1 1974

prace

Instytutu Maszyn Matematycznych

rok XVI

E r r a t a

jest	powinno być
str. 78	str. 80
str. 79	str. 78
str. 80	str. 79

Redakcja przeprasza Autora
i Czytelników za ten błąd.

1 1974

p r a c e
I n s t y t u t u
M a s z y n
M a t e m a t y c z n y c h

Rok XVI

Warszawa 1974

Adres Redakcji: Instytut Maszyn Matematyoznych
Branżowy Ośrodek INTE
Warszawa, ul. Krzywickiego 34
tel. 28-37-29

Druk IMM z. 8/75 n. 500 GP-II-1435/68

OD REDAKCJI

Informujemy naszych czytelników, że w roku 1973 ukazał się jedynie zeszyt nr 1/73 "PRAC Instytutu Maszyn Matematycznych". W roku 1974 ukażą się dwa zeszyty "PRAC IMM".

SPIS TREŚCI ZESZYTU 1

СОДЕРЖАНИЕ

CONTENTS

Prace IMM
Zeszyt 1

681.327.2.04.001

WYBRANE ZAGADNIENIA ZWIĄZANE ZE STRUKTURAMI DANYCH

Danuta KOŁACKA
Anna WIECZOREK
Jan WIERZBOWSKI
Julian WINIEWSKI

Pracę złożono 16.01.1974

W pracy podjęto próbę wyboru i określenia podstawowych pojęć z zakresu teorii struktur danych oraz opisania związków występujących między tymi pojęciami. Poruszono następujące problemy: struktury danych i struktury zapisu, organizacja pliku i zbioru danych, operowanie danymi i metody dostępu, zagadnienia związane z bankiem danych, przytoczono próby formalizacji tych pojęć. Opracowanie zamyka spis podstawowych terminów oraz obszerna bibliografia.

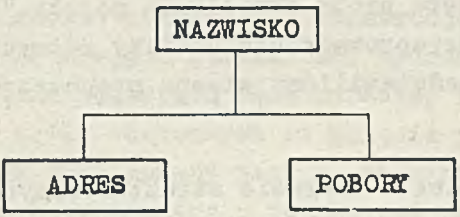
S p i s t r e ś c i

1. WSTĘP

Niniejsze opracowanie stanowi próbę wyboru i określenia podstawowych pojęć z zakresu teorii struktur danych oraz opisania związków występujących między tymi pojęciami. Literatura dotycząca tego tematu jest bardzo bogata, jednakże sama teoria jest tak młoda, że dotychczas brak jest jednoznacznej terminologii (nie tylko zresztą polskiej), jak również jednoznacznej interpretacji i klasyfikacji różnych pojęć.

W praktyce bardzo często przedstawia się dane graficznie, np. tak jak pokazano na rys. 1 i 2. Interpretacja tych rysun-

ków staje się jasna dopiero wówczas, gdy rozróżni się pojęcie dokumentu - tj. obiektu, którym operuje użytkownik - od pojęcia zapisu, tj. obiektu, który jest przechowywany w pamięci maszyny. Za pomocą tych pojęć można np. następująco zinterpretować omawiane rysunki:



Rys. 1

Rys. 2

Dokument (rys. 1) jest złożony z trzech pól o nazwach NAZWISKO, ADRES i POBORY. W pamięci maszyny dokument jest przechowywany w postaci zapisu złożonego z pól o tej samej nazwie (rys. 2). Pierwszy rysunek pokazuje jak użytkownik "widzi" dane, drugi - jak są one zapisane w maszynie.

Powyższe uwagi wskazują na potrzebę rozróżniania niektórych bliskoznacznych pojęć, takich jak zbiór - plik, dokument - zapis, struktura zapisu - struktura danych itp. Rozwiniętą w dalszym ciągu opracowania (rozdziały 2 i 3) próbę określenia tych pojęć można schematycznie przedstawić następująco:

odpowiadające sobie pojęcia z punktu widzenia:	
użytkownika	zapisu w komputerze
struktura (organizacja) danych	struktura (organizacja) zapisu
plik	zbiór
dokument	zapis
pozycja	pole
operowanie danymi	metody dostępu

W końcowym okresie opracowywania niniejszej publikacji ukazało się tłumaczenie zeszytu Diebolda pt. "Programy operowania danymi" [53]; w obu opracowaniach stosowana jest podobna terminologia.

W rozdziale 4 podjęto próbę określenia pojęć: "bank danych" i "baza danych". Po przeprowadzeniu analizy różnych interpretacji tych pojęć, przedstawiliśmy własne propozycje na ten temat.

Do podstawowych dzieł w zakresie struktur danych należy zaliczyć przede wszystkim prace Berztissa [15], Floresa [60], Knutha [94] i Turskiego [146]. Prace te dobrze ilustrują różne sposoby podejścia do teorii struktur danych: Knuth podaje systematyczny i ścisły (ale nie sformalizowany) opis różnych zagadnień, podczas gdy u Turskiego znajdujemy przede wszystkim (choć nie tylko) formalizację. Niektórzy autorzy próbują stosować aparat matematyczny do rozwiązywania niektórych zagadnień, bądź też za pomocą znanych pojęć matematyki interpretować pojęcia teorii struktur danych. Tego typu zagadnienia zostały ogólnie omówione w rozdziale 5.

Jak już wspomniano bibliografia omawianego zagadnienia jest bardzo obszerna. Autorom niniejszego opracowania nie udało się dotrzeć do wszystkich pozycji, jednakże uznaliśmy, że wszystkie ważne pozycje powinny się znaleźć w wykazie, nawet jeśli nie są one dostępne. Bibliografia nie obejmuje natomiast publikacji firmowych dotyczących zagadnień zarządzania danymi.

W pracach, które doprowadziły do niniejszej publikacji brał również udział Marek Cichy, który w szczególności opracował punkty 4.4, 4.6 i 5.5, za co autorzy składają mu podziękowanie.

2. STRUKTURY DANYCH I STRUKTURY ZAPISU

Program jest opisem określającym jakie działania i na czym należy wykonać. Opis ten musi spełniać kanony języka, w któ-

rym został napisany, a język musi być oczywiście zrozumiały dla maszyny. W języku bezpośrednio zrozumiałym dla maszyny można zapisać operacje z pewnego zbioru. Argumentami tych operacji są słowa (rejestry) pamięci maszyny. Jednakże w ogromnej większości zastosowań posługiwanie się słowami pamięci jest uciążliwe. Dlatego też konstruuje się bardziej złożone, ale wygodniejsze języki tzw. wyższego szczebla. Języki te operują na obiektach bądź bardziej złożonych od słów pamięci bądź w ogóle oderwanych od pojęcia pamięci maszyny. Każdy nowy język realizowany jest na maszynie za pomocą programu translatora napisanego w języku już istniejącym, w efekcie każdy tekst języka wyższego szczebla zostaje sprowadzony do tekstu w języku maszyny.

Obiekty, na których działa się w językach wyższego szczebla zwane są strukturami danych tego języka. Sposób przedstawienia danych za pomocą obiektów "rzeczywistych" dla maszyny, tj. takich jak np. bit, słowo, rejestr itp. nazywa się strukturą zapisu. Sposób wykonania operacji na strukturach danych zależy oczywiście od wybranej struktury ich zapisu.

Należy podkreślić, że struktury danych i ich implementacja czyli struktura zapisu są całkiem niezależnymi pojęciami. Konkretną strukturę danych można zimplementować na wiele różnych sposobów, a ta sama struktura zapisu może być wykorzystana do reprezentowania wielu różnych struktur danych.

2.1. Dane, struktury, relacje porządkujące

W każdym języku programowania występuje pewien zbiór wartości prostych (strukturalnie niepodzielnych), zwanych na ogół stałymi, np. liczby, stałe logiczne, napisy itp. Daną nazywa się parę (nazwa, wartość), w której nazwa odpowiada konwencjom danego języka, a wartością może być stała tego języka, inna dana lub zbiór stałych i/lub danych.

Jeśli wartością jest stała, to daną taką nazywa się daną prostą, a gdy inna dana - to mówi się o hierarchicznej struk-

turze nazw. Trzeci przypadek jest najbardziej ogólny. Można w nim wyróżnić dwie możliwości: zbiór będący wartością jest uszeregowany lub nie. Strukturami nieuszeregowanymi są np. pliki o organizacji bezpośredniej lub grupowej. Struktury uszeregowane to takie, w których zbiór będący wartością jest quasi-uporządkowany, tzn. jest w nim określona relacja zwrotna i przechodnia. Quasi-uporządkowane są np. pierścienie (lista, w której po ostatnim elemencie następuje znowu pierwszy) lub ogólniej sieci (grafy). Pewne struktury są uszeregowane relacją mocniej niż quasi-porządkującą. Np. strukturami o porządku częściowym (quasi-porządek i antysymetria) są drzewa, czy ogólniej sieci (grafy) bez cykli, a o porządku liniowym (porządek częściowy i spójność) np. listy. Relacje quasi, częściowo i liniowo porządkujące będą dalej nazywane relacjami porządkującymi.

Relacje porządkujące strukturę są jednak pojęciem zbyt oderwanym od maszyny, by dały się w niej łatwo zrealizować, bowiem dla każdego elementu struktury konieczne byłoby badanie, które elementy struktury są od niego wcześniejsze, a które późniejsze, lub które nie są z nim w relacji. Zamiast tego można skorzystać z własności przechodniości relacji porządkującej i rozpatrywać te tylko elementy, które bezpośrednio poprzedzają lub następują po danym elemencie. Przez związki między elementami sąsiednimi można prosto opisać strukturę. Np. drzewo to taka struktura, w której tylko jeden element nie ma poprzednika, a każdy z pozostałych ma tylko po jednym. W liście tylko jeden element nie ma poprzednika i tylko jeden następnika (początek i koniec listy), a każdy pozostały po jednym poprzedniku i po jednym następniku; ponadto wychodząc z początku listy można dotrzeć do każdego jej elementu. Lista dwukierunkowa jest to struktura różniąca się tym od listy, że określono w niej drugą relację porządkującą liniowo; poprzednik każdego elementu według jednej relacji jest jednocześnie jego następnikiem według drugiej.

Pierścień jest strukturą podobną do listy z tym, że poprzednikiem początku jest koniec listy. Analogiczna różni-

ca występuje między pierścieniem dwukierunkowym a listą dwukierunkową.

Wartościami elementów struktur mogą być również struktury. Bardzo popularną i często spotykaną w językach programowania strukturą tego typu jest tablica. Tablica jest listą, w której elementami są listy o równej długości. W tego typu strukturach mogą występować nieregularności nazw. Np. dwie różne podstruktury tej samej struktury mogą posiadać jeden lub więcej wspólnych elementów. Zwane one bywają listami przewleczonymi.

Struktura zapisu struktur uporządkowanych polega na umieszczeniu w pamięci maszyny elementów struktury oraz na zaznaczeniu przy każdym z nich, które elementy są jego następnikami. Konkretna realizacja zależy częściowo od struktury danych, jak również od rodzaju nośnika danych. Najczęściej stosowanymi metodami zapisu jest metoda dowiązywana (łańcuchowa) i sekwencyjna (seryjna). Dla przykładu listę można zapisać pierwszą z tych metod w sposób następujący: każdemu z elementów listy przydziela się pewne miejsce w pamięci, które dzieli się na 2 części. W pierwszej z nich znajduje się wartość elementu, a w drugiej tzw. "dowiązanie", czyli adres miejsca w pamięci zawierającego następny element listy. Jeśli wartością elementu jest struktura (nazwana wtedy podstrukturą), to w pierwszej części miejsca pamięci zapisany jest adres pierwszego elementu tej podstruktury.

Druga metoda polega na tym, że elementy są umieszczane jeden po drugim w pamięci w sposób ciągły. Element następny na liście zajmuje następne miejsce w pamięci. Jak więc widać relacja porządkująca zbiór jest w swej realizacji na maszynę cyfrową metodą dostępu do elementu: mając jakiś element struktury można pobrać jeden z elementów bezpośrednio po nim następujących. Metoda pobierania elementów wyznaczona przez strukturę danych jest główną metodą dostępu ale nie jedyną. Inne określone mogą być przez strukturę zapisu lub nośnik danych. Np. w pamięci szybkiej maszyny, ze struktury sekwencyjnej moż-

na pobrać element bezpośrednio, podając po prostu adres przydzielonego mu miejsca pamięci, pomijając natomiast elementy poprzednie tej struktury.

2.2. Plik i jego organizacja

Istnieje pewna klasa języków lub "dobudówek" do języków przeznaczonych do przetwarzania danych, zwanych również systemami zarządzania bankiem danych (GDBMS). Występujące w nim struktury danych różnią się od poprzednio omawianych. Podstawową strukturę nazywa się tradycyjnie plikiem, ale ostatnio coraz powszechniej używa się terminu "zbiór danych".

Plik jest to zbiór elementów zwanych dokumentami. Dokument jest również strukturą. Jest on ciągiem (porządek liniowy) pozycji i/lub grup. Pozycja elementarna jest daną prostą, a grupa (inaczej zwana pozycją złożoną), podobnie jak dokument, ciągiem pozycji i/lub grup. Grupa jest więc strukturą umożliwiającą tworzenie hierarchii nazw w dokumencie. Ze względu na postać dokumentu pliki dzieli się na trzy rodzaje: formatowe, półformatowe, nieformatowe. Plik, w którym we wszystkich dokumentach występują pozycje o tych samych nazwach i w tej samej kolejności, a odpowiadające sobie pozycje mają wartości o tej samej długości, nazywa się plikiem formatowym. Plik półformatowy różni się od formatowego tym, że odpowiadające sobie pozycje mogą mieć wartości różnej długości, a nieformatowy tym, że dokumenty mogą mieć różną liczbę pozycji.

Chociaż dla użytkownika dokument stanowi najczęściej zwartą całość, to jednak ogólnie tak być nie musi (por. system IDS), co stwierdza m.in. formalna definicja pliku, którą przytoczymy za McGee [109]. W pliku występuje pewien (skończony) zbiór cech. Każda cecha posiada przyporządkowany sobie zbiór wartości. Plik definiuje się jako zbiór trójek postaci (d_i, c_j, w_{ij}), gdzie w_{ij} jest wartością cechy c_j w dokumencie d_i. Innymi słowy d_i jest nazwą dokumentu o numerze i, a c_j nazwą pozycji o numerze j w tym dokumencie.

Celem porównania z podaną wyżej definicją pliku przedstawimy inną definicję (Abraham [2]), która za podstawową jednostkę przyjmuje cały dokument.

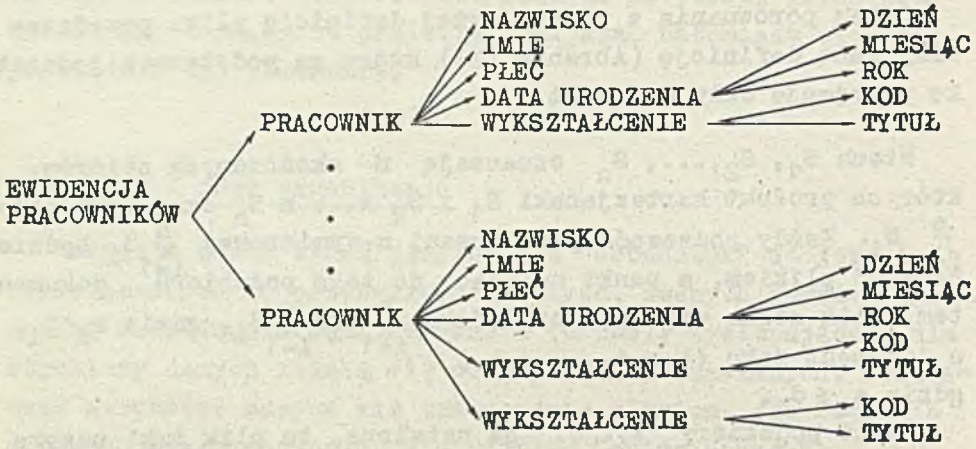
Niech $S_1, S_2, \ldots, S_n$ oznaczają n skończonych zbiorów, których produkt kartezjański $S_1 \times S_2 \times \ldots \times S_n$ oznaczymy przez $\prod_{i=1}^{n} S_i$. Każdy podzespół przestrzeni n-wymiarowej $\prod_{i=1}^{n} S_i$ będziemy nazywać plikiem, a punkt należący do tego podzbioru dokumentem. Plik można więc oznaczyć jako $\prod_{i=1}^{n} s_i \subset \prod_{i=1}^{n} S_i$, gdzie $s_i \subset S_i$, a dokument jako $(a_1, a_2 \ldots, a_n)$, gdzie $a_i \in S_i$.

Jeśli podzbiory $s_i \subset S_i$ są ustalone, to plik taki nazywa się plikiem formatowym, w przeciwnym przypadku - półformatowym. Definicja ta nie obejmuje jednak trzeciego rodzaju plików tj. nieformatowych, których dokumenty nie muszą być (wg terminologii drugiej definicji) punktami przestrzeni n-wymiarowej.

