerwania), z obszaru roboczego do obszaru pośredniego pamięci dyskowej zostaje przepisana część tekstu znajdująca się przed wybranym wierszem. Wiersz jest wyświetlany na ekranie. W celu wyświetlenia następnego i dalszych wierszy należy w lewym dolnym rogu ekranu umieścić specjalny znak i przez ustawienie znacznika (kursora) wskazać miejsce na ekranie, w którym wiersz ma być wyświetlony. Wysłanie przerwania powoduje wyświetlenie wiersza i ustawienie się znacznika na początku następnej linii ekranu.

Odpowiednio ustawiając znacznik można przestawiać poszczególne wiersze. Należy pamiętać, że wyświetlane wiersze znajdują się tylko na ekranie, tzn. „giną" z pamięci dyskowej. Po pojawieniu się żądanego fragmentu, na ekranie można pisać dowolne ciągi znaków, zmieniać lub kasować wyświetlone wiersze itp.

Dla zapamiętania fragmentu tekstu aktualnie wyświetlonego na ekranie, na jego początku należy umieścić znacznik, a na końcu znak końca transmisji i wysłać przerwanie. Powoduje to dopisanie znaków z wybranej części ekranu do istniejącej zawartości obszaru roboczego pamięci dyskowej. W przypadku konieczności wysłania ciągu znaków przekraczającego jednorazową pojemność ekranu, znaku końca transmisji nie umieszcza się. Ograniczeniem jest wtedy pojemność ekranu, a po jego zapisaniu system czeka na dalsze fragmenty tekstu. Gdy po wyświetleniu zachodzi potrzeba skasowania niektórych wierszy, do pamięci dyskowej przesyła się tekst pusty.

Po zapisaniu poprawki system ponownie żąda podania numeru wiersza. Podanie zerowego numeru oznacza, że dokonano już wszystkich poprawek. Przepisywany jest wtedy pozostały fragment tekstu z obszaru roboczego do obszaru pośredniego pamięci dyskowej, a następnie cały zmodyfikowany tekst z powrotem do obszaru roboczego. Sterowanie systemu przechodzi wówczas do początku (wybór rodzaju pracy).

Translacja

W trybie translacji tekst traktowany jest jako program zakodowany w języku SAWIK.

Wykorzystany został standardowy translator języka SAWIK, korzystający z drukarki znakowo-mozaikowej oraz dziurkarki taśmy papierowej. Tłumaczony program pobierany jest w obu przebiegach translacji nie z taśmy papierowej, lecz z obszaru roboczego pamięci dyskowej i równocześnie wyświetlany jest na ekranie monitora. Ułatwia to szybką analizę błędów zauważonych w czasie translacji. Produkt translacji (postać binarna programu) zapisywany jest w pamięci dyskowej i dziurkowany na taśmie papierowej dopiero po zakończeniu pracy translatora. Następnie sterowanie systemu przechodzi do początku (wybór rodzaju pracy).

Składowanie

Składowanie umożliwia zapisywanie w specjalnym obszarze bibliotecznym pamięci dyskowej zawartości często używanego obszaru roboczego. Wyróżniono 15 obszarów składowania tworzących bibliotekę. Obszary zostały nazwane literami A, B,...,O. Po wpisaniu odpowiedniej litery (żąda tego napis na ekranie) i wysłaniu przerwania, w dowolnej części ekranu zostaje wyświetlonych 5 początkowych wierszy tekstu mieszczącego się w wybranym obszarze bibliotecznym. W przypadku, gdy zniszczenie istniejącej zawartości zostanie uznane za dopuszczalne, użytkownik wysyła przerwanie i tekst zostaje przesłany. W innym przypadku należy wybrać inny obszar składowania.

```
PODAJ OBSZAR SKŁADOWANIA
A—B—C—D—E—F—G—H—I—J
OBSZAR: C
```

```
(PROGRAM—TRANSLATOR JEZYKA BADANIO-
WEGO. KOWALSKI JAN TEL — 16
NIE KASOWAC DO 12/12/79)
Q = 320;
AAA) US.; PI /5; ZA.; PZ. 1;
```

Rys. 2. Wyświetlanie początkowego fragmentu tekstu po wybraniu obszaru składowania (wybrano obszar C)

Jak widać zrezygnowano z zabezpieczeń programowych, uniemożliwiających niszczenie zawartości obszarów bibliotecznych, wychodząc z założenia, że byłyby one skomplikowane w obsłudze i mało skuteczne. Istnieje jedynie możliwość podawania w kilku (maksymalnie 5) pierwszych wierszach przechowywanego tekstu symbolu użytkownika, okresu ważności itp. W przypadku programu, informacje te powinny być umieszczone jako komentarze. Zostanie to wyświetlone po wybraniu danego obszaru (rys. 2). Po zakończeniu składowania, sterowanie systemu przechodzi do początku (wybór rodzaju pracy).

Pobieranie

Pobieranie pozwala na sprowadzenie tekstu z jednego z obszarów bibliotecznych do obszaru roboczego pamięci dyskowej. Po wybraniu tego trybu pracy zostaje wyświetlony napis żądający określenia obszaru biblioteki. Po wpisaniu symbolu (A, B,...,O.) i wysłaniu przerwania, tekst zostaje przepisany do obszaru roboczego (nadal tekst jest dostępny również w bibliotece). Następnie sterowanie systemu przechodzi do początku (wybór rodzaju pracy).

Dziurkowanie

Ten rodzaj pracy umożliwia dziurkowanie na taśmie papierowej znaków tekstu zapisanego w obszarze roboczym pamięci dyskowej. Znaki dziurkowane są w kodzie ISO. Po wydziurkowaniu całego tekstu sterowanie przechodzi do początku (wybór rodzaju pracy).

Wczytywanie taśmy dziurkowanej w kodzie ISO do pamięci dyskowej

Informacje zawarte na taśmie dziurkowanej w tym trybie pracy systemu zostają przepisane do obszaru roboczego pamięci dyskowej. Koniec taśmy traktowany jest jako koniec wczytywanego tekstu. Po zakończeniu wczytywania sterowanie systemu przekazywane jest do początku (wybór rodzaju pracy).

Wczytywanie taśmy binarnej do pamięci operacyjnej

System wczytuje informacje z taśmy binarnej wydziurkowanej przez translator języka SAWIK (translacja), sprawdzając podczas czytania sumy kontrolne. Po wykryciu symbolu końca danych minikomputer znajduje się w stanie „STOP". Należy przypomnieć, że wczytanie programu użytkowego do pamięci operacyjnej może zniszczyć system ELBE. Możliwe jest też zniszczenie programu czytania binarnego umieszczonego na końcu obszaru pamięci. Aby jednak nie ograniczać możliwości programowania, zrezygnowano z zabezpieczeń lub automatycznego przeadresowywania programu, dając programiście prawo decydowania.

* * *

Omawiany system wykorzystywany jest od września 1978 r. w Zakładzie Miernictwa i Automatyzacji Badań Instytutu Łączności. Stosowanie systemu pozwala na znaczną oszczędność czasu pracy potrzebnego programiście na opracowanie programu.

System może być dalej rozbudowany np. poprzez możliwość współpracy z translatorami innych języków, wykorzystanie pamięci na taśmie magnetycznej itp. Planuje się też stworzenie dyskowego katalogu programów binarnych.

Doświadczenia wskazują, że posługiwanie się systemem ELBE jest rzeczywiście proste i nie wymaga specjalnego przygotowania. Na ogół programiście wystarcza kilkuminutowe wprowadzenie, połączone z praktyczną ilustracją możliwości funkcjonalnych systemu.

Lech BRENNEK, Bogdan LEBIEDZIUK
Instytut Łączności
Warszawa

Efektywność systemów informatycznych zarządzania*)

Książka składa się z wprowadzenia oraz pięciu rozdziałów problemowych, spisu literatury wykorzystanej i uzupełniającej.

W rozdziale I, pt. „Kryteria oceny i kierunki badań efektywności systemów informatycznych" dokonano analizy pojęcia efektywności systemów informatycznych, przedstawiono różnorodność i ewolucję poglądów na ten temat, a także scharakteryzowano podstawowe kierunki badań dotyczących tego problemu.

W rozdziale II, który można uznać za podstawowy, pt. „Metody oceny efektywności systemów informatycznych", dokonano szerokiego przeglądu stosowanych w praktyce metod tego rodzaju oceny. Wyróżniono tu dwie podstawowe grupy tych metod: metod opartych na ocenie jakości systemu zarządzania oraz metod badających efektywność nakładów ponoszonych na przygotowanie i wdrożenie systemu informatycznego. W grupie metod opartych na ocenie jakości systemu zarządzania skoncentrowano się na dwóch metodach: metodzie analitycznej G. J. Pierce'a, w której miarą oceny efektywności systemu informatycznego jest przyrost zyskowności przedsiębiorstwa oraz metodach symulacyjnych, stosowanych przy ocenie systemów bardziej złożonych. Metody te wykorzystywane są nie tylko do oceny efektywności istniejących już rozwiązań systemowych, ale również do udoskonalania tych rozwiązań w kierunku uzyskania większej efektywności działania systemu zarządzania.

W ramach drugiej grupy metod oceny efektywności systemów informatycznych (badanie efektywności nakładów na przygotowanie i wdrożenie systemu informatycznego) wyróżnić można metody liczbowe i opisowe. Spośród metod liczbowych przedstawiono metody służące do analizy nakładów na systemy informatyczne i oceny efektów powstałych w wyniku zastosowania tych systemów, a także metody globalnej oceny efektywności systemów informatycznych (zarówno nakładów, jak i efektów).

O ile metody liczbowe stosowane są głównie do analiz wykonywanych a posteriori w systemach dobrze rozpoznanych, to do analiz wczesnych stadiów rozwoju systemu, a także tam, gdzie brak jest bardziej szczegółowego rozpoznania wielkości i struktury faktycznych nakładów i efektów, stosuje się metody opisowe (ankietowanie użytkowników, porównanie rozwiązania rzeczywistego z wzorcowym).

Rozdział III książki jest poświęcony metodom oceny wartości informacji ekonomicznej. Wartość informacji, a nie jej ilość decyduje bowiem o efektywności systemu informatycznego. Poglądy na wartość informacji i metody jej mierzenia przechodzą jednak ciągłą ewolucję i dlatego zaprezentowano cztery różne podejścia do zagadnienia pomiaru wartości informacji: mierzenie aktualnej wartości informacji poprzez symulowanie jej wpływu na efektywność decyzji, mierzenie oczekiwanej wartości informacji poprzez symulację jej wykorzystania w probabilistycznych modelach decyzyjnych, opisowe wyrażanie wartości informacji za pomocą tzw. cech użytkowności informacji, oraz mierzenie wartości informacji w aspekcie sprawności, wyrażonej w jej ilości i szybkości.

W rozdziale IV dokonano analizy efektywności wybranych systemów informatycznych zarządzania. Efektywność systemu zależy przede wszystkim od jego zakresu, przedmiotu i funkcji, a także umiejętności wykorzystania możliwości, jakie system ten stwarza w procesach działalności gospodarczej i zarządzania. Kolejnym czynnikiem decydującym o efektywności systemów informatycznych jest kompleksowość rozwiązań systemowych oraz konwergencja nowoczesnych technik zastosowanych w tych rozwiązaniach. Wreszcie ostatnim spośród podstawowych stymulatorów efektywności systemu jest technika i technologia na której oparto system. W rozdziale tym przedstawiono również syntetyczne wyniki badań efektywności zastosowań informatyki w Polsce przeprowadzonych w 1977 roku przez Komitet Informatyki.

W rozdziale ostatnim (piątym) rozważa się podstawowe warunki efektywności systemów informatycznych. Dokonano tu analizy istoty oraz wpływu następujących czynników: czynnika ludzkiego (przekonanie do systemu i zintegrowanie użytkownika z systemem), planowania potrzeb informatycznych użytkownika strategii wdrażania (w szczególności typizacji rozwiązań projektowych), organizacji i technologii projektowania i eksploatacji systemu.

Temat podjęty przez autorów opracowania jest od wielu lat aktualny nie tylko w Polsce, ale i na całym świecie, zarówno w aspekcie efektywności zastosowań informatyki, lub inaczej ekonomiki zastosowań informatyki, jak i w aspekcie efektywności eksploatacji systemów informatycznych. Na ten temat istnieje już bardzo obszerna literatura przedmiotu, w większości jednak o charakterze przyczynkowym, którą w znakomitej większości przytoczono w wykazie i przypisach zawartych w opracowaniu.

Sama książka jest więc jakby przewodnikiem po istniejącej literaturze na ten temat i ma charakter bardziej porządkujący i komentujący w stosunku do już istniejących rozwiązań niż nowatorski i odkrywczy. Wydaje się to jednak zaletą tej książki, gdyż na temat efektywności zastosowań informatyki napisano, moim zdaniem, i tak zbyt wiele w stosunku do korzyści metodycznych jakie na ten temat mogą czerpać na co dzień z tej literatury (lub z przepisów normatywnych powstałych na jej bazie) informatycy i użytkownicy systemów informatycznych.

Ostatnio daje się odczuć pewien regres zainteresowań liczeniem efektywności systemów informatycznych a także praktycznych zastosowań metod analizy ekonomicznej w odniesieniu do opracowywanych i wdrażanych systemów informatycznych. Jedną z przyczyn jest zapewne i to że większość obecnych wdrożeń to systemy już znane i wcześniej wypróbowane. Sądzę jednak, że główną przyczyną tego zjawiska jest raczej niedoskonałość dostępnego do dziś aparatu metodycznego.