Pozycję elementarną można opisać podając jej nazwę i wartość np. NAZWISKO: KOWALSKI. Jeśli plik jest formatowy, to operując na pozycji można zastąpić nazwę jej (NAZWISKO) numerem tej pozycji w dokumencie. Pewne pozycje elementarne można połączyć w grupę i nadać tej grupie nazwę. Na przykład pozycje elementarne określające DZIEŃ URODZENIA, MIESIĄC URODZENIA, ROK URODZENIA składać się będą na grupę o nazwie DATA URODZENIA.

Następnie pozycje elementarne i grupy można połączyć w dokument i nadać mu nazwę. Na przykład pozycje elementarne NAZWISKO, IMIĘ I PŁEĆ oraz grupy: DATA URODZENIA I WYKSZTAŁCENIE tworzą dokument o nazwie DANE OSOBOWE.

Zbiór dokumentów zawierających dane o pracownikach tworzy plik o nazwie EWIDENCJA PRACOWNIKÓW. Plik ten można zilustrować graficznie w postaci drzewa (rys. 3).



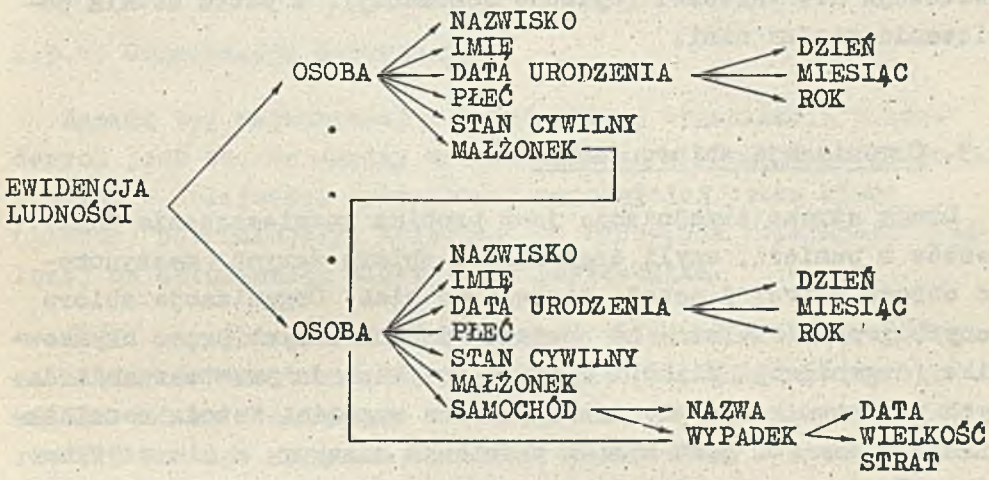
Rys. 3

Zaznaczone na powyższym rysunku strzałki przedstawiają powiązania deklarowane przez użytkownika. Całość takich powiązań w pliku nazywa się jego organizacją. Bardziej skomplikowane organizacje plików, tzn. takie pliki, w których dokumenty nie są powiązane hierarchicznie, można opisać za pomocą grafów skierowanych (rys. 4).

Wydaje się, że omawiane obecnie języki do przetwarzania danych różnią się istotnie w podejściu do struktur danych od algorytmicznych języków programowania (np. ALGOL). Mimo, że formalnie dana zarówno w językach jednego jak i drugiego rodzaju jest parą (nazwa, wartość), to jednak różnice w podejściu do danych rzutują istotnie na sposób ich zapisywania i operowania nimi.

Te różnice wynikają oczywiście z przeznaczenia języków. W przetwarzaniu danych występuje z reguły dużo informacji, przy czym operacje dokonywane są na nich wielokrotnie. Duża ilość informacji, które muszą być przechowywane w pamięciach maszyny powoduje, że struktury danych języka często składają się z dużej liczby elementów. Z drugiej strony większość wykonywanych operacji nie dotyczy całego pliku, a tylko wybranych dokumentów. Dokument może być identyfikowany bądź przez

nazwę bądź przez swą wartość. Ponieważ jednak dokumentów o takich samych nazwach pozycji może być w pliku wiele (a nawet wszystkie), to w rzeczywistości dokumenty można rozpoznawać tylko wg wartości pozycji, bowiem nazwa dokumentu służąca identyfikacji musiałaby z konieczności kryć w sobie opis całego dokumentu. Dlatego też dokumenty są rozpoznawane za pomocą tzw. klucza dokumentu. Klucz definiuje się jako jedną z pozycji w dokumencie.



Rys. 4

W praktyce kluczami dokumentów nazywa się również funkcję kluczy, np. kluczem może być ciąg pozycji elementarnych, ich funkcja logiczna itp. Wśród kluczy wyróżnia się klucze porządkowania i wyszukiwania. Klucze porządkowania służą do wprowadzania dokumentów do pliku, a klucze wyszukiwania do ich znajdowania. Klucze wyszukiwania i porządkowania mogą być (i bardzo często są) identyczne. Klucz, który jest jednoznaczny w pliku nazywa się identyfikatorem.

Sytuacja jest więc istotnie różna niż w przypadku np. algolowej tablicy. Tablica jako tzw. argument strukturalny stosowana jest zasadniczo w algorytmach, w których należy powtarzać operacje na pewnej liczbie argumentów. Dla ułatwienia łączy się je w tablice i operacje wykonuje się w pętli; zmie-

niając indeksy tablicy zmienia się argumenty operacji. W języku występują tylko nazwy elementów (tj. nazwa tablicy i jej indeksy), natomiast ich wartości są nieistotne, a więc przeciwnie niż w plikach. Widać więc, że elementy struktury są tu powiązane przez nazwy. Wynika to z zaznaczonego uprzednio faktu, że w językach takich jak ALGOL najpierw definiuje się strukturę nazw, a potem przyporządkowuje (nadaje) im wartości. W pliku kolejność działań jest niejako odwrotna, tj. najpierw definiuje się wartości (opisuje dokumenty), a potem ustala powiązania między nimi.

2.3. Organizacja zbioru danych

Drugą stroną zagadnienia jest problem rozmieszczenia dokumentów w pamięci, czyli organizacja zbioru danych (maszynowego obrazu pliku) i metoda dostępu do nich. Organizacja zbioru danych jest niezależna od powiązań definiowanych przez użytkownika (organizacji pliku). W wielu językach do przetwarzania danych użytkownik nie wie nawet jak ona wygląda. Metoda rozmieszczania w pamięci jest bowiem problemem maszyny, a nie użytkownika. Powiązania między dokumentami ustalone przez użytkownika (tzw. powiązania logiczne) nie muszą pokrywać się z powiązaniami między ich maszynowymi obrazami (zapisami). Powiązania między zapisami (tzw. powiązania fizyczne) wyznaczone są przez organizacje zbioru danych. Innymi słowy oznacza to, że zapisy odpowiadające powiązanym ze sobą dokumentom nie muszą być ze sobą powiązane i na odwrót.

Ta dwoistość struktur danych nie występuje w językach algorytmicznych z taką ostrością, bowiem programista jest podporządkowany metodzie rozmieszczenia. Dobrym tego przykładem jest tablica. Jest ona bowiem formą miejsc pamięci, w które wstawia się wartości. Identyfikatorom (nazwom) używanym w programie odpowiadają identyfikatory wewnętrzne maszyny, czyli adresy miejsc pamięci.

Mimo niezależności organizacji zbioru danych od użytkownika, organizacja ta powinna być dobierana zależnie od konkretnego zastosowania. Każda ze znanych metod organizowania zbioru danych ma zarówno wady, jak i zalety i zależnie od wykorzystania może okazać się lepsza bądź gorsza od innych.

Omówimy teraz pięć typów organizacji zbioru danych.

2.3.1. Organizacja sekwencyjna

Zasadą tej najstarszej i najprostszej organizacji zbioru danych jest to, że zapisy umieszczane są w pamięci jeden po drugim, w kolejności wyznaczonej najczęściej przez klucz (klucze) porządkujący. Kolejność ta odpowiada kolejności ustalonej na dokumentach pliku przez użytkownika.

Jeśli użytkownik definiuje jakieś zależności hierarchiczne między dokumentami (tj. typu nadrzędny-podrzędny), to te zależności są realizowane w ten sposób, że w dokumencie nadrzędnym zostaje zapisane, że jest on nadrzędny w stosunku do pewnej klasy dokumentów i odpowiednio w podrzędnych. Szukanie dokumentów będących w tego typu relacji z podanym dokumentem odbywa się na zasadzie asocjacyjnej, tj. przez sprawdzanie wartości odpowiedniego pola (obrazu pozycji), w każdym zapisie. Organizacja ta jest przeznaczona głównie dla taśm magnetycznych.

Nowszą wersją organizacji tego typu jest organizacja indeksowo-sekwencyjna opracowana z myślą o dyskach.

Dokumenty pliku można podzielić na klasy ze względu na wartości pewnej cechy. Do danej klasy należą te dokumenty, których wartości danej cechy należą do pewnego ustalonego przedziału wartości tej cechy. Zbiór wartości cechy jest uporządkowany (np. rosnąco).

Zbiór danych, który będzie obrazem tak podzielonego pliku można zorganizować następująco:

- z wszystkich zapisów dokumentów tej samej klasy tworzy się zbiór zorganizowany sekwencyjnie,

- wartość cechy, która jest początkiem (lub końcem) przedziału wyznaczającego klasę dokumentów i adres pierwszego (lub ostatniego) zapisu znajdującego się w danej klasie tworzą parę. Zbiór wszystkich takich par nosi nazwę indeksu.

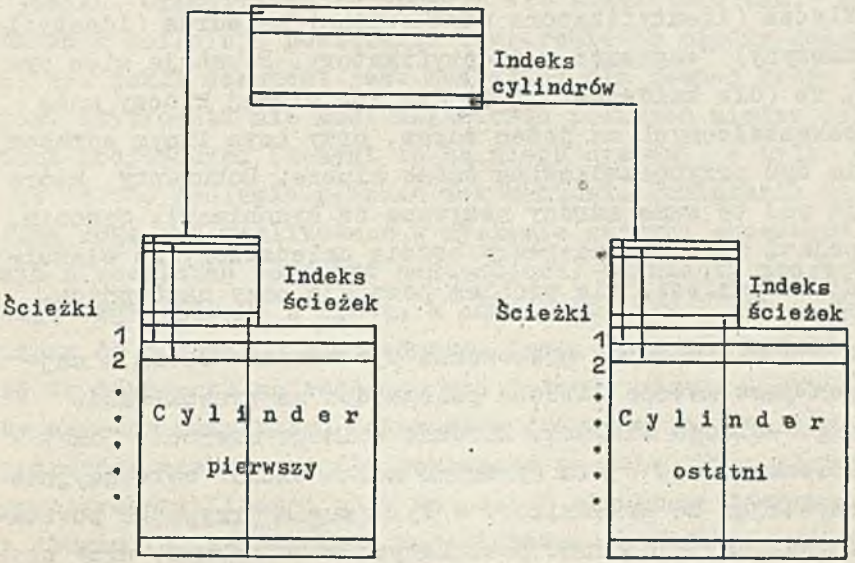
Organizacja zbioru danych, w której zapisy podzielone są na podzbiory zorganizowane sekwencyjnie, a adresy początków tych podzbiorów znajdują się w indeksie nosi nazwę organizacji indeksowo-sekwencyjnej.

W zbiorze zorganizowanym indeksowo-sekwencyjnie może istnieć kilka poziomów indeksu, ponieważ każdy z podzbiorów można również zorganizować indeksowo-sekwencyjnie. W takim przypadku indeks utworzony przy pierwszym podziale na klasy nosi nazwę indeksu głównego. Organizacja indeksowo-sekwencyjna jest odpowiednia szczególnie dla urządzeń o dostępie bezpośrednim, np. dla dysku lub bębna.

Przykład organizacji indeksowo-sekwencyjnej na dysku pokazany jest na rys. 5.

Opisana wyżej organizacja została opracowana przez firmę IBM.

Ogólnie jednak przyjmuje się, że organizacja indeksowo-sekwencyjna polega na tym, że tworzy się katalog (indeks) dokumentów zorganizowany sekwencyjnie, natomiast same dokumenty (z ewentualnie wykasowanym kluczem) mogą być umieszczone w pamięci dowolnie.



Rys. 5

2.3.2. Organizacja bezpośrednia

Zbiór danych o takiej organizacji jest strukturą nieuszeregowaną. Obrazy dokumentów napływające do pamięci są w niej umieszczone w miejscu, którego adres jest wyznaczony w wyniku wykonania pewnej operacji na kluczu dokumentu. Znanych jest wiele metod przekształcania klucza na adres np. przez usuwanie pewnych cyfr z klucza, przez mnożenie, potęgowanie, składanie itp. (patrz Lum 105], 106]). Umieszczenie dokumentu w pamięci jak i wyszukanie go w niej odbywa się tak samo, tj. przez przekształcenie klucza bezpośrednio na adres i dlatego organizacja ta bywa też nazywana organizacją o dostępie obliczanym.

Z organizacją bezpośrednią związany jest pewien problem. Otóż, przedział z jakiego pochodzą wartości kluczy dokumentu jest na ogół bardzo duży, a oczywiście pamięć maszyny przydzielona na pomieszczenie pliku powinna być jak najmniejsza

(liczba miejsc pamięci przeznaczonych na dokumenty powinna być niewiele tylko większa od liczby dokumentów). Przekształcenie klucza (identyfikatora użytkownika) na adres (identyfikator maszyny) "zagęszcza" identyfikatory. Powstaje więc sytuacja, że (dla każdej z metod) dwa lub więcej kluczy może być przekształconych na jeden adres, przy czym innym adresom mogą nie być przyporządkowane żadne klucze. Dokumenty, które trafiają pod te same adresy nazywane są synonimami. Synonim, który pojawi się jako pierwszy będzie umieszczony we właściwym miejscu pamięci, ale problem powstaje przy następnych.

Istnieje wiele metod operowania synonimami. Jedną z najstarszych jest metoda liniowa polegająca na wyszukiwaniu następnego wolnego miejsca. Zarówno wolnego miejsca, jak i potem umieszczonego w nim synonimu można szukać sekwencyjnie lub korzystając ze wskaźników. W tym drugim przypadku powstają tzw. łańcuchy synonimów powiązanych wskaźnikami, przy czym miejsca wolne również tworzą taki łańcuch. Metodą podobną do powyższej jest metoda umieszczania synonimów i tworzenia z nich łańcuchów, ale nie w głównym obszarze pamięci lecz w nadmiarowym. Inną jeszcze metodą jest tzw. metoda Vyssotsky'ego polegająca na tym, że ustala się ciąg przekształceń klucza na adres. Jeśli po pierwszym przekształceniu trafi się na miejsce już zajęte, to dokonuje się następnego przekształcenia klucza.

Niezależnie od wszystkich tych metod stosuje się jeszcze tzw. agregowanie pamięci, polegające na tym, że kilku miejscom pamięci nadaje się ten sam adres. A więc pod jednym adresem może się wtedy pomieścić pewna ustalona liczba dokumentów. Jeśli jednakże liczba synonimów będzie większa, to sytuacja się powtarza.

2.3.3. Organizacja dowiązywana

Zapisy w tak zorganizowanym zbiorze danych są ze sobą powiązane za pomocą wskaźników. Powiązania te definiuje użyt-

kownik, oczywiście w zależności od możliwości zawartych w danym języku. Organizację tę stosuje się zasadniczo tam, gdzie użytkownik definiuje powiązania hierarchiczne między dokumentami, np. jakiś dokument jest nadrzędny dla pewnej grupy dokumentów. Użytkownik nie musi definiować powiązań między dokumentami podrzędnymi (uczyni to za niego system), a tylko określa, że są one podległe pewnemu dokumentowi. Powiązania hierarchiczne mogą być realizowane w systemie różnymi sposobami. Jednym z rozwiązań może być następujące: dokumenty podrzędne zostają organizowane w listę, a adres początku listy zostaje dopisany do dokumentu nadrzędnego. Innym sposobem będzie dopisanie do dokumentu nadrzędnego nie jednego adresu listy, lecz listy adresów wszystkich dokumentów podrzędnych. Wtedy dokumenty podrzędne nie są w ogóle powiązane ze sobą. Przy pierwszej z powyższych realizacji plik ma postać struktury listowej, a przy drugim (o ile nie będą zadeklarowane powiązania między gałęziami) - drzewa. Najogólniejszą strukturą jaką tworzą dokumenty przy takiej organizacji jest sieć.

Specyficzną organizacją dowiązywaną dość często stosowaną (głównie do celów wyszukiwania informacji) jest tzw. organizacja multilistowa. W każdym dokumencie pliku wyróżnia się pewne pozycje jako tzw. słowa kluczowe. Dokument musi zawierać co najmniej jedno takie słowo - pozostałe pozycje nie odgrywają tu żadnej roli. Spis tych słów kluczowych oraz przyporządkowanych im adresów (po jednym) stanowi katalog pliku. Adres w katalogu wskazuje na pierwszy z dokumentów, który zawiera dane słowo kluczowe. W dokumencie tym jest zapisany wskaźnik będący adresem następnego takiego dokumentu. Listy dokumentów przeplatają się ze sobą, ponieważ dokumenty mają wiele słów kluczowych, mogą mieć więc także wspólne początki, a nawet całkowicie się ze sobą pokrywać.