Z tego też względu omawiana pozycja stanowi pożyteczne opracowanie zarówno dla praktyków jak i teoretyków oraz studentów informatyki. Dla praktyków opracowanie to byłoby bardziej użyteczne, gdyby autorzy dokonali wyboru najbardziej dojrzałych i perspektywicznych, a zarazem przydatnych w naszej praktyce poglądów i metod i opisali je szerzej, popierając nawet przykładami zastosowań, a być może i danymi charakteryzującymi ich przydatność. Tymczasem dokonano przeglądu obszernej literatury, ale w wielu przypadkach myśl jest urywana w najciekawszym miejscu lub z konieczności prowadzone rozważania mają charakter ogólniejszy niż byłoby to pożądane przez czytelnika. Ponadto, niektóre myśli zawarte w opracowaniu zawierają pewne niedomówienia lub nieścisłości, które mogą powodować wątpliwości u czytelnika. Przykładowo na str. 70 opracowania stwierdza się, że przykład podany na str. 73 „ilustruje metodę szacowania efektów z tytułu obniżenia stanu zapasów materiałowych...", natomiast na str. 73 owych efektów, wcale się nie szacuje, lecz przytoczony przykład kończy się określeniem wielkości obniżki średniego zapasu materiałów, co w istocie stanowi źródło efektów, ale nie jest efektem ostatecznym sensu stricto.

Wątpliwości również budzi zasadność użycia pojęcia „informacja doskonała" (str. 120), zwłaszcza w podanej na tejże stronie interpretacji. Napisano przy tym, że rezultatem strategii opartej na informacji doskonałej byłyby zawsze optymalne wyniki ekonomiczne. Zapomina się w tym miejscu zarówno o tym, że pełna informacja bez optymalizacji nie musi zapewnić optymalnej decyzji jak i o tym, że nawet optymalna decyzja nie musi zapewnić optymalnych wyników przy probalistycznym modelu funkcjonowania przedsiębiorstwa. Zresztą i sami Autorzy rozmywają pojęcie „informacji doskonałej", stawiając znak równości pomiędzy informacją doskonałą, a informacją użyteczną (str. 131, ustęp 2).

Niepełna jest również errata do błędów powstałych w druku. Oto najważniejsze spośród zauważonych dodatkowo błędów:

*) Janusz Iłczuk i Maria Jerczyńska: Efektywność systemów informatycznych zarządzania. PWE, seria „Informatyka w praktyce" Nakł. 3000 egz., stron, 207, cena 38 zł. Warszawa 1979.

1) str. 44, nazwa źródła pod rys. 2 — powinno być: A user's...
2) str. 61, zmienna S powinna być nazwana średnią stopą amortyzacji, a nie stawką, gdyż nie można dodawać stopy do stawki,
3) str. 86, we wzorze na N powinno być: — E, a nie + E
4) str. 119, wzór w odnośniku 17 z lewej strony równości powinien mieć postać: $P/A_i(B)$, a nie $P/A_1(B)$.
5) str. 190, tabl. 22 w wierszu 4, kol. 3 powinno być: 0—22, a nie 0,22.

Pomimo wyrażonych życzeń w zakresie pożądanych uzupełnień treści, niekiedy dyskusyjnych stwierdzeń oraz drobnych usterek technicznych, omówiona pozycja stanowi cenne uzupełnienie krajowej myśli informatycznej i z przyjemnością pragnę polecić ją szerokiemu gronu odbiorców: informatyków i użytkowników systemów, zaś Autorom publikacji pragnę wyrazić podziękowanie za jej przygotowanie.

Zygmunt BIEŃKO

Włajemniczenie teorio(hiper)grafowe

Motto:
Z matematycznego punktu widzenia teoria grafów nie wnosi nic nowego.

Żeby nie było nieporozumień, zdradzimy od razu, że powyższy cytat pochodzi od samego autora ostatnio wydanej swego rodzaju minimonografii teorii grafów i sieci *). Autor zresztą, jako wieloletni wykładowca tego przedmiotu w WAT, daje wyraz swemu głębokiemu przeświadczeniu o pożyteczności grafów, ba wręcz ich doniosłości, jako świetnego narzędzia pojęciowego do modelowania procesów. Co nie zmienia faktu, że z matematycznego...

Bohdan Korzan opracowywał swą książkę pod kątem określonych odbiorców: ekonomistów, genetyków, lingwistów, elektroników, informatyków, automatyków, a nawet inżynierów innych specjalności, którzy z teorią grafów być może wiążą ogromne nadzieje. Już we wstępie Autor usiłuje rozwiać złudzenia, że grafy są panaceum na bolączki problemowe. W zakresie dotychczasowych zastosowań należałoby bowiem mówić właściwie bardziej o zastosowaniu niektórych pojęć teorii grafów, aniżeli o zastosowaniach samej teorii.

To nie teoria grafów zrobiła karierę, bo jest to teoria skomplikowana i mało kto sięga do jej matematycznych głębi. Szczytem matematycznym teorii grafów jest słynne zagadnienia czterech barw, którego rozwiązanie akurat się zbiegło **) z przygotowywaniem omawianej książki do druku. Ale, trawestując słowa Autora, rozwiązanie tego słynnego problemu, nad którym bezradnie ślęczały pokolenia matematyków, nie wniosło specjalnie nic nowego. Na razie wydaje się, że jest to matematyczna finezja, czy cztery barwy wystarczą do pokolorowania każdej mapy, czy też trzeba więcej barw. Karierę zrobiło natomiast pojęcie grafu jako wygodnego narzędzia modelowania. Jest tu pewna analogia do kariery matematyki hinduskiej — właściwą karierę zrobiły *cyfry hinduskie*, w dodatku w formie przetworzonej przez Arabów. Albo, żeby już użyć bardziej współczesnego przykładu, podobną karierę zrobiło słowo *macierz*.

Powyższa zimna woda lana na rozpalone głowy entuzjastów stosowania teorii grafów, nie ma na celu odstraszania ich od lektury książki Korzana, lecz ostrzeżenie, że jest to tylko podbudowa naukowa dla coraz powszechniej stosowanego, a może często wręcz nadużywanego pojęcia. Przy okazji bowiem Autor podjął pewną próbę uporządkowania polskiej terminologii teoriografowej ***). Co najwyżej można mieć do niego pretensję, że nie zamieścił słowniczka angielsko-polskiego. Jest to o tyle ważne, że pewne pojęcia teorii grafów, wykorzystywane w analizie powiązań, są często w praktyce zbyt bezpośrednio wiązane z sieciami czynnościowymi. Tymczasem choćby tak bujnie rozwijająca się ostatnio dziedzina informatyki, jaką jest problematyka baz danych, korzysta z całego szeregu innych sieci, które można interpretować jako grafy w pewnym sensie hierarchiczne, zwane hipergrafami.

Większość czytelników krajowych z problematyką teorii grafów na poziomie monograficznym zapoznała się z polskiego przekładu słynnego podręcznika Oysterna Ore. Od czasu napisania oryginału upłynęło tymczasem wiele lat, w ciągu których teoria grafów poczyniła znaczne postępy. Dlatego też recenzent z przyjemnością odnotowuje fakt podawania przez Autora daty sformułowania ważniejszych twierdzeń, a zwłaszcza twierdzeń o oszacowaniach tzw. liczb chromatycznych, jak np. Weinsteina (1963), czy też Diraca (1964). Na tle obszernej informacji bibliograficznej, doprowadzonej praktycznie do połowy bieżącej dekady, świadczy to ewidentnie o „świeżości monograficznej" omawianej książki. Co najwyżej można mieć pretensję do Korzana, że zbyt skromnie potraktował swoje przyczynki i uogólnienia.

Warto tutaj nadmienić, że w bibliograficznych wydawnictwach matematycznych, takich jak *Mathematical Review* czy też *Referatywnyj Żurnał*, teoria grafów obejmuje już szereg działów klasyfikacyjnych. Oznacza to praktyczną niemożliwość przedstawienia aktualnego stanu teorii grafów w tak małej książce. Jest to temat w ogóle raczej nie na książkę, zwłaszcza wydawaną w naszym cyklu wydawniczym, ile na referat przeglądowy na którymś z kongresów matematycznych. Natomiast „Elementy teorii grafów i sieci" są z założenia czymś innym: dziełem naukowym, popularyzującym w możliwie systematyczny i ścisły sposób zawiłe niekiedy pojęcia.

W sumie Korzan omówił ponad 300 różnych pojęć teoriografowych i pokrewnych, w tym ponad 30 różnych typów klasyfikacyjnych. Całość materiału ujęto w blisko 70 twierdzeń matematycznych, w tym 10 dotyczących hipergrafów i sieci, zilustrowanych starannie dobranymi prawie 50 przykładami. Istotnym uzupełnieniem dzieła jest dodatek omawiający algebry boolowskie oraz metodę programowania dynamicznego. Tutaj można by wysunąć zarzut, że definiując ogólnie algebry boolowskie i natychmiast potem ograniczając się do algebry dwuelementowej, można było wspomnieć o algebrach skończenie-wieloelementowych i twierdzeniu, że liczba tych elementów musi być zawsze całkowitą potęgą liczby 2. Byłoby to jednak może już wchodzenie w zbytnie finezje matematyczne, w zasadzie nie przeznaczone dla modelowego Czytelnika naszego miesięcznika.

Krótko więc mówiąc, „Elementy teorii grafów i sieci" są lekturą uzupełniającą, jaką szanujący się informatyk powinien przynajmniej posiadać w swej oszklonej biblioteczce i od czasu do czasu postudiować. Pamiętając jednak, że jest to lektura przeznaczona do wnikliwego zgłębienia i wręcz niestrawna przy próbie jednorazowego przełknięcia, ale za to po właściwym przyswojeniu dająca nowe punkty widzenia wielu ważnych zagadnień praktycznych.

Adam B. EMPACHER

*) Korzan Bohdan: Elementy teorii grafów i sieci. Metody i zastosowania. WNT, Warszawa 1978. Wydanie I, nakład 6000 egz. Ark. wyd. 15,4; 279 str., 76 rys; cena 32 zł

**) W języku polskim jedyną na razie publikacją, traktującą o rozwiązaniu (komputerowym) problemu czterech barw jest cykl popularnonaukowych artykułów w mies. „Młody Technik" (nry 1—4/79), opracowanych według oryginalnej pracy Appela i Hakena w Illinois Journal of Math.

***) Kłopoty terminologiczne istnieją zresztą już w samej literaturze źródłowej, zwłaszcza dotyczącej różnych dziedzin zastosowań teorii grafów. Przykładowo, występującemu czasem praktycznie w analizie powiązań stosunkowo prostemu pojęciu *grafu chronologicznego*, w omawianej monografii odpowiada *digraf acykliczny*, jako nazwa niemal wyłącznie stosowana przez matematyków.

Możliwości kształcenia informatyków we Francji

W chwili kiedy mówi się dużo o kształceniu informatyków, najczęściej podkreślając że jest ono niedostosowane lub niewystarczające w stosunku do potrzeb rynku zatrudnienia, warto przypomnieć jakie istnieją możliwości oraz główne placówki kształcenia w tej dziedzinie we Francji.

Co roku na rynku francuskim poszukuje pracy w informatyce ponad 15 tysięcy młodych dziewcząt i chłopców o różnym poziomie wykształcenia, co stanowi od 8% do 9% zatrudnionych w tym zawodzie. Wykształcenie to zapewnia im około 160 zakładów dydaktycznych podległych szkolnictwu państwowemu. Liczba tych szkół w ciągu ostatnich 10 lat powiększyła się więcej niż dwukrotnie.

W maju 1977 przeprowadzono ankietę dotyczącą struktury wykształcenia informatyków francuskich. Okazało się, że kwalifikacje do wykonywania zawodu informatyka osoby ankietowane uzyskały:

w 33% przypadków — na wyższych uczelniach
w 22% w średnich szkołach zawodowych
w 27% na zawodowych kursach specjalistycznych
w 18% — w inny sposób.

Kształcenie informatyków we Francji

Rodzaje specjalistów wg dziedzin działalności[1])	Rodzaj placówek dydaktycznych
● **Badania naukowe**	
główny projektant, analityk funkcji, analityk organizacji, programista	szkolnictwo państwowe w miejscu pracy, szkolnictwo państwowe, szkoły prywatne
● **Eksploatacja**	
Perforowanie przygotowanie i kontrola danych,	w miejscu pracy
Obsługa konsoli operatorskiej	w miejscu pracy, szkoły prywatne
● **Personel Kierowniczy**	
programista systemu	szkolnictwo państwowe
inżynier systemu	szkolnictwo państwowe, szkoły prywatne
Kierownik przygotowania danych	w miejscu pracy
Kierownik eksploatacji	w miejscu pracy
Kierownik prac badawczych	szkolnictwo państwowe

[1]) Nie wszystkie nazwy zawodów lub wykonywanych czynności mają swoje ścisłe odpowiedniki w języku polskim

Przewiduje się, że w najbliższych latach wzrośnie udział procentowy absolwentów szkół wyższych i średnich zawodowych (w roku 1970 wynosił on zaledwie 25%). Tabela pokazuje zjawisko znane, niestety, od lat: w szkolnictwie państwowym wyraźnie brak jest kształcenia użytkowników w zakresie wszystkich poziomów wiedzy informatycznej. Nadal główny nacisk kładzie się na przygotowanie kadr dla badań naukowych i projektowania systemów. Zresztą wydaje się, że żaden kraj nie potrafił rozwiązać tego problemu, który w rzeczywistości jest bardzo złożony z powodu różnorodności zastosowań informatyki. Różnorodność ta nie pozwala ani na ujednolicenie szkolenia, ani też na rozszerzenie zagadnień teoretycznych zastosowań.

Jeśli chodzi o zainteresowanie studentów kierunkiem studiów „informatyka", wydaje się, że nie ma już tak złej sytuacji jak w latach ubiegłych. Wynika z faktu to przede wszystkim, że doradcy w zakresie wyboru kierunku studiów nie zachęcają absolwentów szkół średnich (lub czynią to już w znacznie mniejszym stopniu) do wyboru tej specjalizacji, pamiętając pamiętny kryzys lat 1972—1973. Jest również faktem, że zawód ten zrobił się zwyczajnym, nie jest już taką nowością, stał się mniej nowatorski, skończyła się epoka pionierów. Wreszcie czyż nie mówi się coraz częściej, że w przyszłości zawód programisty nie będzie miał już racji bytu?

Jest prawie pewne, że w tym roku wszyscy informatycy, którzy opuszczą szkoły będą natychmiast zaangażowani. Jednak ogólne możliwości przyjęć na studia wpłyną raczej na pewne zmniejszenie się liczby kształconych informatyków (np. ograniczenia w Politechnikach). Należy zaznaczyć, że w tej ciągle rozwijającej się i nadal zapowiadającej liczne zmiany dziedzinie techniki prognozować jest szczególnie trudno.

Poniższy zwięzły przegląd obecnych możliwości kształcenia informatyków we Francji, mimo znacznych różnic formalnych i merytorycznych w hierarchii szkół, a także całkowicie odmiennej nomenklatury, może być ciekawy również dla osób interesujących się problematyką szkolenia informatycznego w Polsce.

Średnie szkoły zawodowe

Matura H[2]) otrzymana po zakończeniu średniej szkoły zawodowej (technikum) umożliwia łatwiejsze dostanie się do wyższych szkół technicznych[3]) lub politechnik[4]). Wydaje się, że tylko nieliczne przedsiębiorstwa przyjmują pracowników z samą maturą H, chyba, że umożliwiają one naszym pracownikom uzyskanie wykształcenia uzupełniającego np. w programowaniu albo też same szkolą na miejscu w zakresie eksploatacji maszyn. W zasadzie matura H nie jest również zbyt atrakcyjna dla uczniów, którzy zamierzają studiować informatykę, na wyższych uczelnich, ponieważ znacznie częściej zdają oni maturę C[5]) (matematyczno-fizyczną). Niestety z Ministerstwa Szkolnictwa nie można otrzymać najnowszych danych statystycznych dotyczących liczby absolwentów z informatyczną maturą H (w roku 1975 było ich 600).

Wyższe szkoły techniczne

Obecnie we Francji istnieje 8 wyższych szkół technicznych pozwalających uzyskać dyplom BTS w zakresie informatyki. Jednakże należy zauważyć, że spośród tych ośmiu szkół tylko jedna znajduje się w rejonie Paryża.

Dyplom Politechniki (DUT)[6]) wyraźnie góruje nad dyplomem wyższej szkoły technicznej (BTS), zarówno z punktu widzenia osobistych zainteresowań studentów, jak i potencjalnych pracodawców.

Politechniki

Politechniki zwane Uniwersyteckimi Instytutami Technologicznymi (IUT) istnieją od 1966 roku. Obecnie jest ich 22, w tym tylko 6 w rejonie Paryża. W latach 1972—1975 zaznaczyło się zmniejszenie liczby studentów tego typu uczelni, ale również przedsiębiorstwa zaczęły doceniać wartość tego typu absolwentów.

Uczelnie te nie zawsze były jednak dobrze przyjmowane w środowisku uniwersyteckim.

Ministerstwo Szkolnictwa podejmowało w przeszłości wiele działań mających na celu podwyższenie poziomu tych szkół Dziś wykształcenie to jest już w pełni doceniane, a absolwenci są bardzo poszukiwani przez przedsiębiorstwa. Można sądzić, że w sumie uczelnie te w bieżącym roku akademickim wypuszczą łącznie około 1700 informatyków. Niektórzy z nich (szacunkowo 1/3) podejmą dalszą naukę w celu otrzymania najbardziej atrakcyjnego dyplomu informatyka ze specjalności w dziedzinie zarządzania przedsiębiorstwem.

[2]) Franc. Bac H

[3]) Ukończenie daje dyplom BTS — (Brevet de technicien supérieur) — dyplom technika z wykształceniem wyższym

[4]) IUT (Institut Universitaire de Technologie = Uniwersytecki Instytut Technologiczny)

[5]) Franc. Bac C

[6]) DUT (Diplôme Universitaire de Technologie = dyplom uniwersytecki w zakresie technologii)

Uniwersytety [7])

Obecnie 28 Uniwersytetów pozwala uzyskać następujące rodzaje dyplomów:

— specjalista informatyk (franc. *Maîtrise d'informatique)*
— specjalista informatyk w dziedzinie zarządzania przedsiębiorstwem (franc. *Maîtrise d'informatique appliquée à la gestion des entreprises--Miage)*
— dyplom studiów poszerzonych (franc. *Diplôme d'études approfondies)* dyplom studiów wyższych (franc. *Diplôme d'études supérieures)* doktorat w dziedzinie informatyki (franc.: Doctorats en informatique)
— dyplom programisty-badacza i eksperta przetwarzania informacji (franc.: *Diplôme de programmeur d'etudes et d'expert en traitement de l'information)*

Na uniwersytety te dostają się absolwenci szkół dających dyplom ogólnych studiów uniwersyteckich (DEUG) [8]) bądź wspomniany już dyplom politechniki (DUT). Należy zaznaczyć, że kształcenie informatyczne odbywa się na kursach wieczorowych dla pracujących oraz, że istnieją sekcje kształcenia na poziomie elementarnym i zaawansowanym.

Szkoły inżynierskie [9])

Większość szkół inżynierskich w latach 1967—1970 włączyła informatykę do swego programu nauczania w formie zajęć obowiązkowych, bądź fakultatywnych. Obecnie absolwenci tych szkół legitymują się dyplomem specjalisty [10]) lub świadectwem ukończenia szkoły [11]), albo też otrzymują zaświadczenia o odbyciu określonego przeszkolenia w danej gałęzi informatyki [12]).

Towarzystwa AFPA [13]) **Kształcenia Zawodowego Dorosłych**

AFPA kształci programistów, analityków oraz specjalistów w dziedzinie zarządzania. Absolwenci tego typu kształcenia otrzymują dyplom państwowy.

Państwowy Instytut Sztuki i Rzemiosła [14])

Państwowy Instytut Sztuki i Rzemiosła wydaje świadectwo ukończenia studiów Państwowego Stowarzyszenia Rękodzieła i Rzemiosł (CNAM) [15]), równoważne dyplomowi politechniki (DUT) w zakresie informatyki lub dyplomowi szkoły inżynierskiej (DEST) [16])

Placówki awansu społecznego i doskonalenia zawodowego [17])

Po ukończeniu szkolenia w placówkach awansu społecznego i doskonalenia zawodowego można uzyskać wspomniany już dyplom uniwersytecki technologa (DUT). Świadectwo zawodowe [18]) lub zaświadczenie kwalifikacji zawodowych (CAP) [19]) albo dyplom programisty eksperta.

Szkolenie w dziedzinie zarządzania

Należy wymienić również szkoły i placówki kształcenia w dziedzinie zarządzania. Niektóre z nich dają świadectwo przeszkolenia w zakresie informatyki lub oferują zaświadczenie odbycia kursu z wybranej specjalizacji informatyki.

Prywatne placówki dydaktyczne

Prywatne placówki dydaktyczne w zakresie informatyki są prawdopodobnie jeszcze dość liczne (zarejestrowanych jest 38), ale nie są one obecnie uwzględniane w oficjalnych statystykach. Wiadomo jedynie, że liczba ich uczniów w ciągu ostatnich lat zmniejszyła się znacznie.

W latach, gdy szkoły prywatne wchodziły do roczników statystycznych wypuszczały one rocznie tyle samo absolwentów co placówki państwowe. Nadal przedsiębiorstwom oferuje się kursy kształcenia programistów. Kursy te organizowane są stosownie do określonych potrzeb. W rejonie Paryża aktualnie zauważyć można ożywienie tej działalności, ponieważ przedsiębiorstwa napotykają znaczne trudności przy rekrutacji odpowiednich specjalistów.

Opracowała Bogna Lichodziejewska na podstawie ZERO-UN-INFORMATIQUE, nr 123/1978

[7]) Franc. Universites

[8]) DEUG (Diplôme d'etudes universitaires générales = dyplom ogólnych studiów uniwersyteckich)

[9]) Franc. Ecole d'ingenieurs

[10]) Franc. Diplôme de spécialisation

[11]) Franc. Attestation

[12]) Franc. Mention de la filiére suivie

[13]) AFPA — Association pour la formation professionnelle des adultes

[14]) Franc. Conservatoire national des Arts et Métiers

[15]) CNAM — Confédération Nationale de l'Artisanat et des Métiers

[16]) DEST — Diplôme d'ecole supérieure de technique = dyplom szkoły inżynierskiej

[17]) Franc. Organismes de promotion Sociale et de perfectionnement

[18]) Franc. Brevet professionnel

[19]) CAP — Certificat d'aptitude professionnell

KSPZ—komputerowy system planowania zajęć

W wielu ośrodkach prowadzi się badania nad zastosowaniem komputerów do wyznaczania rozkładów zajęć. W Polsce próby wdrożenia podejmowane były między innymi na Wydziale Społeczno-Ekonomicznym Uniwersytetu Łódzkiego, na Politechnice Gdańskiej i w Wojskowej Akademii Technicznej.

Komputerowy System Planowania Zajęć (KSPZ), opracowany w Akademii Ekonomicznej w Poznaniu [1]), wykorzystywany jest do wyznaczania rozkładów zajęć na studiach stacjonarnych. System ten jest eksploatowany od roku 1977.

CHARAKTERYSTYKA SYSTEMU

Za pomocą KSPZ można planować zajęcia wszystkich typów, tzn. wykłady, ćwiczenia, laboratoria, lektoraty, seminaria, wychowanie fizyczne itp. Wymagane jest jednak dokładne określenie struktury grup zajęciowych [2]). Za podstawową grupę planistyczną w AE Poznań przyjęto grupę dziekańską. Dla zajęć prowadzonych w innych grupach należy rozstrzygnąć, do jakich grup dziekańskich należą studenci odbywający dane zajęcie.

KSPZ wyznacza rozkłady dla grup dziekańskich i w miarę potrzeby dla innych grup zajęciowych (np. grupy seminaryjne, lektorskie, laboratoryjne), rozkłady zajęć wykładowców oraz rozkłady obciążeń sal. System stwarza przy tym możliwości:

— zblokowania zajęć studentów w określonej porze dnia (np. przed lub po południu),
— uwzględnienia w rozkładach zajęć studentów i wykładowców dni i godzin wolnych od zajęć
— blokowania w określonych dniach i godzinach sal dydaktycznych
— planowania wskazanych zajęć w określonej sali, dniu i godzinie
— uwzględnienia indywidualnych preferencji wykładowców dotyczących wyboru sal, dni i godzin w których chcieliby prowadzić zajęcia oraz maksymalnej liczby godzin zajęć w dniu
— umieszczania w różnych dniach tygodnia zajęć, które nie powinny odbywać się tego samego dnia

[1]) KSPZ opracowany został przez Asystencko-Studencki Zespół Badawczy „Plan" przy Zakładzie Ekonometrii AE Poznań

[2]) Przez strukturę grup zajęciowych rozumie się zachodzące pomiędzy nimi relacje zawierania i częściowego zawierania. Tak więc w notacji graficznej nie musi być ona drzewem

— dowolnego kształtowania ilości i długości jednostek zajęciowych, momentów ich rozpoczęcia i zakończenia oraz przerw

— uwzględnienia przerw obiadowych.

Oprócz głównej funkcji, jaką jest wyznaczanie rozkładów zajęć, system umożliwia również:

— sporządzenie wykazów obciążeń dydaktycznych wykładowców oraz planu studiów w układzie grup dziekańskich

— sporządzenie bilansów zapotrzebowań na sale i ich dostępności

— wprowadzanie do rozkładów dowolnych zmian i poprawek oraz wydrukowanie aktualnych rozkładów po dokonaniu poprawek.

Zajęcia wprowadzane są do rozkładu zajęć sukcesywnie według nadanych im priorytetów [3]) i kolejności umieszczenia w zbiorze danych, w godzinach preferowanych przez wykładowców i dopuszczalnych dla studentów oraz w odpowiedniej sali. Jeżeli nie można spełnić tych warunków i postulatu unikania w rozkładach „okienek", wówczas o dalszym postępowaniu decyduje hierarchia priorytetów [4]). Może być ona zmieniana i dostosowywana do warunków konkretnej uczelni.