2.3.4. Organizacja odwrócona

Zasada tej organizacji jest zbliżona do zasady organizacji multilistowej. Różnica polega na tym, że w katalogu podany

jest nie tylko adres pierwszego dokumentu zawierającego dane słowo kluczowe, lecz lista adresów wszystkich takich dokumentów. Wyróżnia się dwa rodzaje organizacji odwróconej, mianowicie: częściowo odwrócona i całkowicie odwrócona.

Organizacja całkowicie odwrócona polega na tym, że katalog zawiera spis wszystkich pozycji z wszystkich dokumentów. Właściwe obrazy dokumentów są więc niepotrzebne i można je w ogóle usunąć. Wtedy oczywiście adresy występujące w katalogu należy zastąpić pewnymi innymi jednoznacznymi identyfikatorami dokumentów.

Organizacja częściowo odwrócona różni się od powyższej tym, że obrazy oryginalnych dokumentów są zachowane bądź dlatego, że użytkownik tak chce, bądź też dlatego, że katalog nie zawiera wszystkich pozycji dokumentów. Oznacza to, że w dokumentach nie wszystkie pozycje są traktowane jako słowa kluczowe.

Tradycyjny termin "plik odwrócony" (ang. inverted) wywodzi się stąd, że w początkach przetwarzania plików technika wyszukiwania była następująca: najpierw rozmieszczano dokumenty w pamięci, a dopiero potem przy poszukiwaniu dokumentów sprawdzano każdy dokument, czy posiada dane słowo kluczowe. Na plikach odwróconych działa się rzeczywiście na odwrót, tj. najpierw sprawdza się jakie słowa kluczowe dokument zawiera, a dopiero potem umieszcza się go odpowiednio w pamięci.

2.3.5. Organizacje grupowe

Zbiory danych o organizacji tego typu przeznaczone są do wyszukiwania i to głównie według wielu słów kluczowych (do wyszukiwania według jednego słowa kluczowego stosuje się organizacje odwrócone lub multilistowe). W pliku takim dokumenty dzieli się na grupy. Celem tej organizacji jest ułatwienie wyszukiwania. Na żądanie wyszukiwania dokumentu spełniającego pewne warunki otrzymuje się zazwyczaj w odpowiedzi wiele

takich dokumentów. Ułatwieniem byłoby więc, gdyby dokumenty te mogły być wyszukane jednocześnie jako jedna grupa. Ułatwieniem byłoby to nawet wtedy, gdyby były one umieszczone w pamięci obok siebie, bowiem dla każdego z nich należałoby sprawdzić czy zawiera on taką jak trzeba kombinację słów kluczowych. A więc organizacja ta jest dla wyszukiwania według wielu kluczy odpowiednikiem organizacji odwróconej.

Dokumenty dzieli się na grupy badając ich podobieństwo między sobą. Jako miarę podobieństwa można przyjąć wartość cosinusa kąta między dwoma dokumentami przedstawionymi w postaci n-elementowych wektorów, gdzie n jest liczbą słów kluczowych w pliku. Podział na grupy odbywa się więc przez wyszukanie zagęszczeń dokumentów, to jest ustalając pewną dolną wartość podobieństwa, poniżej której dokumenty są już zbyt mało podobne, a następnie badając dla każdego dokumentu czy jest dostatecznie dużo dokumentów do niego podobnych. Po znalezieniu takich zagęszczeń oblicza się w nich centrum i dokument odpowiadający centrum będzie reprezentantem grupy. Wyszukiwanie w pliku dokumentów odpowiadających zadanemu pytaniu będzie więc polegało na zbadaniu podobieństw między pytaniem (też n-elementowym wektorem) a reprezentantami grup.

Inną metodą podziału na grupy, przeznaczoną raczej do wyszukiwania według funkcji logicznych słów kluczowych, jest metoda (Wong [155]) oparta na algebrze Boole'a. W metodzie tej rozpatruje się zbiory dokumentów zawierających dane słowo kluczowe (n słów kluczowych - n zbiorów). Grupami są tu niepuste przecięcia wszystkich tych zbiorów lub ich uzupełnień (do całego pliku). Każdą funkcję logiczną n słów kluczowych można przedstawić w postaci kanonicznej, tj. jako sumę logiczną iloczynów logicznych, których czynnikami są słowa kluczowe albo ich negacje. A więc każda grupa odpowiada jednej koniunkcji z postaci kanonicznej, czyli każdemu pytaniu w postaci funkcji logicznej słów kluczowych odpowiada pewna suma grup.

Istnieją jeszcze metody tzw. zrównoważonego grupowania dokumentów. Wszystkie metody tego typu spełniają trzy następujące postulaty:

1) wszystkie dokumenty odpowiadające jednemu pytaniu powinny się znajdować w jednej grupie,

2) każdą grupę zidentyfikować można algebraicznie (np. obliczyć adres grupy wg podanego wzoru),

3) grupy dokumentów są prawie równoliczne.

Przykład takiej metody podano w pracy Abrahama i in. [2]. Opisana metoda działa tylko w dość szczególnym przypadku, mianowicie gdy w pytaniach występują tylko dwie cechy oraz gdy plik jest binarny (każda cecha ma dwuelementowy zbiór wartości). Rozważa się wtedy geometrię euklidesową lub rzutową. Przestrzeń jest n-wymiarowa, a współrzędne przyjmują wartości należące do pierścienia całkowitego Zp, gdzie p - liczba pierwsza. Cechę reprezentować będzie punkt tej przestrzeni, a grupę - prosta. Prostą można oczywiście wyznaczyć algebraicznie (postulat 2), rozwiązując układ n-1 równań opisujący tę prostą w geometrii. Każde pytanie (po dwie cechy) określa dwa punkty przestrzeni, a one wyznaczają jednoznacznie prostą (czyli grupę).

Metody podane w pracach Abrahama [1] i Gosha [67], [68] omawiają przypadki ogólniejsze od powyższego. Niestety, każda metoda ogólniejsza albo jest wyraźnie gorsza albo w ogóle nie da się zastosować do przypadku bardziej szczegółowego. Podobne podejście zaprezentowano w pracy Yamamoto [157] (również stosującej geometrię na zbiorach punktów dyskretnych oraz w pracy Chowa [26] opierającej się na prawach kombinatoryki).

3. OPEROWANIE DANYMI I METODY DOSTĘPU

W rozdziale 2 niniejszego opracowania omówione zostały struktury danych i struktury zapisu ze szczególnym wyróżnie-

niem pliku - podstawowej dla systemu przetwarzania danych struktury danych. Strukturę zapisu pliku może być bardzo różna. W rozdziale 3 przyjmiemy, że obrazem pliku w pamięci maszyny jest zbiór danych, a obrazami dokumentów pliku są zapisy. W istniejących systemach przetwarzania danych obrazy dokumentów tego samego pliku, tzn. zapisy mogą występować w różnych zbiorach danych.

Niniejszy rozdział składa się z dwóch zasadniczych części: pierwszej - omawiającej operacje niezbędne użytkownikowi do korzystania z systemu przetwarzania danych i drugiej - omawiającej operację i metody umożliwiające wykonanie operacji użytkownika.

3.1. Operowanie danymi

W typowych systemach przetwarzania danych można wyróżnić następujące operacje (jedno lub wieloargumentowe) niezbędne użytkownikowi:

- definiowanie i przedefiniowywanie pliku,
- aktualizacja pliku,
- wypisywanie informacji z pliku,
- wybieranie informacji z pliku,
- operacje pomocnicze oraz
- inne operacje przeważnie wynikające z implementacji systemu.

3.1.1. Definiowanie i przedefiniowywanie pliku

Definiowanie pliku jest operacją, za pomocą której użytkownik określa organizację pliku. Określenie organizacji obejmuje:

a. definicję pozycji elementarnych, która składa się z podania:
   - nazwy pozycji elementarnej,
   - typu wartości,

- przedziału wartości,
- innych parametrów.

b. definicję grup, która składa się z podania:
   - pozycji elementarnych tworzących grupę,
   - powiązań tych pozycji elementarnych,
   - nazwy grupy,
   - innych parametrów,

c. definicji dokumentu, która podaje pozycje elementarne i grupy tworzące dokument i określa ich wzajemne powiązania oraz nazwę dokumentu,

d. definicji kluczy dokumentów,

e. definicji pliku, która podaje dokumenty tworzące plik oraz ich wzajemne powiązania.

Przedefiniowanie pliku jest podobne do definiowania z tym, że określa zmiany w organizacji pliku już zdefiniowanego.

3.1.2. Aktualizacja pliku

Aktualizacją pliku jest operacja, za pomocą której użytkownik może zmieniać elementy pliku nie zmieniając jego organizacji.

Aktualizacja jest operacją złożoną i może obejmować:

- dodawanie dokumentów do pliku,
- wykreślanie dokumentów z pliku,
- modyfikowanie wartości pozycji elementarnych w dokumentach pliku,
- dodawanie grup lub pozycji elementarnych do dokumentów w pliku,
- wykreślanie grup lub pozycji elementarnych z dokumentów w pliku.

Proces wykonywania aktualizacji w zależności od rodzaju przetwarzania może być:

- partiowy (batch),
- bezpośredni (random).

W przetwarzaniu partiowym dokumenty aktualizujące grupowane są w tzw. plik transakcyjny i sortowane zwykle według kluczy porządkowania w taki sam sposób, jak dokumenty w pliku do aktualizacji. W czasie przetwarzania plik jest aktualizowany dokumentami innego pliku. Na ogół dokumenty aktualizujące zawierają informacje o rodzaju aktualizacji tzn. dodawaniu, wykreślaniu itd.

W przetwarzaniu bezpośrednim, każdy dokument aktualizujący powoduje natychmiastowe przetwarzanie pliku i wykonanie operacji aktualizacji.

Przetwarzanie wielodostępne może być zarówno partiowe jak i bezpośrednie, a wykonanie operacji aktualizacji zależy od uprawnień użytkownika.

Z operacją aktualizacji związany jest bardzo ważny dla użytkownika problem przechowywania informacji. Często jest to rozwiązywane przez przechowywanie kopii plików: aktualizowanego i aktualizującego.

3.1.3. Wypisywanie informacji z pliku

Operacja wypisywania informacji z pliku służy do wybrania z pliku określonych informacji i wyprowadzenia ich w żądanej postaci na urządzenie wyjściowe np. drukarkę wierszową, monitor ekranowy. Operacja ta nie zmienia ani organizacji, ani zawartości pliku.

Wypisywanie informacji z pliku jest operacją złożoną z:

- wyszukiwania w pliku dokumentów spełniających pewne kryteria podane przez użytkownika, np. zgodność klucza lub kluczy wyszukiwania z kluczem lub kluczami dokumentu,

- ewentualnego posortowania wybranych dokumentów wg podanego przez użytkownika klucza lub kluczy porządkujących,
- ewentualnego wykonania na określonych pozycjach elementarnych dokumentów operacji arytmetycznych, logicznych lub innych,
- zredagowania dokumentu wyjściowego w postaci określonej przez użytkownika.

Wynikiem wykonania operacji jest wyprowadzenie na określone urządzenie wyjściowe żądanych informacji w żądanej postaci.

3.1.4. Wybieranie informacji z pliku

Operacja wybierania informacji z pliku służy do tworzenia wg podanych warunków nowego pliku, który można wykorzystywać w dalszym przetwarzaniu. Operacja ta jest identyczna z operacją wypisywania informacji z pliku z tym, że zredagowane dokumenty wyjściowe przenoszone są do innej pamięci pomocniczej.

3.1.5. Operacje pomocnicze

Najczęściej są to operacje arytmetyczne (dodawania, odejmowania, mnożenia, dzielenia), logiczne (alternatywa, koniunkcja, negacja) oraz porównywania dwu pozycji elementarnych.

Operacje te wykonywane są często na częściach pozycji elementarnych.

3.1.6. Inne operacje

Oprócz omówionych powyżej operacji wykonywanych na plikach można określić dodatkowe operacje występujące w wielu istniejących systemach. Jedną z nich jest operacja tworzenia pliku.

Za pomocą tej operacji, na podstawie operacji definicji pliku, tworzy się z dokumentów wejściowych plik. Operacja ta często jest zastępowana operacją aktualizacji wykonywaną na pliku, tzw. pustym, tzn. zdefiniowanym pliku nie zawierającym żadnego dokumentu. Oprócz powyższej operacji można określić operacje organizacyjne wykonywane na pozycjach elementarnych. Są to operacje:

- łączenia pozycji elementarnych, tzn. tworzenia grup,
- rozłączania pozycji elementarnych, tzn. rozdzielania grup na pozycje elementarne.

3.2. Metody dostępu

3.2.1. Jednostki struktury zapisu

Przy omawianiu sposobów wykonania operacji użytkownika konieczne jest wprowadzenie pewnych jednostek struktury zapisu. Jako te jednostki można przyjąć: element zapisu, zapis, blok, zbiór danych, bazę danych.

"Element zapisu" - najmniejsza jednostka danych. Może być bitem, bajtem, słowem. Na ogół elementem zapisu jest jednostka pamięci głównej pobierana lub zapamiętana w czasie jednego cyklu.

"Zapis" - zbiór powiązanych ze sobą elementów zapisu, który można zidentyfikować. Sposób powiązania zależy od organizacji zapisu.

"Blok" - jednostka wymiany pomiędzy pamięcią główną a pomocniczą. Blok składa się z części lub całości jednego zapisu lub wielu zapisów.

"Zbiór danych" - zbiór bloków posiadający nazwę.

"Baza danych" - zbiór danych spełniający określone warunki (patrz rozdział 4).

Omówione powyżej jednostki można powiązać z elementami podstawowej struktury użytkownika - plikiem.

Pozycję elementarną, a właściwie jej wartość można przedstawić w zależności od podanego przez użytkownika typu i przedziału wartości, za pomocą elementów zapisu lub ciągu elementów zapisu.

Grupę i dokument można przedstawić za pomocą zapisu. Na ogół uważa się, że zapis jest obrazem dokumentu pamięci maszyny.

Zapisy, które są obrazami dokumentów tego samego pliku mogą się znajdować w różnych zbiorach danych. Jednak bardzo często zbiór danych jest obrazem jednego pliku.

Niezależnie od organizacji zbioru danych zapisy grupowane są w bloki. W praktyce występują przypadki, że blok zawiera jeden zapis, a nawet tylko część zapisu. Oprócz zapisów w bloku występują informacje charakteryzujące blok, np. długość bloku, długość zapisów, liczbę zapisów, położenie (adresy) zapisów w bloku itp. oraz informacje kontrolne np. suma kontrolna bloku, numer bloku.

Poza blokami zawierającymi zapisy istnieją bloki organizacyjne, które zawierają ogólne informacje o zbiorze danych.

Można wyróżnić dwa typy bloku organizacyjnego:

- blok nagłówka
- blok końca

Najczęściej w bloku nagłówka znajdują się informacje identyfikujące zbiór danych np. nazwa zbioru danych, data utworzenia zbioru danych oraz informacje zabezpieczające np. klucz dostępu. Blok końca służy do zaznaczenia końca zbioru. Może zawierać także pewne informacje kontrolne. Przy omawianiu sposobów umieszczania zbioru danych w pamięci pomocniczej wygodniej jest wprowadzić pojęcie tomu. Tom jest całkowitą informacją dostępną z urządzenia we-wy w pojedynczym ładowaniu

(Engles [49]). Dla taśm magnetycznych tomem jest szpula; dla dysku - pakiet dyskowy.

W przypadku, gdy zbiór danych będzie tak duży, że nie zmieści się w jednym tomie, blok końca będzie blokiem końca tomu, tzn. będzie w nim zawarta informacja, że dalsza część zbioru znajduje się w innym tomie.

3.2.2. Związki metod dostępu z urządzeniami pamięci i organizacją zbiorów

Metoda dostępu jest algorytmem znajdowania w pamięci miejsca zapisu lub miejsca na umieszczenie zapisu.

G.H. Mealy [111], uważa, że dostęp do danych jest cechą przetwarzania danych, a nie danych lub sposobu ich przedstawienia. Kolejność w jakiej dane są pobierane lub umieszczane nie zależy lub nie powinna zależeć od organizacji zbioru danych.

W istniejących obecnie systemach zarządzania danymi metody dostępu są związane z organizacjami zbiorów danych, a także z rodzajem pamięci, w której te zbiory się znajdują. Metoda dostępu, za pomocą której organizuje się zbiór nosi nazwę głównej metody dostępu.

W tym rozdziale zostaną omówione metody dostępu: sekwencyjnego, indeksowo-sekwencyjnego i bezpośredniego.

Podstawowym elementem większości operacji jest operacja wyszukania zapisu w zbiorze. Sposób wykonywania tej operacji jest ściśle związany z metodą dostępu.