[3]) Każdemu zajęciu przyporządkowuje się pewien priorytet według czterostopniowej skali

[4]) Hierarchia priorytetów jest bardzo rozbudowana i określa na przykład czy w przypadku konfliktu preferencji „pierwszeństwo" mają preferencje wykładowcy czy studentów, czy „lepsze jest okienko" czy zaplanowanie zajęć w niezbyt odpowiedniej sali

DANE WEJŚCIOWE

Dane dotyczące grupy dziekańskiej składają się z czterech części:

1) zbioru zajęć grupy, w którym dla każdego zajęcia określa się: nazwę przedmiotu, osobę prowadzącą dane zajęcie, jego czas trwania, inne grupy dziekańskie realizujące razem dane zajęcie, podział grupy (grup) na podgrupy zajęciowe, termin realizacji zajęcia i salę (jeżeli zostały przesądzone) oraz priorytet zajęcia

2) preferencje godzinowe grupy; każdą jednostkę zajęciową ocenić można w skali czteropunktowej od 0 do 3 punktów. 0 pkt oznacza godzinę preferowaną, a 3 pkt. godzinę niedopuszczalną (zablokowaną)

3) listy zajęć, które nie powinny odbywać się tego samego dnia

4) listy sal, w których należy planować wskazane zajęcia

PROGRAM FHR6 DATA WYKONANIA 14/02/78 CZAS 18/33/45

ROZKŁAD ZAJEC GRUPY

AKADEMIA EKONOMICZNA EKONOMIA I ORGANIZACJA HANDLU ZAGRANICZNEGO ROK III GRUPA 3 2373
POZNAN

GODZ	PONIEDZIAŁEK	WTOREK	SRODA	CZWARTEK	PIATEK	SOBOTA
8/00		SZKOL. WOJSKOWE	FILOZOFIA MARK. W DOC. FOKINA SALA 207 BANK	PSYCHOL. SPOŁECZ. PROF. TALEJKO SALA 206 BANK	JEZYK ROSYJSKI	
8/50	PLAN. PROGN. HZ W DOC. WOJCIECHOWSKI SALA 206 BANK	SZKOL. WOJSKOWE	FILOZOFIA MARK, W DOC. FOKINA SALA 207 BANK		JEZYK ANGIELSKI JEZYK NIEMIECKI	
9/40	PLAN, PROGN. HZ W DOC. WOJCIECHOWSKI SALA 206 BANK	SZKOL. WOJSKOWE		JEZYK ANGIELSKI JEZYK NIEMIECKI	MET. NAUK EKONOM, PROF. L. NOWAK SALA V	
10/40	JEZYK ROSYJSKI	SZKOL. WOJSKOWE	MARKETING WHZ W DOC. FOLTYNSKI SALA V	JEZYK ROSYJSKI	WF-BASEN	
11/30	JEZYK ANGIELSKI JEZYK NIEMIECKI	SZKOL. WOJSKOWE	EKONOMIKA HZ W DOC. FOLTYNSKI SALA 203 BANK	FILOZOFIA MARC. C MGR J. SIKORA SALA I	WF-BASEN	
12/20	PLAN PROGN. HZ C MGR TRAWA SALA 418 TOWAROW	SZKOL. WOJSKOWE	MARKETING W HZ C MGR Z. JANCZAK SALA I	FILOZOFIA MARK· C MGR J. SIKORA SALA.	WF-BASEN	
13/10	PLAN PROGN. HZ C MGR TRAWA SALA 418 TOWAROW	SZKOL. WOJSKOWE	JEZYK ANGIELSKI JEZYK NIEMIECKI	OCHRONA SRODOW. DOC. MŁYNAREK SALA 203 BANK	EKONOMIKA HZ MGR RYNARZEWSKI SALA 218 TOWAROW	
14/00		SZKOL. WOJSKOWE	MARKETING W HZ C MGR Z. JANCZAK SALA VII	OCHRONA SRODOW. DOC. MŁYNAREK SALA 203 BANK	EKONOMIKA HZ E MGR RYNARZEWSKI SALA 218 TOWAROW	
14/50						
15/40						
16/40						
17/30						
18/20						
19/10						

Rysunek 1

- Każdy wykładowca wypełnia „Arkusz preferencji godzinowych wykładowcy", oceniając wszystkie jednostki zajęciowe również w skali czteropunktowej, oraz podaje maksymalną liczbę godzin zajęć w dniu
- Dla każdej sali dydaktycznej w „Arkuszu dostępności sali" określa się jej typ (np. audiowizualna, laboratorium językowe), pojemność oraz godziny niedopuszczalne (zablokowane)
- Informacje dodatkowe dotyczące ilości i czasu trwania jednostek zajęciowych, hierarchię priorytetów itp.

Zbiór danych poddawany jest automatycznej, szczegółowej kontroli formalnej i logicznej. W celu przeprowadzenia kontroli merytorycznej drukowane są tabulogramy kontrolne planu studiów, obciążeń wykładowców i preferencji godzinowych.

DOKUMENTY WYNIKOWE

KSPZ wyprowadza tabulogramy zawierające: rozkłady zajęć grup dziekańskich (patrz rys. 1) oraz podgrup, rozkłady zajęć wykładowców (rys. 2) rozkłady obciążeń sal dydaktycznych (rys. 3) oraz listę zajęć nie zaplanowanych. Po analizie otrzymanych rozkładów można nanieść w nich dowolne poprawki i powtórnie wydrukować.

PARAMETRY TECHNICZNE

KSPZ może być eksploatowany na maszynach ODRA serii 1300 oraz ICL 1900 w konfiguracji dyskowej z minimalną pojemnością pamięci operacyjnej 64 K słów. Podstawowym nośnikiem danych wejściowych są karty dziurkowane. W skład systemu obok programów standardowych wchodzą trzy wielomodułowe programy opracowane i uruchomione przez autorów systemu: program kontroli danych, główny program obliczeniowy i program edytorski. Cechuje je duża niezawodność [5]).

W eksploatowanej wersji systemu liczba wykładowców, grup i sal nie może przekroczyć pięciuset, nie ma natomiast praktycznie ograniczeń co do liczby zajęć. Czas pracy komputera ODRA 1305 przy planowaniu zajęć na jeden semestr łącznie na wszystkich etapach sporządzania rozkładu w warunkach AE Poznań waha się od 10 do 13 godzin [6]).

ROZWÓJ SYSTEMU

KSPZ jest systematycznie rozwijany i doskonalony. Zamierza się przystosować go do potrzeb planowania zajęć w szkołach podstawowych i średnich oraz uczelni o rozproszonej lokalizacji sal dydaktycznych. Ponadto prowadzone są prace nad uwzględnieniem dodatkowych kryteriów, jak np. minimalizacja przemieszczeń studentów i wykładowców z sali do sali i „okienek" w rozkładach, uwzględnienie niektórych dodatkowych postulatów wykładowców itp. Z drugiej strony stale ulepsza się programy obliczeniowe z punktu widzenia skrócenia czasu przebiegów.

[5]) Obok standardowych procedur wznawiania obliczeń główny program obliczeniowy ma możliwość zawieszania i wznawiania pracy w każdym momencie, także w przypadku awarii komputera

[6]) 3 wydziały, ponad 110 grup dziekańskich, 300 podgrup i 300 wykładowców, około 2000 zajęć, 40 sal (wykorzystywanych prawie w 100%). 14 jednostek zajęciowych w dniu

PROGRAM FHR6 DATA WYKONANIA 20/09/77 CZAS 13/13/12

ROZKŁAD ZAJEC WYKŁADOWCY

AKADEMIA EKONOMICZNA

POZNAN MGR H. RUNKA 5239

GODZ	PONIEDZIAŁEK	WTOREK	SRODA	CZWARTEK	PIATEK	SOBOTA
8/00						
8/50						
9/40						
10/40						
11/30						
12/20						
13/10	EKONOMETRIA C E 00 U R2 G6 SALA 107 BANK					
14/00	EKONOMETRIA C E 00 U R2 G6 SALA 107 BANK					
14/50	EKONOMETRIA C E 00 U R2 G6 SALA 107 BANK					
15/40				PROGR. EMC D. P. EK INF. I CYB. R4 G1 SALA III		
16/40			EKONOMETRIA C E 00 U R2 G5 SALA III	PROGR. EMC D. P. EK INF. I CYB. R4 G1 SALA III		
17/30			EKONOMETRIA C E 00 U R2 G5 SALA III	EKONOMETRIA C E 00 U R2 G4 SALA III		
18/20			EKONOMETRIA C E 00 U R2 G5 SALA III	EKONOMETRIA C E 00 U R2 G4 SALA III		
19/10				EKONOMETRIA C E 00 U R2 G4 SALA III		

Rysunek 2

PROGRAM FHR6 DATA WYKONANIA 28/97/78 CZAS 10/54/17

SALA 218 TOWAROW

AKADEMIA EKONOMICZNA 3218
POZNAN

GODZ	PONIEDZIAŁEK	WTOREK	SRODA	CZWARTEK	PIATEK	SOBOTA
08/00 08/45	MGR M. JAGIELSKI EK. POLIT. SOCJ. C E I OP R2 G2	MGR M. JAGIELSKI EK. POLIT. SOCJ. C E I OP R2 G5	DR SOBOLEWSKI E-KA PRZEDSIEB. C EK-SPOL. R3 G1	DR T. JUJA MET. I TECH. PL. W PL. I FIN. R3 G1	MGR M. ROSZAK EKON. POL. KAPIT. C E OHZ R1 G3	
08/50– 09/35	MGR M. JAGIELSKI EK. POLIT. SOCJ. C E I OP R2 G2	MGR M. JAGIELSKI EK. POLIT. SOCJ. C E I OP R2 G3	DR B. PILARCZYK EK. OBR. I USLUG C E 00 U R3 G2	TECH. PRACY UMYSL ORG I ZARZ. R1 G2	MGR M. ROSZAK EKON. POL. KAPIT. C E OHZ R1 G3	
09/40– 10/25	MGR. M JAGIELSKI EK. POLIT. SOCJ. C E I OP R2 G2	MGR M. JAGIELSKI EK. POLIT. SOCJ. C E I OP R2 G3	DR W. PIOTR EKON. POL KAPIT. C EK. TRANS R1 G1	DOC. N. FOKINA FILOZOFIA MARK. C EOHZ R3 G2	MGR A. JANC POL. EK. I PLAN C. E OHZ R3 G2	
10/40– 11/25		MGR E. FRACKO- WIAK ORG. I ZARZADZ C E OHZ R1 G3	DR W. PIOTR EKON. POL KAPIT. C EK. TRANS R1 G1	DOC. N. FOKINA FILOZOFIA MARK. C E OHZ R3 G2	MGR A. JANC POL. EK. I PLAN C E OHZ R3 G2	
11/30– 12/15	MGR. I. CIESIELSKA PODST. RACHUNK. E OHZ R2 G2	DR A. MICHALAK SEMINARIUM KURS. E 000 R2 G2	DR L. BIELOWSKA SEMINARIUM KURS. OZ R2 G4	DR I. BIERNAWSKA PODS-. RACHUNK. W ORG. I ZARZ. R2	TECHN. PRACY UMYSL. E 00 U R1 G2	
12/20– 13/05	MGR. I. CIESIELSKA PODST. RACHUN. C E OHZ R2 G2	DR A. MICHALAK SEMINARIUM E 00 U R2 G2	DR L BIELOWKA SEMINARIUM OZ R2 G4	DR I. BIERNAWSKA PODST. RACHUNK. ORG. I ZARZ. R2	TECH. PRACY E I OP R1 G6	
13/10– 13/55	MGR R. KARLIK SEMINARIUM KURS. EHZ R2 G4	DR Z. POTOK SEMINARIUM KURS. ER R2 G1	DOC. E. BITTNER SEMINARIUM KURS. EP R2 G2	MGR G. PLEWIŃSKA SEMINARIUM KURS. PF R2 G3	DR W. IGNATCZAK SEMINARIUM KURS. CEI R2 G3	
14/00– 14/45	MGR. R. KARLIK SEMINARIUM KURS. EHZ R2 G4	DR Z. POTOK SEMINARIUM KURS. ER R2 G1	DOC. E. BITTNER SEMINARIUM KURS. EP R2 G2	MGR G. PLEWINSKA SEMINARIUM KURS. PF R2 G3	DR W. IGNATCZYK SEMINARIUM KURS. PF R2 G3	
14/50– 15/35	MGR R. JAKUBOW- SKI PODST. RACHUN. C ORG. I ZARZ. R2 G1	DR M. HOPPE EKON. POLIT. KAPIT. C CE I R1 G2	MGR E. URBANOW- SKA TECHNOL. HANDLU C E 0 U R3 G3	MGR R. JAKUBOW- SKI PODST. RACHUNK. C ORG. I ZARZ. R2 G3	MGR R. JAKUBOW- SKI PODST. RACHUNK. C ORG. I ZARZ. R2 G2	
15/40– 16/25	MGR R. JAKUBOW- SKI PODST. RACHUN. C ORG. I ZARZ. R2 G1	DR. M. HOPPE EKON. POLIT. KAPIT. C CE I R1 G2	MGR E. URBANOW- SKA TECHNOL. HANDLU C E 00 U R3 G3	MGR R. JAKUBOW- SKI PODST. RACHUNK. C ORG. I ZARZ. R2 G3	MGR R. JAKUBOW- SKI PODST. RACHUNK. C ORG. I ZARZ. R2 G2	
16/40– 17/25	DR W. WEGNER PODST. NAUK POL. C EK-SPOL. R3 G2		MGR R. JAKUBOW- SKI PODST. RACHUNK. C PL. I FIN. R2 G2	MGR J. PIWOSZ GOSPOD. BUDŻETOWA C PL. I FIN. R3 G2	MGR P. KUCHLEW- SKI PODST. RACHUNK. C E OHZ R2 G1	
17/30– 18/15	DR W. WEGNER PODST. NAUK POL. C EK-POL. R3 G2		MGR J. JAKUBOW- SKI PODST. RACHUNK. C PL. I FIN. R2 G2	MGR J. PIWOSZ GOSP. BUDŻETOWA C PL. I FIN. R3 G2	MGR P. KUCHLEW- SKI PODST. RACHUNK. C E OHZ R2 G1	
18/20– 19/05	MGR K. GOLATA PODST. NAUK POL. C E OHZ R4 G2		DR A. GACARZE- WICZ PL. I FIN. ROZW. W PL. I FIN. R4 G2	MGR W. BACHORZ FIN. ORG. GOSP. C PL. I FIN. R3 G2	MGR R. STAWAR- SKA MIEDZ. STOS. GUS. C E OHZ R2 G1	
19/10– 19/55	MGR K. GOLATA PODST. NAUK POL. C E OHZ R4 G2		DR K. KRUSZKA ILOSC. MET. RAD. W PL. I FIN. R4 G2	MGR W. BACHORZ FIN. ORG. GOSP. C PL. I FIN. R3 G2	MGR R. STAWAR- SKA MIEDZ. STOS. GUS. C E OHZ R2 G1	

Rysunek 3

Jan POLOWCZYK, Henryk RUNKA, Zbigniew RZEMYKOWSKI,
Wojciech SIKORA
Zakład Ekonometrii Akademii Ekonomicznej, Poznań

JERZY PASULA
Centrum Projektowania i Zastosowań Informatyki ZETO
Warszawa

Maszyny bazy danych

W przeciągu ostatnich kilku lat daje się zauważyć wyraźny wzrost zainteresowania badaniami nad nową architekturą maszyn cyfrowych. Szybki rozwój systemów przetwarzania danych i wzrost wymagań stawianych tym systemom doprowadził do odkrycia, że klasyczna architektura von Neumanna jest być może optymalna dla obliczeń numerycznych, ale znacznie mniej efektywna dla przetwarzania danych.