Wynikiem wyszukiwania może być:

- przesłanie wyszukanego zapisu do określonego obszaru pamięci głównej lub
- podanie adresu zapisu lub

- podanie informacji, że szukany zapis nie występuje w przeszukiwanym zbiorze danych; w tym ostatnim przypadku może być podana informacja gdzie ten zapis powinien być umieszczony.

Operację wyszukiwania stosuje się oczywiście do zbioru danych, w którym spodziewane jest znalezienie szukanego zapisu.

Podane dalej opisy metod dostępu są tylko ich ogólną charakterystyką. Może być wiele różnych realizacji metod dostępu w zależności od maszyny oraz zastosowań.

3.2.2.1. Sekwencyjna metoda dostępu

Metoda dostępu sekwencyjnego do zapisu mającego określoną wartość klucza polega na przeszukiwaniu zbioru danych zapis po zapisie i porównywaniu wartości klucza każdego zapisu z wartością klucza szukanego zapisu. Jeśli porównywane wartości są równe, wtedy wynikiem będzie podanie adresu zapisu lub przesłanie zapisu.

Przeszukiwanie zbioru danych wykonywane jest następująco:

1. z tomu pamięci pomocniczej, w którym znajduje się przeszukiwany zbiór danych, przenoszony jest do pamięci głównej blok,

2. blok różny od bloku organizacyjnego przeszukiwany jest w podany powyżej sposób. Jeśli w bloku nie występuje zapis lub jeśli klucz nie jest identyfikatorem, zostaje wykonana ponownie czynność 1, w przeciwnym przypadku przeszukiwanie zostanie zakończone,

3. przeniesienie do pamięci bloku końca powoduje przerwanie przeszukiwania i ewentualne podanie informacji o braku zapisu w zbiorze.

Metodę dostępu sekwencyjnego można stosować do zbiorów danych o dowolnej organizacji. Jednak przede wszystkim jest ona stosowana do zbiorów danych zorganizowanych sekwencyjnie.

W przypadku, gdy klucz wyszukiwania jest jednocześnie kluczem porządkowania można do przeszukiwania zbioru zorganizowanego sekwencyjnie zastosować wyszukiwanie binarne, polegające na kolejnym podziale zbioru na dwie części i stwierdzeniu, w której z tych części znajduje się poszukiwany zapis.

Jeśli klucz wyszukiwania jest identyfikatorem, wtedy w zbiorze może wystąpić najwyżej jeden zapis o wyszukiwanej wartości klucza.

3.2.2.2. Indeksowo-sekwencyjna metoda dostępu

W celu znalezienia zapisu mającego określoną wartość klucza indeksowo-sekwencyjną metodą dostępu przeszukiwany jest indeks. Jeśli wartość klucza szukanego zapisu należy do przedziału wyznaczonego przez dwie sąsiednie wartości kluczy w indeksie, to zostanie przeszukany sekwencyjnie podzbiór zapisów ograniczony adresami przyporządkowanymi tym kluczom.

Przeszukiwanie indeksu odbywa się następująco:

1. z tomu pamięci pomocniczej pobierany jest blok zawierający indeks dla przeszukiwanego zbioru danych. Indeks nie musi znajdować się przy zbiorze danych,

2. wartość klucza szukanego zapisu porównywana jest z wartością klucza w indeksie. Oznaczmy przez k' - wartość klucza szukanego dokumentu, a przez k_j ($j = 1,\ldots, n$) - wartości klucza w indeksie. Jeśli $k_j \leq k' < k_{j+1}$ $(1 \leq j \leq n-1)$ lub $k_n \leq k'$, to należy rozpocząć poszukiwanie zbioru danych od adresu podanego odpowiednio przy wartości klucza k_j lub k_n. Metodę dostępu indeksowo-sekwencyjnego można stosować tylko do zbioru danych o organizacji indeksowo-sekwencyjnej.

3.2.2.3. Metoda dostępu bezpośredniego

W tej metodzie znajdowanie miejsca zapisu mającego określoną wartość klucza polega na znalezieniu wartości funkcji, której argumentem jest wartość tego klucza. Wartość funkcji jest adresem szukanego zapisu. W metodzie dostępu oprócz określenia funkcji należy określić sposób postępowania z synonimami.

Przekształcenie wartości klucza w adres składa się z dwóch etapów:

- wartość klucza przekształcana jest do postaci, która pozwala na wykonywanie operacji arytmetycznych,
- liczba odpowiadająca wartości klucza jest przekształcana do przedziału, w którym zawarte są adresy.

Etap pierwszy należy wykonać w przypadku, gdy wartością klucza jest ciąg znaków alfabetycznych lub alfanumerycznych. Można wtedy np. zakodować litery a, b,... z jako liczby dziesiętne 11, 12,.... 36.

Długość klucza będzie liczbą cyfr po przekodowaniu.

Istnieje wiele metod przekształcania wartości klucza na adres. Najbardziej znane są metody:

- obcięcia lub ekstrakcji,
- dzielenia,
- podnoszenia do kwadratu,
- składania.

Metoda obcięcia polega na przyjęciu za adres pierwszych cyfr wartości klucza. Metoda ekstrakcji polega na wybraniu jako adresu pewnej liczby cyfr ze środka wartości klucza. Obie są proste i szybkie, ale dają dużo synonimów.

W metodzie dzielenia wartość klucza jest dzielona przez dodatnią liczbę całkowitą, często liczbę pierwszą. Liczba dodatnia oznacza liczbę dostępnych adresów dla danego zbioru

danych. Reszta z dzielenia jest adresem zapisu o tej wartości klucza.

Metoda podnoszenia do kwadratu polega na podnoszeniu wartości klucza do kwadratu, a następnie stosowana jest metoda obcięcia lub ekstrakcji.

W metodzie składania, wartość klucza jest podzielona na kilka części, które są dodane do siebie modulo 2 lub 10. Sposoby postępowania z synonimami omówione są w rozdziale 2 niniejszego opracowania.

Znajdowanie miejsca zapisu w metodzie bezpośredniego dostępu odbywa się następująco:

1. przekształca się wartość klucza szukanego zapisu w adres wg podanych powyżej zasad, korzystając z metody przekształcenia, za pomocą której ten zbiór danych został zapisany,

2. pobiera się z tomu pamięci pomocniczej odpowiedni blok i sprawdza, czy pod tym adresem w zbiorze danych znajduje się zapis. Jeśli występuje, to nastąpi porównanie nieprzekształconej wartości klucza szukanego zapisu z wartością klucza tego zapisu. Jeśli te wartości są różne, to należy przeszukać (przez porównywanie kluczy) obszar synonimów związany z tym adresem.

Metodę bezpośredniego dostępu stosuje się przede wszystkim do zbiorów zorganizowanych bezpośrednio.

3.2.3. Operacje na zbiorach danych

Na zbiorach danych o dowolnej organizacji można wykonywać działania, które zapewnią wykonanie operacji określonych przez użytkownika.

Najczęściej spotykanymi działaniami są:

- dodawanie zapisu do zbioru,

- wykreślanie zapisu ze zbioru,
- modyfikowanie zapisu w zbiorze,
- redagowanie zapisu w zbiorze,
- sortowanie zbioru danych,
- reorganizacja zbioru danych.

Ostatnia z powyższych operacji może w pewnych przypadkach zmienić organizację zbioru danych. Sposób wykonania operacji zależy od organizacji zbioru. Operacje dodawania, wykreślania i modyfikowania zapewniają wykonanie operacji aktualizacji pliku określonej przez użytkownika. Operacje redagowania i sortowania zapewniają wykonanie operacji wybierania i wypisania informacji z pliku.

3.2.3.1. Dodawanie zapisu do zbioru danych

Dodawanie zapisu jest wstawieniem do zbioru danych w odpowiednie miejsce nowego zapisu z zachowaniem organizacji zbioru. Wykonanie dodawania polega na:

- wyszukaniu w zbiorze danych za pomocą odpowiedniej dla organizacji zbioru metody dostępu (najczęściej głównej) miejsca, w które należy wstawić zapis,
- wstawieniu w odszukane miejsce nowego zapisu. Sposób wstawienia zależy od organizacji zbioru i rodzaju urządzenia, na którym ten zbiór się znajduje.

Na przykład wstawienie zapisu do zbioru zorganizowanego sekwencyjnie, znajdującego się na taśmie magnetycznej, polega na przeniesieniu wszystkich zapisów poprzedzających miejsce wstawienia na nową taśmę magnetyczną, wstawieniu na tę taśmę nowego zapisu i przeniesieniu pozostałych zapisów na tę samą taśmę.

3.2.3.2. Wykreślanie zapisu ze zbioru danych

Wykreślenie zapisu jest usunięciem zapisu ze zbioru z zachowaniem organizacji zbioru lub z zaznaczeniem w zapisie, że nie należy on do zbioru. Wykonanie wykreślenia przebiega podobnie jak dodawanie. Na przykład wykreślenie zapisu ze zbioru o organizacji odwróconej polega na usunięciu jego adresu z indeksu.

3.2.3.3. Modyfikowanie zapisu w zbiorze danych

Modyfikowanie zapisu jest zmianą wartości elementów zapisu. Wykonanie modyfikacji polega na znalezieniu zapisu w zbiorze, a następnie dokonaniu w nim podanych zmian.

3.2.3.4. Redagowanie zapisu w zbiorze danych

Redagowanie jest operacją służącą do zmiany kolejności elementów zapisu bez zmiany długości zapisu, jeśli zapis pozostaje w zbiorze danych lub z ewentualną zmianą długości i dodaniem nowych elementów zapisu, jeśli zapis zostanie przeniesiony do innego zbioru danych.

3.2.3.5. Sortowanie zbioru danych

Sortowanie jest jedną z najczęściej wykonywanych operacji i na ogół jest rozumiane jako takie przestawienie zapisów w zbiorze, aby występowały one w określonej z góry kolejności. Kolejność ta jest wyznaczona przez zadaną na kluczach zapisów relację zwrotną, spójną i przechodnią.

Ogólniej rzecz biorąc sortowanie można określić jako przekształcenie dowolnego zbioru (np. zapisanego w postaci listy) w listę zgodną ze wspomnianą wyżej relacją.

3.2.3.6. Reorganizacja zbioru danych

Reorganizacja jest operacją wykonywaną najczęściej w celu uzyskania najefektywniejszego wykorzystania obszaru, w którym znajduje się zbiór danych bez zmiany jego organizacji. Polega na usunięciu wykreślonych zapisów, często utworzeniu innych bloków. Reorganizacja może także służyć do zmiany organizacji zbioru danych, np. w przypadku przeniesienia zbioru danych z taśmy magnetycznej na dysk. Omówione powyżej przyczyny reorganizacji najczęściej nie wynikają z potrzeb użytkownika. Za pomocą tej operacji można także zapewnić wykonanie żądanej przez użytkownika operacji przedefiniowania pliku.

4. BANK DANYCH

Jednym z niedawno wprowadzonych, a już bardzo rozpowszechnionych pojęć z zakresu organizacji danych jest tzw. "bank danych". Pojęcie to nie jest dotychczas jednoznacznie określone, a różne osoby różnie to pojęcie interpretują. Nie wiadomo również, jakie warunki musi spełnić plik, lub zespół plików, aby go można było nazwać bankiem danych. Przykładem różnej interpretacji banku danych może być stwierdzenie, że jedno z przedsiębiorstw zorganizowało dla własnych potrzeb 243 banki danych; wiele osób stwierdzi, że przytoczone 243 banki nie są bankami danych w ich rozumieniu.

Pojęciem ściśle związanym z bankiem danych jest baza danych. Niektóre interpretacje obu tych pojęć zostaną przedyskutowane w pierwszym punkcie niniejszego rozdziału. W drugim punkcie zawarta jest propozycja określenia banku danych, w następnych zaś punktach omówione są różne zagadnienia związane z tym pojęciem.

4.1. Bank danych a baza danych

4.1.1. Pojęcie banku danych wg Normy Polskiej

Polska Norma pt.: "Przetwarzanie danych i komputery. Podstawowe nazwy i określenia" (PN-71/T-01016) określa bank danych jako "grupę powiązanych ze sobą plików obejmujących całość informacji potrzebnych do eksploatacji określonego systemu elektronicznego przetwarzania danych". W normie tej nie występuje w ogóle pojęcie bazy danych.

Wydaje się, że ta definicja nie oddaje w pełni istotnych cech banku danych. Przede wszystkim nie wiadomo na czym ma polegać powiązanie plików, czy wystarczy po prostu ich suma logiczna. Wydaje się, że jeśli weźmiemy wszystkie zbiory (pliki) potrzebne do eksploatacji systemu EPD dla tego przedsiębiorstwa, to jeszcze nie uzyskamy banku danych.

4.1.2. Baza danych jako wydzielony podzbiór banku danych

W tej interpretacji nie określa się szczegółowo, co to jest bank danych. Intuicyjnie traktuje się, że jest to zespół danych stanowiący pewną całość i wykorzystywany przez wielu użytkowników. Poszczególni użytkownicy mają dostęp do podzbioru banku danych. Podzbiory te, na ogół różne od całego banku danych, noszą nazwę baz danych. Stosując analogię do banków finansowych można powiedzieć, że bank danych odpowiada bankowi, a baza danych - kontu w banku. Bank danych można również nazywać wspólną bazą danych.

Przy tej interpretacji cytowane uprzednio zdanie o 243 bankach (bazach?) można interpretować następująco: wydzielono 243 klasy użytkowników, wyznaczając dla każdej klasy podzbiór dostępnych danych.

4.1.3. Bank danych jako baza plus oprogramowanie

W tej interpretacji znów nie określa się, co to jest baza danych, lecz intuicyjnie traktuje się ją tak, jak bank danych

w interpretacji poprzedniej. Z tak rozumianą bazą danych związane są różne zagadnienia: sposób interpretacji danych, sposób ich zapisu, metody i programy operowania danymi itp. Bankiem danych jest baza danych wraz z całokształtem zagadnień z tą bazą związanych. Należy zwrócić uwagę, jak nieprecyzyjne jest to sformułowanie.

Powyższa interpretacja banku danych i bazy danych znajduje się np. u Ahrensa [6](str. 17). Podane tam definicje brzmią w tłumaczeniu następująco:

- "jeśli dokumenty (rekordy) różnych plików (file, Datei), należące do jednego zakresu organizacyjnego są logicznie między sobą powiązane, wówczas mówi się o bazie danych (data base, Structurdatei, Integrierten Datei)",
- "forma organizacji, przedstawiająca łącznie wszystkie pliki jednego zakresu organizacyjnego jest nazywana bankiem danych (data bank, Datenbank)".

W definicjach powyższych nie wyjaśnia się, co to jest zakres organizacyjny (Organisationsbereich) oraz forma organizacji (Organisationsform). Prawdopodobnie pod pojęciem zakresu organizacyjnego należy rozumieć przedsiębiorstwo (lub inną jednostkę równorzędną), a pod pojęciem formy organizacji - organizację danych (względnie organizację zapisu danych) wraz z metodami operowania danymi (lub z metodami dostępu do danych).

4.1.4. Inne interpretacje banku danych

Istnieje wiele prób definicji banku danych, np.: bank danych jest to zbiór z maksimum powiązań i minimum redundancji (Engles [49]).

Istnieje również wiele pokrewnych pojęć, jak np. wspólna baza danych, zintegrowana baza danych itp. Żadne z tych pojęć nie ma jednak dobrej definicji.

4.2. Definicja banku danych i bazy danych

Zanim zostanie przedstawiona próba definicji banku danych i bazy danych omówimy niektóre cechy, które są istotne dla tych pojęć. Stosowana terminologia (bank i baza danych) jest już tak dobrana, aby była zgodna z proponowanymi w dalszej części definicjami.

4.2.1. Niepowtarzalność danych

Jedną z głównych cech banku danych winno być jednokrotne występowanie każdej danej w bazie danych. Nie wyklucza się możliwości podziału bazy danych na zbiory; w przeciwieństwie do rozwiązań konwencjonalnych, w których dane występują w każdym zbiorze oddzielnie, dane w bazie danych występują tylko raz, a specjalnie zorganizowany system odwołań zapewnia dostępność tych danych w różnych zbiorach.

Jednokrotny zapis każdej danej nie jest podyktowany chęcią zaoszczędzenia miejsca na zapis tej danej - system odwołań zajmuje często więcej miejsca - lecz uniknięciem sprzeczności w przypadku aktualizacji danych.

4.2.2. Korzystanie z danych przez wielu użytkowników

Istotną cechą odróżniającą bank danych od systemu opartego na konwencjonalnych zbiorach jest liczba użytkowników. Nie definiuje się przy tym, co się rozumie przez określenie "użytkownik", przykładowo można powiedzieć, że różnymi użytkownikami są różne przedsiębiorstwa lub różne komórki tego samego przedsiębiorstwa, nie będą natomiast różnymi użytkownikami różni pracownicy tej samej komórki. Przy tym wyjaśnieniu zbiory konwencjonalne ograniczają się zazwyczaj do 1-2 użytkowników; bank danych służy natomiast wielu użytkownikom. Charakterystyczne jest przy tym, że

- wielu użytkowników może jednocześnie korzystać z banku danych,
- zwykle korzystanie z banku danych odbywa się poprzez urządzenia końcowe zainstalowane nie w ośrodku obliczeniowym, lecz bezpośrednio u użytkownika, który dzięki temu ma wrażenie, że jest aktualnie jedynym użytkownikiem banku danych.

Jednoczesne korzystanie z banku danych przez wielu użytkowników jest związane z koniecznością rozwiązania wielu nowych zagadnień. Klasycznym przykładem takiego zagadnienia jest próba jednoczesnej aktualizacji tej samej danej przez dwóch użytkowników.

4.2.3. Zabezpieczenie przed niepowołanym dostępem

Korzystanie z tego samego banku danych przez wielu użytkowników postawiło nowe zagadnienie: każdemu użytkownikowi należy udostępnić tylko te dane, do których jest on upoważniony. Innymi słowy konwencjonalne zabezpieczenie przed niepowołanym dostępem do danych na poziomie zbioru musi być zastąpione zabezpieczeniem na poziomie poszczególnych danych lub ich grup.

4.2.4. Niezależność programów użytkownika od struktury zapisu

Wraz z rozwojem zastosowań zmieniają się wymagania użytkownika w stosunku do banku danych. Użytkownik wprowadza do bazy danych nowe dane, niektóre zbędne usuwa, zmienia zasadnicze operacje na danych itp. Niezbędne jest, aby przy tych wszystkich modyfikacjach stare programy działały bez zmian, nawet wówczas, gdy uległa zmianie struktura zapisu danych, z których te programy korzystają; jedynie efektywność działania tych programów może ulec zmianie.

Niezależność programów użytkownika od struktury zapisu nie oznacza, że jest również odwrotnie: struktura danych i struktura zapisu nie zależą od potrzeb użytkowników. Struktura da-

nych jest przez użytkownika określana i jest z potrzebami tego użytkownika ściśle związana. Natomiast wybranie odpowiedniej struktury zapisu ma zasadnicze znaczenie dla efektywności działania systemu, jest więc dostosowywane do najczęściej wykonywanych przez użytkownika operacji na danych.

4.2.5. Próba definicji banku danych i bazy danych

Reasumując powyższe proponuje się przyjęcie następujących definicji:

- bazą danych nazywamy zbiór danych zapisanych w pamięci maszyny cyfrowej w ten sposób, że każda dana występuje tylko raz oraz, że w zbiorze tym wyróżniane są podzbiory odpowiadające wymaganiom użytkownika w zakresie zapewnienia ochrony dostępu do danych,
- bankiem danych nazywamy bazę danych wraz z odpowiednim aparatem programowym zapewniającym korzystanie z bazy danych wielu użytkownikom jednocześnie, przy czym każdy użytkownik może odwoływać się do określonego podzbioru informacji; wspomniany aparat programowy powinien przy tym zapewniać niezależność programów od przyjętej w bazie struktury zapisu.

4.3. Programowe aspekty banku danych

Rozważając programowe aspekty banku danych należy wyróżnić trzy kategorie osób profesjonalnie związanych z tym zagadnieniem: programiści - użytkownicy, administratorzy danych oraz konstruktorzy banków danych.

4.3.1. Użytkownicy banków danych

Największą grupę osób związanych z programowaniem stanowią programiści - użytkownicy banków danych. Są oni związani przede wszystkim z następującymi zagadnieniami:

- określaniem struktury danych, przede wszystkim wejściowych i wyjściowych,
- określaniem rodzajów operacji na danych, ewentualnie z określeniem częstotliwości ich występowania,
- stosowaniem w programach haseł umożliwiających dostęp użytkownika do określonych części banku danych (głównie w zakresie bazy danych),
- zabezpieczaniem w ramach dostępnych środków przed nieprawidłowym działaniem wynikającym z jednoczesnego działania dwóch użytkowników z tą samą bazą danych.

Programista-użytkownik nie musi znać struktury zapisu danych w bazie danych ani metod dostępu do tych danych.

4.3.2. Administrator danych

Organizowanie banków danych spowodowało konieczność powstania grupy osób, zwanych administratorami danych, odpowiedzialnych za poprawną eksploatację banków danych. Do zadań administratora danych należy m.in.:

- wyznaczanie obszarów pamięci (przede wszystkim dysków) dla bazy danych,
- wyznaczanie haseł zapewniających ochronę poszczególnych części bazy danych,
- wyznaczanie struktur zapisu danych w bazie danych,
- zakładanie zbiorów oraz przeprowadzanie reorganizacji zbiorów, np. w przypadku przekroczenia dostępnych dla bazy danych obszarów,
- odtwarzanie zbiorów w przypadku ich zniszczenia z powodu awarii.

Podstawowym narzędziem pracy administratora danych jest istniejący w ramach banku danych zestaw programów usługowych (manipulacyjnych).

4.3.3. Konstruktorzy banku danych

Do zadań programistów-konstruktorów banku danych (a ściślej oprogramowania wchodzącego w skład banku danych) między innymi należy zaliczyć:

- określenie dopuszczalnych w bazie danych struktur zapisu oraz metod dostępu do danych,
- określenie i zbudowanie aparatu wiążącego struktury danych ze strukturami zapisu oraz sposoby operowania danymi z metodami dostępu do danych,
- określenie i zbudowanie aparatu ochrony danych przed niepowołanym dostępem oraz przed kolizją wynikającą z działania dwóch użytkowników z tą samą bazą danych.

4.4. Zagadnienia związane z bankiem danych w ujęciu Diebolda

W ostatnim okresie Europejski Program Badawczy Diebolda prowadzi prace nad zintegrowanym systemem informatycznym kierownictwa (IMIS). Jednym z ważniejszych zadań tego systemu jest zaprojektowanie banku danych do celów zarządzania. Publikacje Diebolda [50], [51], [52], dotyczące głównie zagadnień organizacyjnych (stworzenie banku danych i wykorzystanie go przez użytkownika), powstały na podstawie przeglądu literatury, wywiadów z użytkownikami systemów oraz analizy problemów banków danych.

Według oceny Diebolda konstruktor systemu powinien w projekcie przewidzieć rozwiązanie podstawowych problemów, takich jak zróżnicowanie celów użytkowników, trudności wcześniejszego określenia potrzeb i życzeń użytkowników, kodowanie nad-

chodzącej informacji oraz powiązania zawartości i wielkości banku z wymaganymi do przetworzenia danymi. W poszczególnych opracowaniach omawia się organizację banku danych z wieloma wariantami i możliwościami wprowadzenia jej w życie.

Autorzy omawiają także funkcje, jakie bank danych powinien spełniać. System powinien zapewnić uzyskiwanie odpowiedzi na typowe, przewidziane z góry przez konstruktora systemu, pytania zadawane przez użytkowników, jak również na pytania nietypowe, formułowane "ad hoc".

Zakłada się, że konieczny udział użytkownika w obsłudze banku będzie sprowadzony do absolutnego minimum. Do wprowadzenia bardziej złożonych poleceń, bank danych powinien dysponować całym wachlarzem procedur i makrorozkazów. W konkluzji autorzy dochodzą do wniosków, że bank danych zapewni użytkownikowi większą zwięzłość, elastyczność i dostępność. Duży nacisk jest położony na potrzebę uogólnień oraz większe współdziałanie człowiek - maszyna. Nadrzędnym celem projektanta i konstruktora banku danych powinna być wygoda użytkownika, a nie efektywność systemu. Jeśli natomiast dla pewnych typów zagadnień najważniejsza jest wydajność systemu, to do tych celów można stworzyć pewne oddzielne pakiety użytkowe.

Przez wspólną bazę danych określa się (Diebold [52]), tę część systemu IMIS, która zarządza danymi. O zbiorze danych zakłada się, że jest on zorganizowany w ten sposób, że wszystkie istotne dane mogą być w nim umieszczone i odszukane w przeciągu wymaganego okresu czasu. W pracy tej scharakteryzowano bank danych oraz omówiono jego budowę. Zeszyt nr 15 [51] zawiera dość interesujący aneks, przedstawiający dziewięć przebadanych przypadków konstrukcji i użytkowania banku danych. Przedsiębiorstwa należące do różnych gałęzi przemysłu udzieliły odpowiedzi na wiele zapytań dotyczących rozwoju, kosztów, organizacji i oprogramowania ich banku danych. Zebrane odpowiedzi stanowią interesujący materiał porównawczy, dający przyszłym konstruktorom banku możliwość weryfikacji własnych rozwiązań i projektów.

4.5. Ocena propozycji Codasylu dotyczącej DBMS

Celem zapoznania się z pełną propozycją Codasylu dotyczącą DBMS (Data Base Management System - system zarządzania bazą danych) należy odesłać do pozycji Codasyl [39] lub Wrotek [156]. Poniższe uwagi przypominają tylko zasadnicze zagadnienie dotyczące omawianej propozycji.

Propozycja zawiera opisy trzech języków: język opisu danych (tzw. DDL schema), przeznaczony dla administratora danych; język opisu danych dla programu (tzw. DDL subschema), przeznaczony dla projektanta systemu oraz język operowania danymi (tzw. DML), przeznaczony dla programisty. Z tych trzech języków tylko "DDL-schema" jest wstępną propozycją standardu, pozostałe opisy są stosowane do języka COBOL i stanowią raczej ilustrację ogólnych zasad. W podanym opisie takie pojęcia, jak zapis (record), zbiór (set), pole (area) itp. mają zupełnie inne znaczenie niż w rozumieniu dotychczasowym.

Oprócz w miarę szczegółowego opisu wspomnianych trzech języków raport Codasylu wymienia jeszcze dwa języki, potrzebne do objęcia całości zagadnień związanych z bankiem danych, z punktu widzenia użytkownika: język sterowania zadaniami (DMCL) oraz język użytkownika (GL-1). Brak jednak bliższego omówienia tych języków.

Z pojęciem banku danych związane są ściśle następujące zagadnienia:

- organizacja danych,
- metody operowania danymi,
- organizacja zapisu danych,
- metody dostępu do danych.

Propozycja Codasylu omawia tylko dwa pierwsze zagadnienia: omówienie trzeciego zagadnienia ograniczono do opisania realizacji list, drzew i sieci, czwarte zagadnienie pominięto całkowicie. Omawiany dokument jest więc tylko propozycją rozwiązania niektórych aspektów banku danych, nie jest natomiast propozycją banku danych.

Propozycja Codasylu zapewnia możliwość spełnienia wszystkich cech banku danych, omawianych w p. 4.2.1 - 4.2.3. Niepowtarzalność danych można zapewnić przede wszystkim dzięki wyróżnieniu w zbiorach (set) zapisów głównych (owner record) i zapisów - członków (member record) oraz możliwości występowania tego samego zapisu (record) w kilku różnych zbiorach (set) jednocześnie. Korzystanie z tych samych danych przez wielu użytkowników zapewnia fakt, że opis danych (DDL schema) i opis danych dla programu (DDL subschema) mogą obsługiwać kilka programów jednocześnie. Ponadto są w tych językach wyrażenia, które gwarantują ograniczony lub wyłączny (w sensie jednoczesności) dostęp do danych każdemu z programów, co zapewnia możliwość bezkolizyjnej aktualizacji danych przez kilka programów jednocześnie. Wreszcie Codasyl proponuje cały system zabezpieczeń, na poziomie zapisu, zbioru, pola, schematu, podschematu itp., zapewniający ochronę danych przed niepowołanym dostępem. Natomias z propozycji Codasylu nie można wyciągnąć wniosków dotyczących spełnienia postulatu podanego w p. 4.2.4.

Propozycja Codasylu ma jednak sporo istotnych wad, które zostaną poniżej omówione.

4.5.1. Niejednoznaczna interpretacja różnych pojęć

Propozycja Codasylu bardzo często pozostawia interpretację znaczenia różnych pojęć zespołowi realizującemu system. Pozostawienie tak dużej swobody w realizacji systemu budzi obawy, że pojęcie "standardu" będzie nieuzasadnione.

4.5.2. Ograniczenie propozycji do pewnej klasy języków

Całość propozycji ograniczono wyłącznie do języków, w których można zbudować środki do opisu i operowania danymi (tzw. host languages), nie rozważając języków samowystarczalnych (self-contained languages). W ramach tego ograniczenia propozycje dwóch języków (DDL subschema i DML) ograniczono do języka COBOL i wydaje się dość problematyczne opracowanie odpowiednich języków np. dla ALGOL-u lub FORTRAN-u Wreszcie ję-

zyk opisu danych (DDL schema) jest tak podobny do COBOL-u, że można mieć obawy, czy będzie się nadawać do opisu bardziej złożonych organizacji danych.

4.5.3. Brak opisu procedur usługowych

W propozycji jedynie wspomina się o rodzajach potrzebnych procedur usługowych (utilities) nie podając sposobów rozwiązania tego zagadnienia. Szczególnie nie ma mowy o metodach odtwarzania danych po awarii maszyny, ani o metodach reorganizacji zbiorów. Można uznać, że tak ogólne ujęcie tych zagadnień jest związane z brakiem opisu organizacji zapisu danych oraz metod dostępu do danych.

4.5.4. Zależność DDL i DML

Aczkolwiek w opisie propozycji Codasylu wielokrotnie wspomniano, że języki opisu danych (DDL) i operowania danymi (DML) są niezależne, to w rzeczywistości tak nie jest. Co więcej, zależność ta jest obustronna. Jako przykład może tu służyć zdanie w języku DDL: ON OPEN..., które żąda, aby w języku DML istniał rozkaz OPEN. Proponowany język DDL narzuca więc wiele wymagań odnośnie istnienia w języku DML określonych rozkazów (np. OPEN, STORE, GET itd.).

4.5.5. Zależność organizacji danych od zastosowań

Propozycja Codasylu zakłada, że w trakcie opracowania organizacji danych znane są główne metody wykorzystania tych danych. Zupełnie inaczej będą organizowane dane, jeśli podstawową operacją ma być w przyszłości wyszukiwanie, a inaczej - gdy wypisywanie wszystkich danych kolejno. Nie można więc zbudować tego, co się popularnie rozumie pod pojęciem banku danych, określając zawczasu tylko rodzaj danych przechowywanych w tym banku, a nie sposób ich wykorzystania.

4.6. Język optymalizacji danych (DOL) według T. Gilba i jego ocena

Jako kontrpropozycja w stosunku do prac Codasylu, dotyczących koncepcji Data Base Management System został opublikowany projekt języka optymalizacji danych DOL (Gilb [70] i CDI [5]). Projektodawcą jest Tom Gilb, konsultant firmy CDI w dziedzinie EPD.

W projekcie zakłada się, że konstruktor systemu będzie mógł część prac projektowych zlecić maszynie. Autor uważa, iż należy zaniechać tworzenia nowych bardziej skomplikowanych języków programowania, a nawet trzeba dokonać redukcji już istniejących do kilku niezbędnych. W takich warunkach będzie możliwe rozpoczęcie prac nad zaprojektowaniem oprogramowania systemu, mającego zdolność optymalizacji rozmieszczenia plików.

Język optymalizacji danych nie ograniczy się do spełnienia powyższych założeń. W miarę rozwoju powinien on zapewnić optymalizację nie tylko na poziomie pliku, ale również takich własności systemu, jak szybkość dostępu, gospodarka pamięcią, itp. Przykładem dalszego rozwoju byłoby zapewnienie optymalizacji przenoszalności (portability) między dwoma systemami.

Język DOL będzie mógł składać się z dwóch różnych części stosowanych na różnych poziomach banku danych:

- język dyrektywny - dający priorytet optymalizacji pewnych własności banku danych. Korzystać z tej formy będzie zarówno projektant, nakładając pewne warunki na uruchamiany system, jak też administrator banku danych narzucając wskaźniki optymalizacji,
- język analizy danych, korzystający z informacji zgromadzonej przez sam system, np. w postaci prób statystycznych. Zbierany materiał (liczba odwołań do danych, plików, itd.) będzie podstawą dalszej optymalizacji.

Według oceny autora główną zaletą tej idei jest zmniejszenie potrzeb związanych ze szkoleniem kadr programistów i projektantów, większa swoboda w wyborze sprzętu oraz zapewnienie wyższego stopnia logicznej niezawodności programów, z możliwością wyszukiwania błędów.

DOL ma być innym językiem niż pozostałe języki programowania. Ważną cechą będzie to, że system może korzystać z DOL-u lub też go ignorować. Jest to konieczne, gdyż może się zdarzyć, że cel optymalizacji będzie sprzeczny z innymi zadaniami systemu, czy też optymalizacja żądanej własności będzie w danym systemie nieopłacalna lub niewykonalna. Przykładem może być próba optymalizacji czasu dostępu do danych przez grupę użytkowników, podczas gdy administrator banku danych narzucił optymalizację rozmieszczenia danych w pamięci.

W początkowym okresie DOL będzie zawierał tylko kilka prostych instrukcji.