Jednym z najciekawszych kierunków badań w ramach poszukiwania optymalnej architektury maszyny cyfrowej są próby wykorzystania nowego sprzętu w dziedzinie systemów zarządzania bazą danych. Powstałe w wyniku tych badań urządzenia są często określone wspólną nazwą „maszyny bazy danych" (*ang. data base machines*). Pod tą nazwą mieści się bardzo szeroka klasa różnych urządzeń począwszy od inteligentnych jednostek sterujących pamięci zewnętrznej poprzez specjalizowane komputery zastępujące klasyczny system zarządzania bazą danych, aż do dużych maszyn stanowiących węzeł danych w sieci komputerowej.

W niniejszym artykule przedstawiono przyczyny, dla których właśnie teraz ten kierunek badań rozwija się tak dynamicznie, a także korzyści jakich można się spodziewać w końcowym rezultacie zainicjowanych prac. Następnie sklasyfikowano i poddano ogólnej ocenie te rozwiązania, które zostały już zrealizowane i opisane w literaturze. Próba oceny przyszłości maszyn bazy danych stanowi podsumowanie artykułu.

PRZYCZYNY PRZYSPIESZONEGO ROZWOJU

Punktem odniesienia dla badań nad maszyną bazy danych jest „klasyczny" system zarządzania bazą danych. Jest to system, którego wszystkie funkcje są realizowane przez oprogramowanie działające na sprzęcie o klasycznej architekturze von Neumanna. Można śmiało powiedzieć, że takie właśnie systemy zarządzania bazą danych zdominowały przetwarzanie danych na przestrzeni ostatnich dziesięciu lat. Z jednej strony umożliwiły rozwiązanie wielu problemów wcześniej nie rozwiązalnych i stworzyły wiele nowych możliwości. Jednocześnie jednak uświadomiły swym twórcom, a przede wszystkim użytkownikom o ile więcej można by osiągnąć, gdyby nie pewne ich ograniczenia i niedoskonałości. Na przestrzeni ostatnich lat systemy te bardzo się rozrastały i komplikowały.

Zmiany te były podyktowane zarówno przez wymagania użytkowników jak i sformułowanie wyszukanych modeli danych i zmianę trybu przetwarzania z wsadowego na bezpośredni. Najlepszą ilustracją opisanych procesów stanowi raport ANSI/X3/SPARC [1] określający przyszłościową architekturę systemu zarządzania bazą danych. Żaden z istniejących systemów tej klasy nawet w przybliżeniu nie odpowiada wymaganiom tego raportu. Co więcej, byłby to system tak rozbudowany, zawierający tak wiele mechanizmów pośredniczących dla wszystkich procesów definicji i manipulacji danych, że nie jest praktycznie możliwe jego zrealizowanie bez przebudowania sprzętu. Wynika to przede wszystkim z olbrzymich kosztów działania tak rozbudowanego oprogramowania (w sensie zużywanych zasobów systemów), jak również z degradacji parametrów eksploatacyjnych systemu, takich jak niezawodność, czas reakcji, przepustowość itp.

Jeszcze groźniej brzmią prognozy dotyczące spodziewanego wzrostu skali zastosowań systemów zarządzania bazą danych. Wg Hsiao [2] należy się liczyć z zapotrzebowaniem na system, który będzie zdolny obsłużyć do 10 milionów odwołań do bazy danych na sekundę, a obsługiwana przez ten system baza danych może zawierać do jednego biliona bajtów. Dla porównania można przytoczyć odpowiednie dane dla współczesnego systemu zarządzania bazą danych. Jest on zdolny do przetwarzania od 10 do 100 odwołań do bazy danych na sekundę, a największe eksploatowane bazy danych sięgają ok. 100 mld bajtów. Należy również podkreślić przepaść dzielącą poziom żądanej niezawodności przyszłościowego systemu i niezawodność współczesnych systemów zarządzania bazą danych, opartych tylko na oprogramowaniu.

Powyższe przyczyny spowodowały rozpoczęcie badań nad nową architekturą systemów zarządzania bazą danych, a obecnie daje się już zaobserwować gwałtowny rozwój w tej dziedzinie. Wynika on z olbrzymich postępów jakie poczyniono ostatnio w dziedzinie sprzętu. A oto niektóre najistotniejsze elementy tego rozwoju:

- **masowa produkcja mikroprocesorów i mikrokomputerów.** Mikroprocesor to układ elektroniczny jednostki centralnej umieszczony w jednym pakiecie układu scalonego o powierzchni ok. 12 cm². Natomiast mikrokomputer to jednostka centralna + pamięć operacyjna umieszczone na jednym lub kilku takich pakietach.

Ważniejszy od miniaturyzacji jest koszt tych układów — niezwykle niski, bo dochodzący już do kilkudziesięciu dolarów za sztukę. Tak więc rozwój mikroprocesorów i mikrokomputerów uczynił ekonomicznie możliwe rozproszenie inteligencji systemu i lokalizowanie jej np. w urządzeniach zewnętrznych, jednostkach sterujących i kanałach. Co więcej procesor centralny może być zastępowany przez zespół mikroprocesorów pracujących równolegle.

- **nowe technologie pamięci.** Nie są one jeszcze w powszechnym użyciu, ale nie ulega wątpliwości, że w najbliższej przyszłości będą produkowane masowo i zrewolucjonizują architekturę maszyn cyfrowych. Tradycyjne pamięci o dostępie bezpośrednim były zdominowane przez dwie, wyraźnie różne technologie, określające podział na pamięć główną (operacyjną) i zewnętrzną:

1) pamięć ferrytowa, a później MOS, o czasie dostępu ok. 1 milisekundy, lecz dosyć droga

2) pamięć na wirujących nośnikach magnetycznych o czasie dostępu 10—100 milisekund, ale znacznie tańsza.

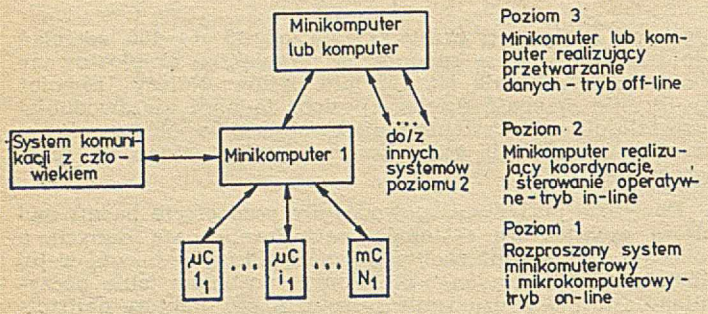
Nowe obiecujące technologie pomięci, jakie się ostatnio pojawiły, wypełnią lukę pomiędzy ww. rozwiązaniami oferując pośrednie czasy dostępu i koszty. Posiadanie bogatego wachlarza różnych typów pamięci pozwoli na zaprojektowanie zupełnie nowej architektury systemów. Te nowe technologie pamięci to m. in. pamięć pęcherzykowa (*ang. bubble memory*), pamięć półprzewodnikowa typu CCD (*ang. charge-coupled memory*), czy wreszcie pamięć adresowana strumieniem elektronów EBAM (*ang. electron beam addressable memory*).

- **rozwój urządzeń mikroprogramowanych.** Mikroprogramowanie dostarcza szerokich możliwości adaptowania istniejących urządzeń dla potrzeb systemów zarządzania bazą danych. Istnieje możliwość przygotowania takiego mikrokodu, że programista używający „języka maszyny" będzie posługiwał się instrukcjami o dosyć już złożonych funkcjach, wyspecjalizowanymi do działania na bazie danych. W skrajnym przypadku istnieje możliwość zrealizowania języka wyższego rzędu, wykonywanego bezpośrednio na maszynie (za pośrednictwem mikrokodu). Należy jednak podkreślić, że w systemach takich podstawowa architektura pozostaje bez zmian, poprawa parametrów eksploatacyjnych będzie więc przede wszystkim wynikiem przyspieszonej pracy urządzeń mikroprogramowanych. Przyspieszenie to wynika z działania na pamięci sterującej, która jest znacznie szybsza od pamięci operacyjnej.

ISTNIEJĄCE ROZWIĄZANIA

Dla potrzeb niniejszego opracowania zastosowano następującą klasyfikację maszyn bazy danych:
1) inteligentne urządzenia pamięci zewnętrznej
2) specjalizowane procesory bazy danych
3) węzły danych.

● **inteligentne urządzenia pamięci zewnętrznej** *(ang. intelligent controllers).* Ich strukturę scharakteryzowano na rys. 1.



Rys. 1. Inteligentne jednostki sterujące pamięci zewnętrznej

Jest to klasa urządzeń charakteryzująca się tym, że pewna część funkcji obsługi dostępu do bazy danych została przeniesiona z jednostki centralnej do jednostki sterującej pamięci zewnętrznej. W najprostszym przypadku będą to procedury dostępu do danych jak np. wykrywanie i korekta błędów, synchronizowanie ruchów głowic i przeglądanie zawartości rekordów.

Najbardziej rozbudowane urządzenia tego typu to specjalne procesory bazy danych rezydujące w jednostce sterującej pamięci. Wykorzystują one ideę pamięci asocjacyjnej, a więc takiej, w której dane są lokalizowane w oparciu o ich zawartość informacyjną (a nie o dostarczony z zewnątrz adres). Co więcej, operacje wyszukiwania danych są wykonywane bezpośrednio w pamięci zewnętrznej, tzn. nie jest wymagane przesyłanie przeglądanych danych do bufora w pamięci operacyjnej.

Maszyny te są oparte na pamięciach, gdzie jedno urządzenie czytająco-piszące przypada na każdą ścieżkę (*ang. head-per-track*). Do klasy tej należą m. in. pamięć bębnowa, pamięć dyskowa ze stałymi głowicami, a także pamięć pęcherzykowa itp. Dzięki temu, że pamięć została podzielona na strefy (*ang. cells*), a każda strefa ma przydzielone urządzenia czytająco-piszące i procesor, zawartość całej pamięci może być przeglądana i ewentualnie reorganizowana jednocześnie.

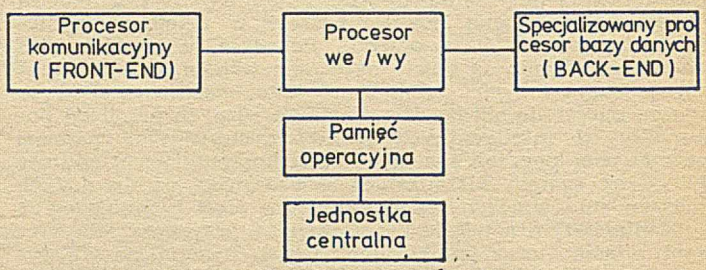
Dla dysku czy bębna, gdzie strefą jest ścieżka — czas wykonania selekcji na całej zawartości pamięci będzie równy czasowi jednego obrotu urządzenia. W czasie gdy każdy rekord danych przesuwa się (mechanicznie lub elektronicznie) pod głowicą czytającą odpowiedniej strefy, następuje jego porównanie z maską, a wynik porównania powoduje ustawienie odpowiedniego wskaźnika. I tak np. dla bazy danych wielkości 160 000 rekordów po 100 bajtów, czas selekcji będzie wynosił 10 msek. Tak więc czas dostępu jest o rzędy wielkości krótszy niż dla rozwiązań konwencjonalnych. Również istotną zaletą jest możliwość ciągłej reorganizacji fizycznej zawartości bazy danych i możliwość dowolnego rozszerzenia jej wielkości poprzez badanie nowych stref. Należy podkreślić, że ta ostatnia operacja nie powoduje żadnych zmian czasu reakcji systemu. Zastosowanie tego rodzaju urządzenia ma również duży wpływ na architekturę całego systemu zarządzania bazą danych przez to, że dane nie mają ustalonych adresów fizycznych. Nie ma potrzeby stosowania żadnych technik przyspieszających dostęp do danych jak np. indeksowanie, które stwarzają tak wiele problemów przy aktualizacji. Najistotniejszą wadą tych urządzeń, która nie pozwala na powszechne ich wprowadzenie, jest wysoki koszt pamięci, które mogą być tu zastosowane. Jednak ze względu na szybki rozwój nowych technologii pamięci, obniżenie kosztów jest tylko kwestią czasu.

Jednym z pierwszych urządzeń tego typu był Context Addressed Segment — Sequential Memory — CASSM [3], w którym główny nacisk położono na zaprojektowanie optymalnej struktury danych i efektywnych procesów selekcji w oparciu o przedstawioną powyżej ideologię.

Rotating Associative Relational Storage — RARES [4] został zaprojektowany dla uzyskania pamięci asocjacyjnej o wysokiej wydajności dla zrealizowania relacyjnej bazy danych. Sprzęt RARES współpracuje z optymalizatorem pytań SQUIRAL — dla relacyjnych języków dostępu.