Przedstawione powyżej założenia dla języka DOL określają przede wszystkim zadania przed nim stojące. Jak dotychczas DOL nie wszedł jeszcze w fazę realizacji. Szczegółową ocenę przydatności języka będzie można przeprowadzić po próbnych eksploatacjach w konkretnych systemach banków danych.

5. ZAGADNIENIA FORMALIZACJI

W niniejszym rozdziale omówione zostaną pewne próby formalizacji pojęć i czynności związanych ze strukturami danych. Próby te poszły w dwu zasadniczych kierunkach.

Z jednej strony stosuje się metody (pojęcia) matematyczne do interpretacji i konstrukcji stosowanych i nowotworzonych w praktyce metod (pojęć). Prace te mają charakter raczej przyczynkowy. Mimo, że ukazują się w wielkiej liczbie i zawierają coraz to nowe pomysły, ich pojawienie się nie wpływa w istotny sposób na stosowaną praktykę.

Z drugiej strony buduje się formalne modele reprezentujące pojęcia podstawowe. Wydaje się, że podejście to, w obecnej swojej formie dalekie jeszcze od doskonałości, spełni ważną rolę w dziedzinie komputerów. Są to być może zalążki nowych teorii związanych z komputerami; ta klasa zagadnień zostanie przez nas potraktowana bardziej szczegółowo. Poniżej omawiamy wybrane prace na ten temat.

5.1. Algebra informacyjna Bosaka

Źródłem poniższego opisu jest raport CODASYL Development Committee - An Information Algebra [17] . Koncepcja algebry informacyjnej została opracowana w latach 1959 - 1961 i ogłoszona w 1962 r. przez grupę specjalistów amerykańskich na podstawie propozycji Bosaka. Jest ona próbą formalizacji pojęć stosowanych w zagadnieniach przetwarzania informacji do celów gospodarczych.

Mimo, że upłynęło 11 lat od opublikowania raportu i mimo, że raport ten jest cytowany przez wielu autorów, sama koncepcja algebry informacyjnej nie została rozwinięta.

Algebra informacyjna jest oparta na trzech podstawowych pojęciach: jednostki (e), wartości (v), cechy (q). Te trzy pojęcia są opisane następującymi postulatami:

Postulat 1. Z każdą cechą związany jest jeden i tylko jeden zbiór wartości cechy.

Postulat 2. Do każdego zbioru V należą conajmniej dwa elementy Ω (nieokreślony) i Θ (określony, ale nie znany - brak informacji).

Postulat 3. Każdej jednostce jest przypisana jedna i tylko jedna wartość ze zbioru wartości każdej cechy.

Na podstawie powyższych pojęć wprowadza się kolejne definicje; niektóre z nich zostaną poniżej przytoczone.

Definicja
Przestrzenią cech P nazwiemy iloczyn kartezjański
$P = V_1 \times V_2 \times \ldots \times V_n$, V_i jest zbiorem wartości i-tej cechy.

Definicja
Punktem danych d jednostki e w przestrzeni P nazywamy taki punkt przestrzeni P, którego współrzędna i jest wartością cechy i przypisanej jednostce e; d nazywamy reprezentacją jednostki e w przestrzeni P.

Tak wprowadzone pojęcie punktu danych odpowiada dokumentowi, natomiast poszczególne jego współrzędne - pozycjom tego dokumentu.

Definicja
Uporządkowane (wg jednej z przyjętych definicji porządku) zbiory punktów z przestrzeni cech P noszą nazwę linii. Liczbę punktów wchodzących do linii nazwiemy rozpiętością linii.

Definicja
Dowolny zbiór w przestrzeni P nazwiemy obszarem. Obszar odpowiada więc przyjętemu w teorii struktur danych pojęciu pliku, tj. zbioru dokumentów, natomiast linia jest uporządkowanym plikiem. Rozpiętość linii jest liczbą dokumentów w pliku.

W dalszym ciągu wprowadza się pojęcia funkcji linii i funkcji obszaru. Pojęcia te odpowiadają działaniom na plikach. Następujące twierdzenie ustala związek między tymi pojęciami:

Twierdzenie
Jeżeli M jest linią zawierającą wszystkie punkty obszaru B i żadnych innych, to dla dowolnej funkcji obszarów f' istnieje funkcja linii f'' taka, że $f'(B) = f''(M)$.

Ponieważ każdej linii odpowiada pewien obszar złożony z punktów tej linii, powyższe twierdzenie można interpretować następująco: dla każdego pliku istnieją działania niezależne od porządku dokumentów w tym pliku.

Następnie wprowadza się pojęcia wiązki (grupa plików) i funkcji wiązek (działania tworzące nowe pliki) oraz pojęcie uporządkowania obszaru, odpowiadające czynności sortowania.

W artykule Bosaka [17] przytoczone powyżej pojęcia są uzupełnione innymi pojęciami i poparte wieloma przykładami.

5.2. Zarys jednolitej teorii struktur danych Turskiego

Praca W. Turskiego [146] została ogłoszona w 1971 r. Jest to koncepcja ogólniejsza od poprzedniej, dotyczy ona struktur danych występujących we wszelkich możliwych zastosowaniach maszyn cyfrowych.

W omawianej teorii podstawowymi pojęciami są:

$\mathcal{N}$ - zbiór przeliczalny, zwany zbiorem nazw
! - element taki, że $! \in \mathcal{N}$; element ! zwany jest znakiem końca
σ - funkcja określona na zbiorze $2^{\mathcal{N}}$ o wartościach w $\mathcal{N} \cup \{!\}$, taka, że dla $N \subset \mathcal{N}$ i $N \neq \emptyset$ zachodzi $\sigma(N) \in N$; funkcja σ zwana jest selektorem naturalnym.
$\mathcal{V}$ i $\mathcal{V}\mathcal{V}$ - dwa zbiory takie, że $\mathcal{V}\mathcal{V} \subset \mathcal{V}$; zbiór $\mathcal{V}$ jest nazywany zbiorem wartości, a zbiór $\mathcal{V}\mathcal{V}$ - zbiorem wartości prostych.

Po wprowadzeniu tych pojęć można wprowadzić definicje:

Definicja
Przestrzenią nazw nazywamy trojkę $\mathcal{N} = (\mathcal{N}, \sigma, !)$

Definicja
Przestrzeń nazw $\mathcal{N} = (\mathcal{N}, \sigma, !)$ nazwiemy systematyczną, jeśli dla dowolnych dwóch zbiorów $N, M \subset \mathcal{N}$ zachodzi związek

$$\sigma(N \cup M) = \sigma(\{\sigma(N), \sigma(M)\})$$

W przestrzeni systematycznej selektor naturalny wyznacza więc jednoznacznie uszeregowanie elementów zbioru nazw; uszeregowanie to jest nazywane naturalnym.

Definicja
Podzbiór nazw $N \subset \mathfrak{N}$ z przestrzeni $\mathfrak{N}$ nazwiemy gęstym, jeśli z tego że $n', n'' \in N$ wynika, że należą do N wszystkie nazwy leżące między n' i n'' w uszeregowaniu naturalnym.

Podstawowym dla wprowadzanej teorii struktur danych pojęciem jest dana.

Definicja
Dana d jest parą (n, v), gdzie $n \in \mathfrak{N}$, $v \in \mathfrak{V}$; jeśli $v \in \mathfrak{V}$ to daną nazywamy prostą lub rangi 1. Jeśli d = (n, v) to będziemy również pisać $n = \vartheta(d)$, $v = \beta(d)$.

Definicja
Strukturą danych nazywamy skończony zbiór danych S taki, że jeśli $d_1, d_2 \in S$ oraz $\vartheta(d_1) = \vartheta(d_2)$ to $\beta(d_1) = \beta(d_2)$.

W strukturze danych każde dwa elementy mają więc różne nazwy. Interpretacją struktury może być zarówno pozycja, jak grupa, dokument, plik, grupa plików itd. Struktura złożona z jednej danej prostej odpowiada pozycji elementarnej.

Rozwinięciem tych pojęć jest wprowadzenie klasyfikacji struktur, tj. zdefiniowanie struktur półregularnych, regularnych i nieregularnych. Struktury półregularne, tj. takie, które są związane z systematycznymi przestrzeniami nazw, odpowiadają najczęściej występującym w zagadnieniach przetwarzania danych obiektom, takim jak spisy, zestawienia itp. Przykładem interpretacji struktur regularnych, które charakteryzują się pewnego rodzaju podobieństwem swoich podstruktur są macierze.

Jeszcze jednym pojęciem, o którym tu warto wspomnieć jest rozmieszczenie struktur danych w ośrodkach przechowywania danych. Pojęcie to odpowiada opisanym w poprzednich rozdziałach związkom pomiędzy plikiem i zbiorem, dokumentem i zapisem itd.

Czytelnika zainteresowanego rozwinięciem tych pojęć oraz wprowadzeniem innych, takich jak jednoznaczność struktury,

ranga struktury, przestrzeń strukturalna itp. odsyłamy do pozycji [146] lub [144] i [145].

5.3. Relacyjny model danych Codda

Relacyjny model danych zaproponowany przez Codda [29] oparty jest na matematycznym pojęciu relacji (podzbiór iloczynu kartezjańskiego zbiorów). Autor nie używa pojęcia dokumentu i pliku, wydaje się jednak, że element relacji czyli każdą n-tkę można interpretować jako dokument, zaś relację jako grupę dokumentów tego samego formatu. Liczba zbiorów (zwanych domenami) określa stopień relacji. Działaniom na plikach odpowiadają wprowadzone przez autora operacje na relacjach. Domenę, która jest relacją nazywamy domeną nieprostą, w przeciwnym przypadku domena jest prosta.

Ponieważ najwygodniejszą relacją jest relacja ze wszystkimi domenami prostymi autor podaje metodę normalizacji relacji (tzn. metodę otrzymywania ze zbioru relacji z nieprostymi domenami zbiór relacji z domenami prostymi) i podaje warunki możliwości wykonania takiego przekształcenia. W pracy zdefiniowane jest pojęcie redundancji silnej i słabej oraz omówiono zalety i wady występowania każdej z nich w banku danych. Pojęcie to, którego interpretacja jest oczywista służy do zdefiniowania i omówienia zagadnienia utrzymania danych w stanie zwartym.

5.4. Interpretacja danych wg Mealy

G.H. Mealy w referacie wygłoszonym w 1967 r. na konferencji Fall Joint Computer [111] przedstawił propozycję teoretycznego modelu systemów przetwarzania danych. Podstawą tej propozycji jest wyróżnienie trzech dziedzin: świata rzeczywistego, idei o tym świecie istniejącej w umyśle człowieka oraz symboli (na papierze lub innych nośnikach pamięci) opisujących te idee.

Autor wprowadza następujące pojęcia:

E - zbiór jednostek,
V - zbiór wartości; dane
D - zbiór odwzorowań postaci

$$\mu : E \longrightarrow V$$

D jest więc zbiorem, którego elementami są podzbiory zbioru E x V, tzn. $D \subset 2^{(ExV)}$

W zbiorze D wyróżnia się podzbiór odwzorowań strukturalnych, które definiuje się jako odwzorowania postaci:

$$\delta : E \longrightarrow E$$

Innymi słowy odwzorowania strukturalne przekształcają zbiór jednostek w siebie.

Oprócz powyższych pojęć autor wprowadza jeszcze zbiór P definiując go następująco:

P - zbiór procedur, którego elementami są odwzorowania postaci

$$\pi : 2^{(ExV)} \longrightarrow 2^{(ExV)}$$

Na podstawie powyższych pojęć autor definiuje pozycję danych i element danych:

- pozycją danych jest uporządkowana para (e, v)
- elementem danych jest zbiór wszystkich pozycji danych związanych z daną jednostką e.

Powyższe definicje są dość zbliżone do definicji Turskiego omówionych w p. 5.2. Interpretacja tych pojęć jest również oczywista. Czwórkę uporządkowaną S = (E, V, D, P) autor w dalszym ciągu nazywa systemem. Dla dwóch danych systemów S i S' zawsze istnieje przekształcenie φ, takie

$$\varphi : S \longrightarrow S'$$

Jeśli S jest systemem przedstawiającym pewną ideę o części świata rzeczywistego, a S' jest systemem przedstawiającym reprezentację maszynową systemu S, to φ nosi nazwę reprezentacji.

Przedstawione przez Mealy'ego koncepcje są punktem wyjścia dla innych autorów, w szczególności rozwinęli te koncepcje Engles [49], Senko i inni [125] oraz Vuillemin [147].

5.5. Propozycja miar danych według Gilba

Poniżej omówiono pracę T. Gilba dotyczącą prób formalizacji oraz wprowadzania miar do pojęć z zakresu systemów EPD. Sama koncepcja wprowadzenia takiej formalizacji wydaje się pomysłem ciekawym, jednakże podane przez Gilba definicje są bardzo mało precyzyjne. Poniżej przytoczono niektóre z tych definicji po to, aby Czytelnik mógł je ocenić.

W pracy pod tytułem: "Datametrics Handbook" [71] autor zaproponował wprowadzenie miar zarówno dla danych jak i dla pojęć z nimi związanych. Gilb zastrzega, że jest to pierwsza próba takiej systematyki. Opis miar ma stanowić materiał do dalszej dyskusji nad samym pomysłem. Praca nie jest jeszcze ukończona, omówiono w niej tylko część propozycji.

Zaprezentowane miary można podzielić na cztery grupy: miary dotyczące niezawodności danych, elastyczności danych, budowy strukturalnej danych oraz pojęć podstawowych.

Wśród miar związanych z niezawodnością danych autor wprowadza następujące:

a) redundancja - określona jako stosunek aktualnie używanej długości danej do minimalnej długości,

b) automatyczne wyszukiwanie (poprawianie) danej - prawdopodobieństwo wyszukania (poprawienia) danej za pomocą pewnego algorytmu,

c) niezawodność - prawdopodobieństwo, że system będzie działał sprawnie w pewnym okresie czasu,

d) prawdopodobieństwo ataku - prawdopodobieństwo ataku na system w pewnym okresie czasu,

e) błąd - ogólna liczba błędów.

Oprócz wymienionych pojęć, autor do tej grupy zalicza i określa jeszcze inne pojęcia, takie jak: precyzja, tolerancja, prawdopodobieństwo otwarcia ataku, dostępność itp.

Wśród drugiej grupy miar najbardziej charakterystyczne są:

a) złożoność - liczba sytuacji w systemie, gdzie należy podjąć decyzję TAK - NIE (warunki IF),

b) elastyczność - użyteczna złożoność - praktyczna lub potencjalna,

c) przenaszalność - określona jest jako $1 - \frac{\text{koszt przeniesienia}}{\text{ogólny koszt systemu}}$

Trzecia grupa to miary dotyczące budowy strukturalnej danych:

a) hierarchia - miara hierarchii, może być liczbą poziomów, liczbą wszystkich wierzchołków, itp.

b) złożoność strukturalna - stosunek powiązań między modelami do ogólnej liczby modułów (podsystemów),

c) modularność - liczba modułów w systemie.

Wreszcie miary dla pojęć podstawowych określa autor następująco:

a) informacja - interpretowana przez daną,
b) dana - mierzona w bitach lub symbolach,
c) ewolucja - stopień zmian w systemie w pewnym okresie czasu (zmienia się od 0 do 1).

6. TERMINOLOGIA

1. ADMINISTRATOR DANYCH (data administrator)
Osoba lub grupa osób zapewniających ciągłą eksploatację bazy danych

2. BANK DANYCH (data bank)
Baza danych wraz z oprogramowaniem; oprogramowanie powinno w szczególności zapewnić możliwość jednoczesnego korzystania z bazy danych wielu użytkownikom oraz niezależność programów od struktury zapisu

3. BAZA DANYCH (data base)
Zbiór zapisany w pamięci maszyny w ten sposób, że każda dana występuje raz; w bazie danych muszą być wyróżnione podzbiory (niekoniecznie rozłączne) danych równoważnych z punktu widzenia ich tajności

4. BLOK (block)
Jednostka wymiany między pamięcią główną a pomocniczą

5. DANA (datum)
Para (nazwa, wartość), gdzie wartością może być stała, dana lub struktura

6. DANA PROSTA
Dana, której wartością jest stała

7. DOKUMENT (logical record, entry)
Dana, której wartością jest zbiór pozycji i/lub grup

8. DOSTĘPNOŚĆ (privacy)
Zespół środków umożliwiających użytkownikowi dostęp do wybranych danych, procedur itp.
(zob. też ZABEZPIECZENIE)

9. ELEMENT ZAPISU (data element, field)
Jednostka pamięci głównej pobierana lub pamiętana w czasie jednego cyklu. Może być bitem, bajtem, słowem

10. GŁÓWNA METODA DOSTĘPU
Metoda dostępu wyznaczająca organizację zbioru danych

11. GRUPA, POZYCJA ZŁOŻONA (group, data aggregate)
Dana, której wartością jest zbiór pozycji i/lub grup

12. IDENTYFIKATOR (identifier)
Klucz dokumentu jednoznaczny w pliku

13. KLUCZ DOKUMENTU (record key)
Jedna z pozycji w dokumencie

14. KLUCZ PORZĄDKOWANIA (major key)
Klucz dokumentu wyznaczający miejsce dokumentu w pliku

15. KLUCZ WYSZUKIWANIA (minor key)
Klucz dokumentu służący do odszukania dokumentu w pliku

16. METODA DOSTĘPU (access method)
Algorytm znajdowania zapisu lub miejsca na zapis. Do najbardziej znanych metod dostępu należą metody dostępu: sekwencyjnego (sequential), indeksowo-sekwencyjnego (index-sequential), bezpośredniego (random)

17. OCHRONA (protection)
Zespół środków zabezpieczający przed przypadkowym lub umyślnym zniszczeniem zbioru danych, procedur itp. oraz przed niepowołanym do nich dostępem
(zob. też ZABEZPIECZENIE)

18. OPEROWANIE DANYMI
Wykonywanie operacji na danych. Operacje na danych można podzielić na dwie klasy: operacje użytkownika oraz operacje na zbiorach danych

19. ORGANIZACJA PLIKU (file organization)
Powiązania w pliku (np. między dokumentami lub grupami) deklarowane przez użytkownika

20. ORGANIZACJA ZBIORU DANYCH (file organization, data set organization)
Powiązania między obrazami dokumentów w pamięci wyznaczone przez główną metodę dostępu. Najbardziej znane organizacje: sekwencyjna (sequential), indeksowo-sekwencyjna (index-sequential), bezpośrednia (random lub direct) inaczej zwana organizacją o dostępie obliczanym (computed access), dowiązywana (chained) inaczej zwana sieciową (network), odwrócone: częściowo i całkowicie (partial i full inverted) oraz grupowa (clustered).

21. PLIK (file)
Dana, której wartością jest zbiór dokumentów

22. PLIK FORMATOWY (formatted file)
Plik, w którym dokumenty mają takie same nazwy pozycji; pozycje występują w tej samej kolejności, a odpowiadające sobie pozycje mają wartości o takiej samej długości

23. PLIK NIEFORMATOWY (unformatted file)
Plik, w którym dokumenty mogą mieć różną liczbę pozycji

24. PLIK PÓŁFORMATOWY (semiformatted file)
Plik różniący się od pliku formatowego tym, że wartości pozycji mają nieokreśloną (zmienną) długość

25. POZYCJA, POZYCJA ELEMENTARNA (data item, data element)
Dana prosta, podstawowa jednostka w przetwarzaniu danych

26. PRZENASZALNOŚĆ (portability)
Możliwość przenoszenia oprogramowania (danych) między dwoma systemami

27. RELACJA SORTUJĄCA
Relacja spójna, zwrotna i przechodnia

28. STRUKTURA DANYCH (data structure)
Zbiór danych i/lub stałych

29. STRUKTURA ZAPISU (storage structure)
Obraz struktury danych w pamięci maszyny

30. TOM (volume)
Zbiór bloków dostępny z urządzenia we-wy w pojedynczym ładowaniu; może zawierać część zbioru danych, bądź jeden lub kilka zbiorów danych

31. WARTOŚĆ PROSTA, STAŁA (simple value, constant)
Wartość niepodzielna w języku (nie zawierająca żadnych innych danych)

32. ZABEZPIECZENIE (security)
Zespół środków ograniczających operowanie danymi, procedurami itp. do określonego kręgu użytkowników. Zabezpie-

ozenie składa się z ochrony i dostępności
(zob. też DOSTĘPNOŚĆ I OCHRONA)

33. ZAPIS (physical record)
Zbiór powiązanych ze sobą elementów zapisów, który można zidentyfikować, najczęściej obraz dokumentu

34. ZBIÓR DANYCH (data set)
Zbiór bloków mających wspólną nazwę, najczęściej obraz pliku

Bibliografia

[1] ABRAHAM C.T., BOSE R.C., GHOSH S.P.: File Organization of Records with Multiple Valued Attributes for Multi-Attribute Queries, IBM Research Report, 1967, s. 1886.

[2] ABRAHAM C.T., GHOSH S.P., RAY-CHAUDHURI D.K.: File Organization Schemes Based on Finite Geometries, Inf. Contr. 12, luty 1968, s. 143-163.

[3] ABRIAL J.R.: Data Semantics, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974

[4] ADELSON-VELSKIJ G.M., LANDIS E.M.: An Information Organization, Algorithm, DANSSSR 146, 1962, s. 266.

[5] Advanced Data Base. Seminarium Control Data Institute. Materiały seminaryjne, Katowice 1973.

[6] AHRENS F., WALTER H.: Datenbanksysteme, W. de Gruyter, Berlin-New York, 1971.

[7] ALTMAN E.B., ASTRAHAN M.M., FEHDER P.L., SENKO M.E.: Specifications in a Data Independent Architectural Model, Proceedings of the ACM SIGFIDET Conference, Denver, Colorado 1972.

[8] ASTRAHAN M.M., ALTMAN F.B., FEHDER P.L., SENKO M.E.: Concepts of a Data Independent Architectural Model, Proceedings of the ACM SIGFIDET Conference, Denver, Colorado 1972.

[9] BARNES R.F.: Mathematico-Logical Foundations of Retrieval Theory, General Concepts and Methods. Bethlehem, Pa. Center for the Inform. Sciences, Lehigh Univ. 1965.

[10] BAYER R.: Symmetric Binary B-trees: Data Structure and Maintenance Algorithms, Acta Informatica, 1972, t. 1, s. 290-306.

[11] BAYER R., Mc CREIGHT E.: Organization and Maintenance of Large Ordered Indexes. Acta Informatica, 1972, t. 1, s. 173-189.

[12] BEKIĆ H., WALK K.: Formalization of Storage Properties, Symposium on Semantics of Algorithmic Languages, Springer Lecture Note Series, Springer-Verlag, 1971.

[13] BELL C.J.: A Relational Model for Information Retrieval and the Processing of Linguistic Data, IBM Research Report RC 1705, Yorktown Heights, New York, 1966.

[14] BERUL L.: Document Retrieval, Annual Review of Information Science and Technology, J. Wiley, 1969, t. 4, s. 203-227.

[15] BERZTISS A.T.: Data Structures: Theory and Practice. Acad. Press, New York 1971.

[16] BLEIER R.E.: Treating Hierarchical Data Structures in the SDC Time Shared Data Management System TDMS, Proceedings of the ACM. 22nd National Conference. MDI Publications, Wayne, Pensylwania, 1967, s. 41-49.

[17] BOSAK R. i in.: An Information Algebra. Communications of the ACM, 1962, t. 5, s. 190-204.

[18] BOULLIER P., JANCENE P.: LAMBDA Language pour la Manipulation d'une Base de Données Associative, Rapport de Recherche nr 25, Inst. RIA, 1973.

[19] BRACCHI G., FEDELI A., PAOLINI P.: A Relational Data Base Management System, Proceedings of the ACM 25th Annual Conference, Boston, 1972.

[20] BRACCHI G., FEDELI A., PAOLINI P.: A Language for a Relational Data Base Management System, Proc. of the 6th Annual Princeton Conference on Information Sciences and Systems, marzec 1972.

[21] BRACCHI G., FEDELI A., PAOLINI P.: The Architecture of an Online Information Management System, Proceedings of the ONLINE 72 International Conference, Brunel University, Uxbridge, Anglia, wrzesień 1972.

[22] BRACCHI G., FEDELI A., PAOLINI P.: A Multilevel Relational Model for Data Base Management Systems, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[23] BUCHHOLZ W.: File Organization and Addressing, IBM Systems Journal, 1963, t. 2.

[24] CHILDS D.C.: Feasibility of a Set Theoretic Data Structure, Proceedings of the IFIP Congress 1968, Amsterdam 1968, s. 420-430.

[25] CHILDS D.L.: Description of a Set-Theoretic Data-Structure, Tech. Rept. CONCOMP proj. University of Michigan, 1968.

[26] CHOW D.K.: New Balanced-File Organization Schemes, Information Control, 1969, t. 15, s. 377-396.

[27] CODASYL - patrz [39], [56] oraz [138].

[28] CODD E.F.: Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks, IBM Research Report, RJ 599, 1969.

[29] CODD E.F.: A Relational Model for Large Shared Data Banks, Communications of the ACM, czerwiec 1970, t. 13, nr 6, s. 377.

[30] CODD E.F.: A Data Base Sublanguage Founded on the Relational Calculus, Proc. 1971 ACM-SIGFIDET Workshop, San Diego, 1971.

[31] CODD E.F.: Relational Completeness of Data Base Sublangueges, Courant Computer Symposia, "Data Base Systems", Prentice Hall, New York, maj 1971, t. 6.

[32] CODD E.F.: Further Normalization of the Data Base Relational Model, Courant Computer Science Symposia: "Data Base Systems", Prentice Hall, New York, 1971, t. 6.

[33] CODD E.F.: Further Normalization of the Data Base Relational Model, IBM Research, wrzesień 1971.

[34] CODD E.F.: The Relevance and Significance of the Relational Approach, Symposium on "Relational Data Base Concepts", British Computer Society, kwiecień 1973.

[35] CODD E.F.: Seven Steps to Randez-vous with the Casual User, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[36] COLLMEYER A.J.: Implications of Data Independence on the Architecture of Data Base Management Systems, Proceedings of 1972 ACM-SIGFIDET Workshop, 1972.

[37] CONWAY R.W., MAXWELL W.L., MORGAN H.L.: On the Implementation of Security Measures in Information System, Communications of the ACM, 1972, t. 15, s. 211-220.

[38] DALEY R.C., NEUMANN P.G.: A General Purpose File System for Secondary Storage, AFIPS Conf. Proc., FJEC 1965, t. 27, s. 213-229.

[39] Data Base Task Group Raport to the CODASYL Programming Language Comm., kwiecień 1971.

[40] DATE C.J., HOPEWELL P.: File Definition and Logical Data Independence, Proceedings of the 1971 ACM-SIGFIDET Workshop, 1971.

[41] DAVIES C.T.: A Logical Concept for the Control and Management of Data, Report R-0803-00, IBM Corporation, New York 1967.

[42] DEARNLEY P.A.: A Model of Self-Organizing Data Management System, The Computer Journal, 1973, t. 16, nr 4.

[43] DEARNLEY P.A.: The Operation Model Self-Organizing Data Management System, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[44] DODD G.G.: Elements of Data Management Systems, Comp. Surveys, czerwiec 1969, t. 1, s. 117-133.

[45] DURCHHOLZ, RICHTER: Das Datenmodell der "Feature Analysis of Generalized Data Base Management Systems", Angewandte Informatik, grudzień 1976, nr 12.

[46] DURCHHOLZ, RICHTER: Concepts for Data Base Management Systems, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[47] EARLEY J.: Towards an Understanding of Data Structures, Communications of the ACM, 1971, t. 14, nr 10, s. 617-628.

[48] EARLEY J.: Relational Level Data Structures for Programming Languages, Acta Informatica, 1973, t. 2, nr 4.

[49] ENGLES R.W.: A Tutorial on Data-Base Organization, Annual Review in Automatic Programming, Pergamon Press, 1972, t. 7, nr 1, s. 1-64.

[50] Europejski Program Badawczy Diebolda, Bank danych, z. 3, Warszawa 1969.

[51] Europejski Program Badawczy Diebolda, Bank danych II, z. 15, Warszawa 1971.

[52] Europejski Program Badawczy Diebolda, Wspólna baza danych III, z. 32, Warszawa 1972.

[53] Europejski Program Badawczy Diebolda, Programy operowania danymi, z. 31, Warszawa 1973.

[54] EVEREST G.C.: Concurrent Update Control and Data Base Integrity, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[55] FALKENBERG E. i in.: Result-Oriented Manipulation of Data Systems, Report University of Stuttgart, Institute of Informatics, 1973.

[56] Feature Analysis of Generalized Data Base Management Systems, CODASYL Systems Committee Technical Report, kwiecień 1971.

[57] FILLAT A.I., KRONING L.A.: Generalized Organization of Large Data Bases; a Set Theoretic Approach to Relations, MIT, Cambridge (Mass.), Project MAC, MAC-TR-70, 1970.

[58] FINKEL R.A., BENTLEY J.L.: Quad Trees - a Data Structure for Retrieval on Secondary Keys, Stanford University.

[59] FLORES I.: Computer Sorting, Prentice-Hall, 1969.

[60] FLORES I.: Data Structure and Management, Prentice-Hall, Inc., 1970.

[61] FLORES I., MADPIS G.: Average Binary Search Length for Dense Ordered Lists, Communications of the ACM, 1971, t. 14, s. 601.

[62] FOSSUM: Data Base Integrity as Provided for by a Particular Data Base Management System, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[63] FOSTER C.C.: Information Storage and Retrieval Using AVL Trees, Proc. ACM, 20th National Conf., 1965, s. 192-205.

[64] FRASER A.G.: Integrity of a Mass Storage Filing System, The Computer Journal, 1969, t. 12, s. 1-5.

[65] FRASER A.G.: On the Meaning of Names in Programming Systems, Communications of the ACM, 1971, t. 14, nr 6.

[66] FRAZER W.D., WENG C.K.: Sorting by Natural Selection, Communications of the ACM, 1972, t. 15, s. 910-912.

[67] GHOSH S.P., ABRAHAM C.T.: Application of Finite Geometry in File Organization for Records with Multiple-Valued Attributes, IBM Journal of Research and Development, 1968, t. 12, nr 2.

[68] GHOSH S.P.: File Organization: The Consecutive Retrieval Property, Communications of the ACM, wrzesień 1972, t. 15, nr 9.

[69] GHOSH S.P.: On the Problem of Query-Oriented Filing Schemes Using Discrete Mathematics, Proc. IFIP Congress, Edynburg 1968, s. 74-79.

[70] GILB T.S.: Data Base Software, Control Data Institute, 1973.

[71] GILB T.S.:Datametrics Handbook, Control Data Institute, 1973.

[72] GOLDSTEIN R.C., STRNAD A.L.: The Mac AIMS Data Management System, Proc. 1970 ACM SIGFIDET Workshop, listopad 1970.

[73] HARRISON M.C.: Data Structures and Programming, Courant Inst. of Math. Sciences, New York Univ., New York 1970.

[74] HENRY W.R.: Hierarchical Structure for Data Management, IBM Syst. Journal, 1969, t. 8, s. 2-15.

[75] HIBBARD T.N.: Some Combinatorial Properties of Certain Trees with Applications to Searching and Sorting, Journal of ACM, 1962, t. 9, s. 13-28.

[76] HOARE C.A.R.: Data Structures in Two-Level Storage, Inf. Processing 68, North-Holland, Amsterdam, 1969.

[77] HOARE C.A.R.: Notes on Data Structuring, Structured Programming by Dahl, Dijkstra, Hoare, Acad. Press, 1972.

[78] HSIAO D., HARARY F.: A Formal System for Information Retrieval from Files, Communications of the ACM, 1970, t. 13, s. 67-73.

[79] HU T.C.: A Comment on the Double Chained Tree, Communications of the ACM, 1972, t. 15, s. 276.

[80] HU T.C., TAN K.C.: Least Upper Bound on the Cost of Optimum Binary Search Trees. Acta Informatica, 1972, t. 1, s. 307-310.

[81] HU T.C., TAN K.C.: Path Length of Binary Search Trees, Rep. 1111. Math. Res. Center Univ. of Wisconsin, marzec 1971.

[82] HU T.C., TUCKER A.C.: Optimum Binary Search Trees. Rep. 1049. Math. Res. Center Univ. of Wisconsin, marzec 1970.

[83] HWANG F.K., LIN S.: Optimum Merging of 2 Elements with n Elements, Acta Informatica, 1971, t. 1, s. 145-158.

[84] IFIP, File Organization, Selected Papers from File 68 - an IAG Conference, Swets and Zeitinger, N.V., Amsterdam, 1969.

[85] D'IMPERIO, M.E.: Data Structures and Their Representation in Storage. Annual Review in Automatic Programming, 1969, t. 5.

[86] JANKO W.: Das Aufsuchen von Tabellenelementen bei verketter Speicherung mittels statistischer Methoden, Angewandte Informatik, 1973, nr 3, s. 114-122.

[87] JERVIS B., PARKER J.L.: An Approach for a Working Relational Data System, Proc. 1972 ACM SIGFIDET Workshop, listopad 1972.

[88] JOHNSON L.R.: Systems Structure in Data, Programs and Computers, Prentice-Hall, Englewood Cliffs, New York 1970.

[89] Joint Guide-share, Data Base Management System Requirements, W.D. Stevens, Skelly Oil Co., Tulsa. Oklahoma 74102, 1970.

[90] KALINICHENKO L.A., RYVKIN V.M.: Problems of High Level Data Base Access Language Implementation in Inverted Storage Structure Environement, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[91] KELLY J.F.: Computerized Management Information Systems, Chapter 8.533, The Macmillan Company, New York, 1970.

[92] KENNEDY S.: A Note on Optimal Double-Chained Trees, Communications of the ACM, 1972, t. 15, s. 997-998.

[93] KENNEDY S.R.: Optimal Weighted Double Chained Trees. Inform. Science Tech. Rept. No 1. California Institute of Technology, Pasadena, Kalifornia, maj 1972.