Rotating Associative Processor RAP [5] także zaprojektowano dla relacyjnych baz danych. Jest on bardzo podobny do CASSM z tym jednak, że został opracowany jako samodzielna maszyna. Wyposażono go we własny język, który służy do pisania programów obsługi zapytań napływających do systemu.

● **specjalizowane procesory bazy danych** *(ang. back — end processors).* Strukturę tych urządzeń charakteryzuje rys. 2.



Rys. 2. Specjalizowany procesor bazy danych

Jest to rozwinięcie idei inteligentnego urządzenia sterującego pamięci w tym sensie, że wszystkie funkcje tradycyjnego systemu zarządzania bazą danych zostały usunięte z maszyny głównej i umieszczone w procesorze bazy danych. Określenie „back-end" powstało przez analogię do „front-end" będącego specjalizowanym procesorem zarządzającym komunikacją danych.

Z punktu widzenia architektury systemu — procesor bazy danych umieścić można pomiędzy maszyną główną, a inteligentnymi urządzeniami pamięci zewnętrznej. Co więcej, jest nawet możliwe, że będzie on związany ze zbiorem inteligentnych jednostek sterujących pamięci. Ma on wyższy stopień funkcjonalności, zazwyczaj stanowi bowiem implementację całego systemu zarządzania bazą danych.

Różnice pomiędzy poszczególnymi rozwiązaniami tej klasy dotyczą przede wszystkim sposobu, w jaki została podzielona praca pomiędzy maszyną główną i specjalizowanym procesorem bazy danych. Można wyróżnić dwie podstawowe grupy rozwiązań w tej klasie: z zastosowaniem standardowego minikomputera jako procesora bazy danych [6] oraz z zastosowaniem urządzenia o specjalnej niestandardowej architekturze [7]. Database Computer-DBC opisany w [7] jest najbardziej kompletną i najciekawszą propozycją w prezentowanej dziedzinie. Zawiera on specjalną — wieloprocesorową — architekturę zapewniającą olbrzymią wydajność, krótki czas reakcji, doskonałą ochronę dostępu do danych i możliwość obsłużenia każdego modelu danych i każdego języka manipulowania danymi.

Jest to system o przetwarzaniu potokowym (*ang. pipeline system*), w którym operacje wykonywane w ramach obsługi żądań nadsyłanych z maszyny głównej zostały podzielone na procesy, wykonywane przez niezależne procesory. Uzyskano dzięki temu olbrzymią wydajność systemu i bardzo krótki czas reakcji. W systemie działają dwie pętle obsługi żądań napływających z maszyny głównej. Pierwsza pętla działa na opisie struktury bazy danych, natomiast druga na samych danych formułując odpowiedzi na obsługiwane żądania. W zakresie działania na fizycznej bazie danych, DBC wykorzystuje opisaną wyżej ideę pamięci asocjacyjnej. Jednakże dla znacznego obniżenia kosztów baza danych została umieszczona na dyskach o ruchomych głowicach.

Umożliwia to równoległe przeszukiwanie tylko części bazy danych (po jednym cylindrze każdego pakietu dyskowego). Aby zminimalizować ilość przeszukiwanych cylindrów zastosowano specjalne algorytmy grupowania danych powiązanych logicznie. Sam opis struktury bazy danych został umieszczony na dyskach o stałych głowicach, ażeby maksymalnie przyspieszyć operacje wykonywane na opisie.

Poza wspaniałymi perspektywami osiągnięcia bardzo dobrych parametrów eksploatacyjnych, idea specjalizowanego procesora bazy danych dostarcza jeszcze innych obiecujących możliwości. Można go sobie wyobrazić jako most, pozwalający łagodnie przechodzić z jednej jednostki centralnej na inną. Wynika to stąd, że nie ma żadnych przeciwwskazań opracowania wielu mechanizmów pośredniczących dla wielu różnych jednostek centralnych, co z kolei umożliwi ich łatwą komunikację. Ponieważ procesor bazy danych jest urządzeniem wysoce wyspecjalizowanym, istnieje możliwość łatwego dołączenia odpowiednich mechanizmów pośredniczących, które umożliwią definiowanie wielu różnych modeli danych, czy też posługiwanie się różnymi językami manipulacji danych w odniesieniu do jednej fizycznej bazy danych.

● **węzeł danych** *(ang. data computer).*

Jest to koncepcja maszyny cyfrowej zarządzającej bazą danych. Maszyna tego typu może być używana jako element sieci komputerowej. W skład tej sieci mogą wchodzić zarówno procesory uniwersalne, jak i procesory specjalizowane np. szereg takich węzłów danych obsługujących wszelkie potrzeby pozostałych elementów systemu.

Węzeł danych musi być jednostką autonomiczną o bardzo wysokim poziomie funkcjonalności ze względu na małą szybkość i wysoki koszt transmisji danych w sieci.

W szczególności np. język manipulacji danych powinien być językiem samodzielnym, a nie „zanurzonym" w języku bazowym.

Dla uzyskania właściwej perspektywy możemy stwierdzić, że węzeł danych może tłumaczyć zapytania wyrażone w języku wyższego rzędu na polecenie zlokalizowane na odpowiednio niższym poziomie, które mogą być obsługiwane przez specjalizowany procesor lub inteligentną jednostkę pamięci.

ZALETY NOWEJ TECHNOLOGII

Powyżej zaprezentowano wiele różnych podejść do problemu maszyny bazy danych. Poszczególne projekty różnią się m. in. także celami, jakie sobie wyznaczyli ich autorzy. Niemniej istnieje szereg korzyści uniwersalnych, jakie można uzyskać z nowej technologii:

● **minimalizacja kosztów przetwarzania danych.**

Jak już stwierdzono wyżej, ceny takich elementów nowej architektury jak mikroprocesory, czy nawet klasyczne minikomputery, ciągle spadają, a w związku z tym przenoszenie funkcji systemu zarządzania bazą danych na te tańsze środki prowadzi do zwolnienia o wiele droższych zasobów części „klasycznej" systemu. Mogą one być o wiele lepiej wykorzystane dla wykonywania części numerycznej procesów użytkowych, co prowadzi do poprawienia globalnej wydajności systemu.

● **poprawa parametrów eksploatacyjnych systemu.**

Dzięki wprowadzeniu specjalizowanych urządzeń w miejsce uniwersalnych uzyskuje się znaczne poprawienie takich parametrów, jak czas reakcji, przepustowość czy wydajność systemu.

Istnieje możliwość usunięcia wąskich gardeł, które często pojawiają się przy rozwiązaniach klasycznych. W nowej architekturze dokonuje się analizy poszczególnych funkcji systemu i dla realizacji każdej z nich projektuje się specjalne urządzenie o takich parametrach, jakie są rzeczywiście potrzebne. Modułowość architektury daje możliwość łatwej rozbudowy, gdy system się rozrasta. System nie będzie się więc degradował użytkowo z upływem czasu.

● **poprawa niezawodności systemu.**

Niezawodność to jeden z podstawowych parametrów eksploatacyjnych, które ulegają wyraźnej poprawie. Duże i skomplikowane systemy oprogramowania mają wyraźne tendencje do zawierania dużej ilości błędów. Po wprowadzeniu systemów o specjalizowanym sprzęcie, niezawodność oprogramowania ulegnie znacznej poprawie ze względu na to, że będzie ono dużo prostsze i znacznie mniejsze. Trzeba dodać, że prace nad efektywnymi metodami weryfikacji poprawności sprzętu są znacznie bardziej zaawansowane niż odpowiednie badania dotyczące oprogramowania.

● **ochrona jakości bazy danych i ochrona dostępu do danych.**

Są to problemy, z którymi dotąd systemy zarządzania bazą danych nie bardzo umiały sobie poradzić. Wynikało to przede wszystkim stąd, że obsługa pełnej ochrony danych — ze względu na związane z nią dodatkowe obciążenie — bardzo silnie wpływała na parametry eksploatacyjne systemu. Poza tym programowa ochrona dostępu zawsze pozostawia jakieś „tylne wejścia" do danych, natomiast istniejące już rozwiązania sprzętowe problemy te rozwiązują w sposób zupełny.

* * *

Wydaje się sprawą pewną, że maszyny bazy danych zajęły trwałe miejsce wśród środków stosowanych przy rozwiązywaniu problemów bazy danych. Jednakże olbrzymie inwestycje poczynione w dziedzinie tradycyjnych systemów zarządzania bazą danych spowodują, że będzie to raczej rozwój ewolucyjny, a nie rewolucyjny. Niezwykle istotną rolę w rozwoju tej dziedziny będą odgrywały możliwości zrównoleglenia wykonywania funkcji systemowych i adresowania asocjacyjnego. Idealnym rozwiązaniem wydaje się opracowanie takiej maszyny bazy danych, która byłaby niewidoczna dla użytkownika i która mogłaby zaakceptować aktualne bazy danych i oprogramowanie użytkowe.

LITERATURA:

[1] ANSI/X3/SPARC study Group on Data Base Management Systems Interim Report 1975

[2] Hsiao D. K., Madnick S. E.: Database Machine Architecture in the Context of Information Technology Evolution. Proc. III Int. Conf. on VLDB, 1977

[3] Su S. Y. W., Lipovski G. J.: CASSM: A Cellular System for Very Large Data Bases. Proc. I Int. Conf. on VLDB, 1975

[4] Lin S. C., Smith D. C. P., Smith J. M.: The Design of A Rotating Associative Memory For Relational Database Applications. ACM TODS 1976

[5] Ozkarahan E. A., Schuster S. A., Smith K. C.: RAP — An Associative Processor for Data Base Management. Proc. NCC, 1975

[6] Cullinane J. and others: Commercial Data Management Processor Study, 1975

[7] Banerjee J., Baum R. I., Hsiao D. K.: Concepts and Capabilities of a Database Computer. ACM TODS 1979

EWA JÓŹWIAK
Centrum Projektowania i Zastosowań ZETO
Warszawa

Wybrane problemy relacyjnej bazy danych. Część 1

Projektowanie i własności języków relacyjnych

W artykule J. Pasuli pt. „Uniwersalny system zarządzania bazą danych RODAN" (INFORMATYKA nr 9/78) została przedstawiona klasyfikacja istniejących systemów zarządzania bazą danych ze względu na struktury danych obsługiwanych przez system. Tak więc wyróżnia się trzy podstawowe typy systemów zarządzania bazą danych: 1) hierachiczne, 2) sieciowe i 3) relacyjne.

Systemy relacyjne, choć nastręczają często trudności przy implementacji, są szczególnie atrakcyjne z punktu widzenia użytkownika. Relacyjny model bazy danych został przedstawiony przez E. F. Codda w 1970 r. [1]. Codd podał definicję relacji n-tego rzędu w odniesieniu do bazy danych i podkreślił korzyści tego podejścia dla niezależności danych oraz symetrii dostępu do danych. Definicje relacji, jej atrybutów, dziedzin i kluczy zostały podane w artykule B. Szymańskiego: „Główne kierunki rozwoju zastosowań relacyjnego modelu bazy danych" w czerwcowym — szóstym numerze INFORMATYKI. Niniejszy artykuł stanowi uzupełnienie przekazanych tam informacji.

Przypomnijmy sobie co się kryje pod pojęciem normalizacji w relacyjnym podejściu do bazy danych.

Relacja, której wszystkie dziedziny są proste może być reprezentowana w pamięci za pomocą tablicy dwuwymiarowej. Możemy mieć jednak relację z jedną lub kilkoma dziedzinami złożonymi, tzn. takimi, których elementami mogą być relacje. Co więcej, relacje te mogą być z kolei znowu określone na dziedzinach złożonych itd., co daje dosyć skomplikowaną strukturę. Dlatego tak istotna jest procedura eliminowania dziedzin złożonych, nazwana normalizacją (normalization). Pomysł ten został wprowadzony i opisany przez Codda [1], [2].

Teoria normalizacji oparta jest na tzw. formach normalnych — pierwszej, drugiej i trzeciej. Relacja w pierwszej formie normalnej to taka, w której żadna dziedzina nie jest listą ani relacją. Relacja w pierwszej formie normalnej może być użyta w dowolnych językach relacyjnych, które będą opisane dalej.

Zanim wprowadzimy definicję drugiej i trzeciej formy normalnej należy omówić pojęcie zależności funkcjonalnej między atrybutami oraz pojęcie atrybutu głównego i niepodstawowego.

Atrybut B relacji R jest funkcjonalnie zależny od atrybutu A relacji R jeżeli w każdej chwili czasu z każdą wartością A wiąże się w relacji R nie więcej niż jedna wartość B. Oznaczamy to następująco $A \rightarrow B$.

Podobnie zbiór atrybutów relacji R może być funkcjonalnie zależny od innego atrybutu lub zbioru atrybutów. Atrybut lub zbiór atrybutów po lewej stronie strzałki (w naszym przypadku A) zwany jest determinantą. Biorąc pod uwagę wcześniej definicję klucza relacji trzeba zauważyć, że każda relacja zawiera co najmniej jedną zależność funkcjonalną, a mianowicie: wszystkie atrybuty relacji są zależne od klucza.

Dowolny atrybut, który partycypuje w co najmniej jednym kluczu relacji R nazywa się atrybutem głównym, a wszystkie pozostałe atrybuty nazwane są niepodstawowymi.

Definicja drugiej formy normalnej brzmi więc następująco: relacja R jest w drugiej formie normalnej, jeżeli jest w pierwszej formie normalnej i każdy niepodstawowy atrybut relacji R jest zależny od każdego klucza relacji R.

Znacznie większe znaczenie ma pojęcie trzeciej formy normalnej. Relacja R jest w trzeciej formie normalnej jeżeli jest w pierwszej formie normalnej i dla każdego zbioru C atrybutów relacji R, jeśli dowolny atrybut nie należący do C jest zależny funkcjonalnie od C, to wszystkie atrybuty w relacji są funkcjonalnie zależne od C.

Sharman [3] podał inną definicję: relacja jest w trzeciej formie normalnej jeśli każda determinanta jest kluczem.

Obie te definicje wyrażają po prostu fakt, że relacja powinna opisywać proste zależności i jeżeli tak nie jest, powinna być podzielona na mniejsze relacje.

Pojęcia i zasady, na których opierają się druga i trzecia forma normalna, mogą być wytycznymi dla projektanta bazy danych, bowiem związane są ze znajomością zależności funkcjonalnych między atrybutami.

Zastosowanie relacyjnego modelu bazy danych umożliwia wprowadzenie relacyjnego podjęzyka danych. Używamy tu terminu „podjęzyk danych" (ang. *data sublanguage*) ponieważ nie zajmuje się on przetwarzaniem czy możliwością obliczania funkcji, a dotyczy raczej przechowywania i wyszukiwania danych. Termin „podjęzyk danych" oznacza tu zbiór operatorów działających na bazie danych, w połączeniu z językiem bazowym *(host language)* programowania.

Inny typ języka, tzw. język zapytań *(query language)*, to język zazwyczaj samodzielny, poprzez który użytkownik bezpośrednio oddziałuje na bazę danych. Większość języków zapytań — poza możliwościami stawiania zapytań — zapewnia także manipulowanie danymi (dopisywanie, usuwanie i poprawianie), definicję danych (tworzenie nowych relacji) oraz kontrolę danych. W porównaniu z podjęzykiem danych, język zapytań jest na ogół wyższego rzędu oraz mniej proceduralny.

Języki relacyjne można podzielić na kilka kategorii, które zostały omówione we wspomnianym artykule dr. B. Szymańskiego. Poniżej przedstawione bliżej języki ukierunkowane na wyszukiwanie wartości (ang. *mapping-oriented-languages)* [4] oraz języki ukierunkowane na graficzne przedstawienie zapytania (ang. *graphic-oriented languages).*

JĘZYKI UKIERUNKOWANE NA WYSZUKIWANIE WARTOŚCI

Języki w tej klasie są językami nieproceduralnymi przeznaczonymi dla użytkowników, którzy muszą operować na dużej bazie danych, ale nie są zawodowymi programistami. Języki te mają ułatwić wyrażenie prostych operacji w celu otrzymania informacji z tablic. Zaliczamy do nich języki SQUARE [9] oraz SEQUEL [7, 8].

SEQUEL (Structured Englisch QUERY Language) jest relacyjnym, nieproceduralnym językiem zapytań, nie zawierającym kwantyfikatorów ani żadnych pojęć matematycznych. Używa on blokowych formatów strukturalnych z angielskich słów kluczowych.

Własności zapytań w języku SEQUEL pokażemy przez podanie przykładów opartych na bazie danych opisującej małą instytucję. Baza danych zawiera dwie tablice:

1) ZATRUDNIENIE (NRZAT, NAZWISKO, NRZAK, KIER, TYTUŁ, PŁACA)
2) ZAKŁAD (NRZAK, NAZWAZAK, MIEJSCOWOSC).

Tablica ZATRUDNIENIE podaje numer każdego zatrudnionego, jego nazwisko, numer zakładu, w którym pracuje, tytuł służbowy oraz płacę.

Tablica ZAKŁAD zawiera numer zakładu, jego nazwę i miejscowość, w której się znajduje.

Podstawowa operacja w języku SEQUEL nazywa się „odwzorowanie" *(mapping)* i odpowiada czynności znajdywania w tablicy wartości, która jest związana z jakąś inną znaną wartością. Operację tę pokazuje przykład 1.

Przykład 1. Znaleźć numery zakładów, które są zlokalizowane w Krakowie.

```
SELECT NRZAK
FROM ZAKŁAD
WHERE MIEJSCOWOSC = „KRAKOW";
```

Operacja „odwzorowanie" zawiera tu trzy słowa kluczowe: SELECT, FROM i WHERE oraz trzy parametry: tablica, do której zapytanie jest skierowane (ZAKŁAD), nazwy pozycji, które mają być podane (NRZAK), oraz warunek, jaki ma być spełniony (MIEJSCOWOSC = KRAKOW"). Ten prosty format zwany jest blokiem zapytania i używany jest w całym języku SEQUEL.

SEQUEL dopuszcza rozszerzenia tego bazowego schematu. Zapytanie może określać więcej niż jedną pozycję do wybrania. Może też zawierać złożony warunek boolowski w klauzuli WHERE, co pokazuje przykład 2.

Przykład 2. Wypisać nazwiska i kierowników wszystkich zatrudnionych w zakładzie nr 9, którzy zarabiają więcej niż 3000

```
SELECT NAZWISKO, KIER
FROM ZATRUDNIENIE
WHERE NRZAK = 9
WHERE PŁACA > 3000;
```

Jeśli klauzula WHERE zostanie opuszczona w bloku zapytania, dostaniemy wszystkie wartości wybranej kolumny tablicy. Czynność ta odpowiada operatorowi rzutowania u Codda.

Kiedy użytkownik chce wybrać cały wiersz, w którym jest spełniony warunek WHERE, może użyć skrótu SELECT *. Ta własność odpowiada operatorowi ograniczenia *(restriction)* (przykład 3).

Przykład 3. Wypisać wszystkie dane dla zatrudnionych, którzy zarabiają więcej niż 3000

```
SELECT *
FROM ZATRUDNIENIE
WHERE PŁACA > 3000;
```

SEQUEL dopuszcza w bloku zapytania specyfikację pewnych funkcji w klauzuli SELECT. Do funkcji tych należą: SUM, COUNT, AVG, MAX, MIN (przykład 4).

Przykład 4. Znaleźć przeciętną płacę zatrudnionych w zakładzie nr 9

```
SELECT AVG (SAL)
FROM ZATRUDNIENIE
WHERE NRZAK = 9;
```

Warunek WHERE może być użyty albo do porównania wartości ze stałą lub w celu sprawdzenia przynależności do zbioru. Zawieranie w zbiorze określa słowo IN.

Badanie zawierania w zbiorze w klauzuli WHERE pozwala umieszczać jeden blok zapytania w drugim aż do wielu poziomów, co stwarza możliwości wyrażania dosyć skomplikowanych zapytań (przykład 5).

Przykład 5. Wypisać nazwiska zatrudnionych, którzy pracują w zakładach mieszczących się we Wrocławiu

```
SELECT NAZWISKO
FROM ZATRUDNIENIE
WHERE NRZAK IN
    SELECT NRZAK
    FROM ZAKŁAD
    WHERE MIEJSCOWOSC = „WROCŁAW";
```

W konstruowaniu zapytań w języku SEQUEL konieczne jest wskazanie związków między dwoma blokami zapytania. W tym celu SEQUEL dopuszcza zmienną korelacji, którą umieszcza się w klauzuli FROM. Zmienna korelacji reprezentuje wiersz wskazanej tablicy i może być użyta w innych blokach zapytania, aby odwołać się do tego wiersza.

Pewne zapytania żądają odpowiedzi, której nie da się otrzymać przez prostą selekcję wartości z tablicy lub użycie wspomnianych wyżej funkcji. Można wówczas używać zmiennej zwanej „zmienna obliczana" (ang. *computed variable*) w klauzuli SELECT. Dla zdefiniowania zmiennej obliczanej używa się specjalnej klauzuli COMPUTE. Zmienna obliczana wraz ze zmienną korelacji oraz wspomnianymi funkcjami dają szerokie możliwości stawiania zapytań.

JĘZYKI UKIERUNKOWANE NA GRAFICZNE PRZEDSTAWIENIE ZAPYTANIA

W przypadku języków tego typu użytkownik nie stawia zapytania w sposób liniowy przy użyciu określonej składni, lecz wypełnia odpowiednie miejsca w tablicy wyświetlanej na ekranie, podając w ten sposób przykład operacji do wykonania. Do tej klasy języków należą Query-by-Example [7, 8] oraz CUPID [10].

W przypadku języka Query-by-Example użytkownik wyświetla na ekranie szkielet tablicy odpowiadający relacji, którą będzie się zajmował. Następnie wypełnia odpowiednie wiersze tablicy przykładem oczekiwanej odpowiedzi. Elementy podkreślone, tzw. elementy przykładowe stanowią przykład możliwej odpowiedzi, natomiast wartości znane, nie podkreślone, tzw. elementy stałe, reprezentują warunek wyrażony w zapytaniu (poleceniu). Operator „P." oznacza żądanie wyświetlenia lub wydrukowania odpowiedzi.

Tak więc np. wypełnienie tablicy ZAKŁAD (NRZAK, NAZWAZAK, MIEJSCOWOSC) w sposób następujący: od-

ZAKŁAD	NRZAK	NAZWAZAK	MIEJSCOWOSC
	P. X		KRAKOW

powiada poleceniu z przykładu 1, a mianowicie: „Znaleźć numery zakładów, które są zlokalizowane w Krakowie".

Query-by-Example dopuszcza przeszkiwanie danych z ich porządkowaniem przy użyciu specjalnych operatorów „AO." oraz „DO.".

Jeżeli do sformułowania zapytania potrzebne są dwie lub więcej tablic, generuje się ich szkielety przez użycie specjalnego klucza. Element przykładowy może wiązać dwa różne wiersze tej samej tablicy (przykład 6) lub dwie różne tablice (przykład 7).

W przykładach tych będziemy się posługiwać podaną poprzednio bazą danych, składającą się z dwóch tablic: ZATRUDNIENIE oraz ZAKŁAD.

ZATRUDNIENIE	NRZAT	NAZWISKO	NRZAK	KIER	TYTUŁ	PŁACA
		P. KOWALSKI ZYCH		ZYCH		>X X

Przykład 6. Znaleźć pracowników, którzy zarabiają więcej niż ich kierownicy.

ZATRUDNIENIE	NRZAT	NAZWISKO	NRZAK	KIER	TYTUŁ	PŁACA
		P. KOWALSKI	X			

ZAKŁAD	NRZAK	NAZWAZK	MIEJSCOWOSC
	X		WARSZAWA

Przykład 7. Podać nazwiska pracowników zatrudnionych w zakładach zlokalizowanych w Warszawie

Język zawiera dodatkowe operacje:
- porównania numeryczne: =, #, >, ⩾, <, ⩽
- operator negacji: ¬
- operatory: JOIN

ALL (każdy możliwy), np. ALL.KOWALSKI oznacza listę wszystkich nazwisk występujących w tablicy
UN.ALL (tworzy zbiór i eliminuje elementy powtarzające się)
- funkcje: SUM, COUNT, AVG, MAX, MIN

Język ten został przedstawiony w artykule „Język Query-by-Example" w numerze 10/78 INFORMATYKI.

Relacyjny model zarządzania bazą danych ma wiele zalet, spośród których najważniejsze, to:
- prostota — użytkownik operuje na prostej strukturze logicznej
- niezależność danych — terminem tym określa się niezależność między informacyjną zawartością danych, a szczegółami ich reprezentacji. Dzięki tej niezależności zmiana reprezentacji pewnych danych nie musi pociągać za sobą zmiany postępowania użytkownika
- symetria dostępu do danych
- mocna podstawa teoretyczna — relacyjny model bazy danych opiera się na dobrze rozwiniętej matematycznej teorii relacji oraz rachunku predykatów rzędu pierwszego.

Nie bez znaczenia jest fakt, że relacyjny model bazy danych umożliwia definiowanie nieproceduralnych języków relacyjnych wysokiego rzędu, łatwych do przyswojenia i wygodnych w użyciu, dzięki czemu użytkownikami mogą być pracownicy nie będący programistami. Nieproceduralne podejście do projektowania języków relacyjnych zapewnia jednolity sposób manipulowania danymi, ich definicji i kontroli, co omówimy w kolejnym artykule.

LITERATURA:

[1] Codd E. F.: A relational model of data for large shared data banks. Communications of the ACM 1970, Volume 13, No 6, str. 377—397

[2] Codd E. F.: Further Normalization of the Data Base Relational Model, Curent Computer Science Symposium, 6 maja 1971, Data Base Systems

[3] Sharman G. C. H.: A new model of relational data base and high level languages. Technical Report TR. 12.136. IBM Hursley Park Laboratory, England, luty 1975

[4] Codd E. F.: A data base sublanguage founded on the relational calculus. 1971. ACM Sigfidet Workshop on Data Description, Access and Control. ACM, New York, 1971, str. 35—68

[5] Chamberlin D. D., Boyce R. F.: SEQUEL: A Structured English Query Language. Proc. ACM — Sigfidet Workshop on Data Deseription, Access and Control, maj 1974, ACM, New York, str. 249—264

[6] Astrahan M. M., Chamberlin D. D.: Implementation of a Structured English Query Language. Communications of the ACM, październik 1975, Volume 18, No 10, str. 580—588

[7] Zloof M. M.: Query-by-Example. AFPIS Conference Proceedings, National Computer conference 44, str. 431—438, 1975

[8] Zloof M. M.: Query-by-Example — a Data Base Language. IBM Systems Journal, Volume 16, No. 4, str. 324—343, 1977

[9] Boyce R. F., Chamberlin D. D.: Specifying queries as relational expressions: SQUARE. Data Base Management, Proc. IFIP Working Conf., kwiecień 1974, North-Holland Publ. Co., Amsterdam, 1974, str. 169—177

[10] McDonald N., Stonebraker M.: CUPID: the fiendly query language Proc. ACM Pacfic 75 Regional Conf., kwiecień 1975 r., ACM, New York, 1975, str. 127—131

Wiedeński kongres ADV

W dniach 17—21 marca 1980 r. odbędzie się w Wiedniu szósty międzynarodowy kongres informatyczny, organizowany co trzy lata przez austriackie stowarzyszenie informatyków (Arbeitsgemeinschaft für Datenverarbeitung).

Podobnie jak poprzednie kongresy, a także większość podobnych imprez międzynarodowych, przyszłoroczny kongres będzie się odbywał pod hasłem „Szanse i granice przetwarzania informacji" („Chancen und Grenzen der Informationsverarbeitung"). Zamierzeniem organizatorów jest bowiem nie tylko zaprezentowanie najbardziej interesujących osiągnięć w dziedzinie zastosowań komputerów, ale również pokazanie perspektywicznych możliwości tych zastosowań oraz istniejących nadal ograniczeń.

Coraz szerszy zasięg zastosowań informatyki zmusza każdego użytkownika i specjalistę-informatyka do zetknięcia się również z konsekwencjami tego rozwoju oraz zrewidowania wielu dotychczasowych poglądów. Dlatego głównym celem kongresu jest przedstawienie tych problemów zarówno w referatach i dyskusjach, jak i projektowanych demonstracjach systemów, ze szczególnym podkreśleniem zaobserwowanych trendów rozwojowych.

Wspomniany temat przewodni będzie się przewijał w trzech podstawowych obszarach tematycznych kongresu:
1) systemy organizacyjne w wymianie informacji (teoria i praktyka, problemy oczekujące rozwiązań)
2) technologia przetwarzania informacji (sprzęt, oprogramowanie, metody, kompatybilność, sieci komputerowe)
3) aspekty społeczno-polityczne oraz humanitarne zastosowań (wolność czy nowe rodzaje przymusu).

Podobnie jak ostatni kongres (marzec 1977 r.), kongres ADV 1980 odbędzie się w wiedeńskim hotelu HILTON.

(W.K.)

Na podstawie czasopisma ONLINE nr 4/1979

Bibliografia wydawnictw polskich z dziedziny informatyki

Język programowania PLAN 3 — DRAŻEK Z., JURKOWSKI J. WNT, Warszawa 1978, s. 324, cena 65 zł.

Ogólne wiadomości z zakresu arytmetyki dwójkowej. Nośniki informacji. Pamięć operacyjna maszyny cyfrowej. Struktura programu. Instrukcje programowe. Ogólne wiadomości o technice programowania. Programowanie urządzeń zewnętrznych. Podprogramy standardowe. Pozostałe dyrektywy i ich zastosowanie. Przykłady podprogramów segmentacji i nakładania programów. Ogólne wiadomości o kompilacji programów. Wykaz błędów kompilacji oraz błędów logicznych programu. Zestawienie instrukcji według kodu wewnętrznego.
Książka przeznaczona jest dla programistów oraz studentów kierunków informatycznych wyższych uczelni.

Przetwarzanie sygnałów metodami analogowymi i cyfrowymi — BEAUCHAMP K. G. Tłum. wyd. ang. z 1973 r. WNT, Warszawa 1978, s. 449, cena 120 zł.

Zbieranie i przechowywanie danych. Przetwarzanie danych. Przetwarzanie wstępne. Procesy przetwarzania analogowo-cyfrowego. Metody statystyczne Szeregi Fouriera i analiza częstotliwościowa. Filtracja cyfrowa. Analiza widmowa. Analiza korelacyjna. Analiza stanu nieustalonego i widmo udarowe. Procesy niestacjonarne.
Książka zawierająca teoretyczne podstawy przetwarzania sygnałów przeznaczona jest dla inżynierów różnych specjalności zajmujących się przetwarzaniem danych pomiarowych oraz specjalistów aparatury pomiarowej.

Metodologia programowania — Turski W. M. WNT, Warszawa 1978, s. 231, cena 80 zł.

Podstawowe konstrukcje występujące w programowaniu. Współpraca modułów. Projektowanie programów.
Książka przeznaczona jest dla programistów.

Systemy cyfrowe wieloprocesorowe — ENSLOW P. H. — red. Tłum. wyd. ang. z 1974 r. WNT, Warszawa 1978, s. 360, cena 78 zł.

Motywy rozwoju systemów wieloprocesorowych i przetwarzania równoległego Sprzęt systemów wieloprocesorowych. Systemy operacyjne i inne oprogramowanie podstawowe dla wieloprocesorów. Teraźniejszość i przyszłość. Słownik. Dodatki (szczegółowe opisy 17 istniejących systemów wieloprocesorowych).
Książka przeznaczona jest dla programistów, projektantów i użytkowników systemów komputerowych.

Umiejętność programowania — DIJKSTRA E. W. Tłum. wyd. ang. z 1976 r. WNT, Warszawa 1978, s. 221, cena 60 zł.

Sprawcza abstrakcja. Rola języków programowania. Stany i ich charakteryzowanie. Charakteryzowanie semantyki. Semantyczne charakteryzowanie języka programowania. Dwa twierdzenia. O budowie konstrukcji kończących się pomyślnie. Raz jeszcze o algorytmie Euklidesa. Formalna analiza kilku drobnych przykładów. O ograniczaniu niedeterminizmu. Esej o pojęciu „zasięgu zmiennych". Zmienne tablicowe. Twierdzenie o Liniowym Przeszukiwaniu. Zadanie o kolejnej permutacji. Zadanie o holenderskiej fladze narodowej. Aktualizacja kartoteki sekwencyjnej. Jeszcze o problemach łączenia. Zadanie przypisywane R. W. Hammingowi. Zadanie o powtórzeniach wzorca. Przedstawienie liczby w postaci sumy dwu kwadratów. Zadanie o najmniejszym czynniku pierwszym dużej liczby. Zadanie o najbardziej izolowanych wsiach. Zadanie o najkrótszym drzewie powiązań. Algorytm Rema zapisywania klas równoważności. Zadanie o trójwymiarowej powłoce. Znajdowanie maksymalnej silnej składowej grafu. O podręcznikach i realizacjach. Rzut oka wstecz.
Książka przeznaczona jest dla programistów i projektantów SEPD.

Modelowanie symulacyjne systemów — KONDRATOWICZ L. WNT, Warszawa 1978, s. 322, cena 90 zł.

Cz. 1. Podstawy metodologii projektowania symulatora: Od systemu oryginalnego do systemu odwzorowującego. Struktura danych modelu symulującego. Struktura funkcjonalna modelu symulacyjnego. Synteza modelu symulacyjnego systemu. Podstawowe właściwości strukturalne modeli symulacyjnych systemów złożonych. Symulator. Przegląd środków symulacji cyfrowej. Cz. 2. Oprogramowanie modeli symulacyjnych w języku CSZ: Wprowadzenie do języka CSL. Reprezentacja struktury danych modelu. Organizacja procesu symulacyjnego. Instrukcja sterującego. Badanie stanu zbioru. Działania na zbiorach. Uwarunkowany wybór elementu zbioru. Losowanie i gromadzenie danych. System wejścia/wyjścia. Systemy operacyjne języka CSL. Cz. 3. Przykładowe modele symulacyjne: Modele symulacyjne typowych systemów obsługi. Symulacja awaryjności urządzenia produkcyjnego. Symulacja bazy przeładunkowo-składowej. Model symulacyjny centrali telefonicznej. Problemy zapasów magazynowych. Symulator układu automatycznej regulacji. Abstrakcyjny model symulacyjny systemu.

Książka przeznaczona jest dla pracowników naukowych oraz praktyków-inżynierów, ekonomistów, analityków systemów zajmujących się teorią i zastosowaniem symulacji cyfrowej.

Praktyka programowania — VAN TASSEL D. Tłum. wyd. ang. z 1974 r. WNT, Warszawa 1978, s. 230, cena 50 zł.

Styl programowania. Projektowanie programu. Sprawność programu. Uruchamianie programu. Testowanie programu. 101 zadań do programowania. Dodatki. Skracanie programów FETE: a FORTRAN Execvtion Time Estimator. Optymalizacja operacji taśmowych.

Książka przeznaczona jest dla programistów, którzy chcą osiągnąć biegłość w programowaniu. Stanowi ona próbę podsumowania dotychczasowych doświadczeń z tej dziedziny. Autor podaje zbiór zaleceń, których powinien przestrzegać programista.

Metody wyszukiwania i klasyfikacji informacji — DĄBROWSKI M., LAUS-MĄCZYŃSKA K. WNT, Warszawa 1978, s. 133, cena 50 zł.

Indeksowanie. Automatyczna klasyfikacja. Metody organizacji i wyszukiwania dokumentów.

Materiały przeznaczone są dla projektantów systemów przetwarzania informacji, programistów oraz pracowników informacji naukowej, technicznej i ekonomicznej.

System automatyzacji projektowania procesów technologicznych — CWIETKOW W. D. Tłum. wyd. ros. z 1972 r. PWN, Warszawa 1978, s. 274, cena 48 zł. Seria: Informacja i sterowanie.

Podstawowe założenia systemo-strukturalnej analizy procesów technologicznych. Sformalizowany opis informacji o częściach maszyn. Systemowo-strukturalna analiza procesu technologicznego. Sformalizowany opis technologicznych charakterystyk przedsiębiorstwa. Wielopoziomowa iteracyjna metoda projektowania procesów technologicznych. Metody i algorytmy projektowania przebiegów procesów technologicznych. Metody i algorytmy projektowania operacji technologicznych. Metody i algorytmy projektowania zabiegu. Budowa zautomatyzowanego systemu projektowania procesów technologicznych. Wybór najbardziej racjonalnego wariantu systemu automatyzacji projektowania procesów technologicznych.
Książka przeznaczona dla inżynierów mechaników-technologów, ma przede wszystkim wartość metodologiczną. Jest to pierwsza monografia w polskiej literaturze technicznej, która stosunkowo dokładnie omawia podstawowe koncepcje automatycznego projektowania technologii wytwarzania części maszyn.

Organizacja przetwarzania danych — BUCHTA D., MESSNER Z. PWE, Warszawa 1978, s. 254, cena 28 zł.

Charakterystyka procesu przetwarzania danych. Środki techniczne mechanizujące proces przetwarzania danych. Rozwój i kierunki zastosowań sprzętu informatycznego. Programowanie komputerów. Programowanie komputerów — przykłady języków programowania. Podstawowe zagadnienia projektowania systemów informatycznych. Proces wdrażania systemów informatycznych. Systemy informatyczne dla potrzeb zarządzania jednostkami gospodarczymi. Ośrodki obliczeniowe.

Podręcznik zapoznaje przyszłą kadrę ekonomistów z podstawowymi zagadnieniami komputeryzacji systemów informacyjnych.

Oprac. (A.K.)

Łamy INFORMATYKI otwarte dla wszystkich!

Zanim jednak nasi Autorzy sięgną po pióro, prosimy, by zechcieli zapoznać się z poniższymi informacjami.

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

W artykułach można omawiać, prezentować lub proponować wszystko, co dotyczy współczesnej informatyki, oraz wszystko, co wiąże się z jej kierunkami rozwoju — zarówno z pozycji informatyka, jak i użytkownika informatyki.

Materiał, oprócz tekstu zasadniczego, powinien zawierać — na oddzielnych stronach — kartę tytułową (strona 1), krótki życiorys zawodowy autora (strona 2) i jego zdjęcie, wykaz literatury, tabele, rysunki, podpisy pod rysunki, zdjęcia.

Na stronie 1 należy podać tytuł naukowy, imię i nazwisko, nazwę zakładu pracy, adres prywatny i telefon, tytuł artykułu oraz informację, jaką drogą przesłać honorarium po opublikowaniu artykułu: kasa Wydawnictwa, poczta, bank (w takim przypadku prosimy podać numer konta PKO).

Konstrukcja artykułu powinno być zwarta i przejrzysta; wstęp musi wprowadzić czytelnika w zagadnienie, w podsumowaniu należy sformułować wnioski; podział na rozdziały, podrozdziały i akapity powinien być logiczny i konsekwentny. Należy zwrócić szczególną uwagę na poprawność stylistyczną i terminologiczną, unikać skrótów, rzadko stosowanych wyrażeń obcych i żargonu fachowego; starannie definiować nowe terminy. Należy również wystrzegać się nieczytelnych i zbyt rozbudowanych wzorów.

Tekst powinien być napisany na maszynie, jednostronnie, na papierze nieprzebitkowym formatu A-4, z marginesem 5 cm (30 wierszy na 1 stronie, 60 znaków w 1 wierszu).

Wykaz literatury powinien zawierać: kolejny numer pozycji (w nawiasie kwadratowym), nazwisko i imię autora, tytuł publikacji (książki lub artykułu), ewentualnie tytuł i numer czasopisma (w przypadku artykułu), miejsce i rok wydania.

Tabele — każda na oddzielnej stronie — powinny być numerowane i opatrzone tytułem oraz ściśle związane z tekstem (odniesienie na marginesie).

Rysunki — każdy oddzielnie (uwaga: nie wklejać rysunków w tekst!) — powinny być czytelne i również ściśle związane z tekstem (odniesienie na marginesie). Format rysunku nie może być mniejszy niż 10 × 10 cm.

Podpisy pod rysunkami, napisane również na oddzielnej stronie, oprócz kolejnego numeru powinny zawierać tytuł rysunku i ewentualnie legendę dotyczącą poszczególnych elementów.

Łączna objętość materiału nie powinna przekraczać w przypadku

- artykułu problemowego — 12 stron
- reportażu — 8 stron
- recenzji, relacji z imprezy — 6 stron
- informacji — 4 stron maszynopisu

Tak przygotowany materiał prosimy dostarczyć w dwóch egzemplarzach pod adresem: redakcja INFORMATYKI, ul. Jasna 14/16, 00-041 Warszawa. Wszelkich dodatkowych informacji udzielamy pod telefonem 27-71-40.

Autor opublikowanego w INFORMATYCE artykułu otrzymuje bezpłatnie egzemplarz okazowy.

Materiałów nie zakwalifikowanych do druku redakcja nie zwraca.