[94] KNUTH D.E.: The Art of Computer Programming, t. 1 (Fundamental algorithms), t. 3 (sorting and searching).

[95] KNUTH D.E.: Length of Strings for a Merge Sort. Communications of the ACM, 1963, t. 6, s. 685-687.

[96] KNUTH D.E.: Optimum Binary Search Trees. Acta Informatica, 1971, t. 1.

[97] KOOPMAN B.O.: The Theory of Search: III. The Optimum Distribution of Searching Effort, Opns. Res. 5, 1957, s. 613-626.

[98] KURKI-SUNIO R.: Formal Description of Input Data, Inf. Processing 68, North-Holland, Amsterdam, 1969.

[99] LANDAUER W.I.: The Balanced Tree and Its Utilization in Information Retrieval, IEEE Transactions on Electronic Computers, 1963, t. EC-12.

[100] LEFKOVITZ D.: File Structures for On-line Systems. J. Wiley, New York 1967.

[101] LEVIEN R.E., MARON M.E.: A Computer System for Inference Execution and Data Retrieval, Communications of the ACM, listopad 1967, t. 10.

[102] LIN A.D.: Key Addressing of Random Access Memories by Radix Transformation. Proc. AFIPS, 1963, SJCC, t. 23, s. 355-366.

[103] LORIE R.A., SYMONDS A.J.: A Schema for Describing a Relational Data Base, Proc. 1970 ACM SIGFIDET Workshop, Houston, listopad 1970.

[104] LUM V.Y.: Multiattribute Retrieval with Combined Indexes, Communications of the ACM, 1970, s. 660-666.

[105] LUM V.Y., YUEN P.S.T.: Additional Results on Key-to-address Transform Techniques; a Fundamental Performance Study on Large Existing Formatted Files. Communications of the ACM, 1972, t. 15, s. 996.

[106] LUM V.Y., YUEN P.S.T., DODD M.: Key-to-address Transform Techniques; a Fundamental Performance Study of Large Existing Formatted Files, Communications of the ACM, 1971, t. 14, s. 228-239.

[107] Mc CUSKEY W.A.: On Automatic Design of Data Organization, Proc, AFIPS, 1970, FJCC t. 37, s. 187-199.

[108] McGEE W.: File Structures for Generalized Data Management, IFIP Congress 68 Edynburg, sierpień 1968, F-68-R-73.

[109] McGEE W.C. Generalized File Processing, Annual Review in Automatic Programming, Pergamon Press, 1969, t. 5, s. 77-149.

[110] McGEE W.C.: A Contribution to the Study of Data Equivalence, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[111] MEALY G.W.: Another Look of Data. Proc. AFIPS 1967 Fall Joint Computer Conference, 1967, t. 31, s. 525-534.

[112] MELTZER H.S.: Data Base Concepts and Architecture for Data Base Systems, IBM Report to SHARE Information Systems Research Projects, 1969.

[113] NAMIAN P.: Algebra of Management Information, IFIP Congress 68 Booklet F. Edynburg, sierpień 1968, s. F63.

[114] PAIR C.: Formalization of the Notions of Data, Information and Information Structure, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[115] PALERMO F.P.: A Data Base Search Problem, 4th International Symposium on Computer and Information Science, Miami Beach, grudzień 1972.

[116] PATT Y.: Variable Length Tree Structures Having Minimum Average Search Time. Communications of the ACM, 1969, t. 12, s. 72-76.

[117] RANDALL L.S.: A Relational Model of Data for the Determination of Optimum Computer Storage Structures, University of Michigan, Systems Engineering Lab., Report nr 54, 1971.

[118] ROTHNIE J.B.: The Design of Generalized Data Management Systems, rozprawa doktorska, Dept. of Civil Engineering, MIT, wrzesień 1972.

[119] ROVNER P.D., FELDMAN I.A.: The LEAP Language and Data Structures, Inf. Processing 68, North-Holland, Amsterdam, 1969.

[120] SALTON G.: Data Manipulation and Programming Problems in Automatic Information Retrieval, Communications of the ACM, marzec 1966, nr 3.

[121] SANDELRUS M.: On an Optimal Search Procedure, Amer. Math. Monthly, 1968, s. 133-134.

[122] SCHNEIDERMAN B.: Optimum Data Base Reorganization Points, Communications of the ACM, czerwiec 1973, t. 16.

[123] SENKO M.E., LUM V.Y., OWENS P.I.: A File Organization Evaluation Model (FOREM), Inf. Processing 68, North-Holland, Amsterdam, 1969.

[124] SENKO M.E.: File Organization and Management Information Systems, Annual Review of Information Science and Technology 4, Chicago, Illinois, 1969, s. 111-143.

[125] SENKO M.E., ALTMAN E.B., ASTRAHAN M.M., FEHDER P.L.: Data Structures and Accessing in Data Base Systems, IBM Systems Journal, 1973, t. 12, nr 1, s. 30-93.

[126] SEVERANEE D.G.: Some Generalized Modelling Structurer for Use in the Design of File Organization, Ph.D. dissertation, Univ. of Michigan, Ann Arbor, Michigan, 1972.

[127] SMITH D.P.: An Approach to Data Description and Conversion, Moore School Report, Pensylwania 1971, nr 72-20.

[128] STACEY G.M.: The Interface Between a Data Base and Its Host Languages, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[129] STANDISH T.: A Data Definition Facility for Programming Languages, rozprawa doktorska, Carnegie Techn. Rept., maj 1967.

[130] STANFEL L.E.: A Comment on Optimum Tree Structures, Communications of the ACM, 1969, t. 12, s. 582.

[131] STANFEL L.E.: Tree Structures for Optimal Searching, Journal of ACM, 1970, t. 17, s. 508-517.

[132] STOCKER P.M., DEARNLEY P.A.: Self Organizing Data Management Systems, The Comp. Journal, 1973, t. 16, nr 2, s. 100-105.

[133] STONE H.S.: Introduction to Computer Organization and Data Structures, McGraw Hill, New York 1972.

[134] STRNAD A.L.: The Relational Approach to the Management of the Data Base, Proc. IFIP Congress, Lubljana, 1971.

[135] STROSS C.O.M.: Operation of a Disc Data Base, The Computer Journal, 1972, t. 15, s. 220-297.

[136] SUNDGREN B.: An Infological Approach to Data Base, Sztokholm, 1973.

[137] SUNDGREN B.: Conceptual Foundation of the Infological Approach to Data Bases, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[138] A Survey of Generalized Data Base Management Systems, CODASYL Systems Committee Technical Report, maj 1969.

[139] SUSSENGUTH E.M. Jr.: The Use of Tree Structures for Processing Files, Communications of the ACM, 1963, t. 6, s. 272-279.

[140] TAYLOR R.W.: Generalized Data Base Management System Data Structures and Their Mapping to Physical Storage, rozprawa doktorska, University of Michigan, Ann Arbor, Michigan 1972.

[141] TAYLOR R.W., STEMPLE D.W.: On the Development of Data Base Editions, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[142] TITMAN P.J.: An Experimental Data Base System Using Binary Relations, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[143] TURSKI W.M.: On a Model of Information Retrieval System Based on Thesaurus, Information Storage and Retrieval, 1971, nr 7, s. 89-94, 201.

[144] TURSKI W.M.: A Model for Data Structures and its Applications (Part I). Acta Informatica, 1971, t. 1, Fasc. 1, s. 26.

[145] TURSKI W.M.: A Model for Data Structures and its Applications (Part II). Acta Informatica, 1972, t. 1, Fasc. 4, s. 282.

[146] TURSKI W.M.: Struktury danych, wyd. 1, Warszawa 1971, WNT.

[147] VUILLEMIN I.: Une formalisation de la notion de structures de données, rozprawa doktorska, Le Faculté des Sciences de Paris, czerwiec 1970.

[148] WATERS S.J.: File Design Fallacies, The Computer Journal, 1972, t. 15, s. 1-4.

[149] WATERS S.J.: Physical Data Structure. Paper 6 of Proceedings of BCS. Conference on Data Organization, 1970.

[150] WEDEKIND H.: Datenorganisation, Walter de Gruyer, Berlin, 1970.

[151] WEDEKIND H.: On the Selection of Access Paths in a Data Base System, IFIP-TC-2 Working Conference on "Data Base Management Systems", Cargese Corsica, 1974.

[152] WEGNER P.: Data Structure Models for Programming Languages, SIGPLAN Notices, 1971, t. 6, nr 2.

[153] WEGNER P.: Programming Languages, Information Structures and Machina Organization, McGraw-Hill, New York 1968.

[154] WILKES M.V.: Associative Tabular Data Structures, SIGPLAN Notices, 1971, t. 6, nr 2.

[155] WONG E., CHIANG T.C.: Canonical Structure in Attribute Based File Organization. Communications of the ACM, 1971, t. 14, s. 593-597.

[156] WROTEK Z.: Informacja o wynikach pracy Codasylu dotyczącej banku danych, ETO Nowości, 1972, nr 4, s. 63-69.

[157] YAMAMOTO S., TERAMOTO T., FUTAGAMI K.: Design of Balanced Multiplevalued Filing Scheme of Order two Based on Cyclically Generated Spread in Finite Projective Geometry, Information and Control 21, 1972, nr 1, s. 72-91.

[158] YOURDON E.: Design of On-line Computer Systems, Prentice-Hall Inc., 1972.

ИЗБРАННЫЕ ВОПРОСЫ ИЗ ОБЛАСТИ СТРУКТУР ДАННЫХ

Резюме

Статья состоит из семи глав, включая вступление. Две из этих глав посвящены одной проблеме, остальные касаются разных вопросов. Во 2 и 3 главах представлены основные понятия из теории структур данных. Подчеркивается принципиальная разница между отдельными понятиями, которые могут быть отождествлены, особенно в польской терминологии.

База данных и банк данных - это тема 4 главы. Среди многих цитированных, разных, а иногда и противоречных определений базы и банка данных, авторы приводят определение, которое по их мнению является наиболее удачным. 4 глава содержит также некоторые данные о COPASYL Data Base Task Group Report /1971/.

5 глава посвящена некоторым из многих публикаций, касающихся математических аспектов структур данных. Эти работы подразделяются на две группы. Одна группа посвящена описанию структур данных при использовании существующих математических теорий, другая - с той же целью - созданию новых теорий.

В 6 главе предлагается польская терминология.

7 глава (названная литературой) - это перечень больше 140 наименований публикаций (книги и статьи) из области структур данных.

SELECTED PROBLEMS OF DATA STRUCTURES

Summary

The paper consists of seven chapters including the introduction; two of them are interrelated, the other ones are concerning quite different themes.

Chapters 2 and 3 present some basic terms used in the theory of data structures. The main differences between some terms which may be treated as similar, especially because of their Polish names are underlined. Examples of such pair of terms are: data structure and storage structure, file and set, data management system and access methods and so on.

The subject of the chapter 4 is data base and data bank. Many different definitions of data base and data bank has been cited and authors have chosen the definition which is in their opinion the most suitable. In chapter 4 there are as well some remarks about the CODASYL Data Base Task Group Report /1971/.

Chapter 5 is devoted to some of the great variety of publications concerning with mathematical aspects of data structures. This contains two aspects: using existing mathematical theories to describe data structures and constructing new theories for this purpose.

Chapter 6 gives proposals for Polish terminology. It contains such terms as: data administrator, data bank and data base, block, record, key, identifier, access method and so on.

Chapter 7 - named bibliography - is the list of over 140 bibliography entries which are publications /books, articles/ dealing with subjects related to data structures.

Prace IMM
Zeszyt 1

681.327.2:621.375.823

ON PERFORMANCE CHARACTERISTICS OF INFORMATION PAGE COMPOSER IN DIGITAL HOLOGRAPHIC MEMORY

Zdzisław WRZESZCZ

Received 28.I.1974

This paper presents description of a performance of a very important submodule in holographic memory assembly - the page composer. Firstly, the cooperation between the processor and the holographic memory, as the working store, is outlined showing the necessity for information transfer from the electronic to light carrier. After providing means for finding equivalent information representation, an abstract system is built with intended characteristics of the page composer. Then the structure and main operation processes are described; these include writing and modulation. An analogy of structure and behaviour between the described page composer and a ferrite core memory matrix is emphasized. This analogy leads to a proposal of using similar approach in solving methodology problems of technical measurement of the page composer.

1. INTRODUCTION

The development of holographic memory depends largely on the digital electrooptic and optoelectronic system component designs. The page composer (PC) is one of them. Variety of materials and techniques have to be used to meet the functional demands of the page composer, this makes the complexity of synthesis.Next, the demands for the holographic memory, as a modern main store, are so high these days that the achiev-

- the surfaces $s^{(j)}(x, y)$,
- the areas $\bar{s}^{(j)}$ in x, y - plane,
- the displacements $\bar{d}^{(j)}$ between the centres of the elements.

The order of the element numeration and the joined problem of element displacement in the area $\bar{S}$ of information page depend greatly upon the shape of $\bar{S}$ and the particular technique of addressing. There are three distinct types of addressing techniques:

- matrix type [5],
- individual [6],
- scanning [7].

The first two are preferable for holographic memories because the page writing time may be greatly reduced.

Generally speaking the information page is a set of elements

$$S = \left\{ s^{(j)} : \bigcap_{j=1}^{N} s^{(j)} = \emptyset \right\} \qquad (6)$$

and possesses its geometrical characteristics:

- a surface

$$S(x, y) \supset \bigcup_{j=1}^{N} s^{(j)}(x, y), \qquad (6')$$

- an area $\bar{S}$ which is the projection of S (x, y) upon X, Y plane,
- and some table of numeration

$$T = \left\{ T[1], T[2], \ldots, T[N] \right\} \qquad (7)$$

in which each position T[j] contains the identifier j of the elementary area $s^{(j)}$ and some identifying number of the particular bit of the given word $B^{(\alpha)}$. The optical flux should have the shape as describes the formula (6), when recovered from the holographic medium (H in fig.1), to enable the physical detection of the information.

ed results in page composer constructions are far from satisfactory.

To enable any solution of the problem a fairly general model must be at the designer's disposal. In this paper we try to reach such a general performance description of the page composer. This description is also necessary when defining technical parameters and measuring methods of the system, it may also serve as a starting point for establishing demands on parameters of different materials used in page composer components.

The main function of the page composer is to transcribe the information from one type of carrier - voltage or current waveform, to another - a beam of coherent light. This may be done either by a passive form of light space modulator or by some active form of it. In this article we shall confine our discussion to the first type of the page composer.

2. REPRESENTATION OF STRUCTURE AND BEHAVIOUR OF THE PAGE COMPOSER

The so-called working memory is an arrangement which stores binary information e.g. data routines and intermediate and final results of processing. Schematic cooperation example of working store-processor is given in figure 1.

Usually the working store contains a set of numbered cells, so the word $B^{(\alpha)}$ in computer and also in the memory may be treated as an ordered set of logical variables

$$B^{(\alpha)} = \langle b^{(1)}, b^{(2)}, \ldots, b^{(P)} \rangle; \; b^{(i)} \in \{0,1\} \qquad (1)$$

where α - the word identifier
i - the bit identifier

The set (1) defines some point $\vec{b}$ in p dimensional binary logical space [1].

In a computer, there is a physical analogue of logical variable - some physical variable v(t) (usually an electric potential), which possesses two distinct waveforms

$$v^0(t),\ v^1(t) \tag{2}$$

in a given time interval

$$\tau_\gamma = [t_\gamma,\ t_{\gamma+1}) \tag{2'}$$

established by synchronization pulses which come from ACR module (fig. 1). Assuming

$$v^0 \equiv 0,\quad v^1 \equiv 1 \tag{3}$$

the ordered set of physical variables

$$\langle v_\gamma^{(1)},\ v_\gamma^{(2)},\ \ldots,\ v_\gamma^{(p)} \rangle\ ;\ v_\gamma^{(i)} \in \left\{ v^0,\ v^1 \right\} \tag{4}$$

defines the same point $\bar{b}$ in p-dimensional binary logical space. To be precise, the sequence (4) reflects the so-called parallel representation of the computer word.

In optical memories the information is carried by an optical beam thus a certain technical device must be used to transcribe the information from one type of the carrier-voltage waveform, to another - light waveform by means of either active or passive [13], [18] space modulator of light.

The transcription device, having also technical nick name page composer should possess $N \geqslant p$ elements $s^{(j)}$ emitting or passing the coherent light [2]

$$\hat{e}^{(j)}(\mathbf{r},\ t) = e^{(j)}(\mathbf{r})\ \exp(i\omega t)\ \exp(-ik^{(j)}\mathbf{r});$$

$$j = 1,\ 2,\ \ldots,\ N \tag{5}$$

depending on the state $\varkappa^{(j)}$ of the element $s^{(j)}$.

The elementary areas of $s^{(j)}$ in principle may have different shape but usually it is either the shape of square with the side $\bar{a}$ or a circle with the diameter $\bar{a}$. Because of technical reasons it is rational to assume equality for all elements $s^{(j)}$ of

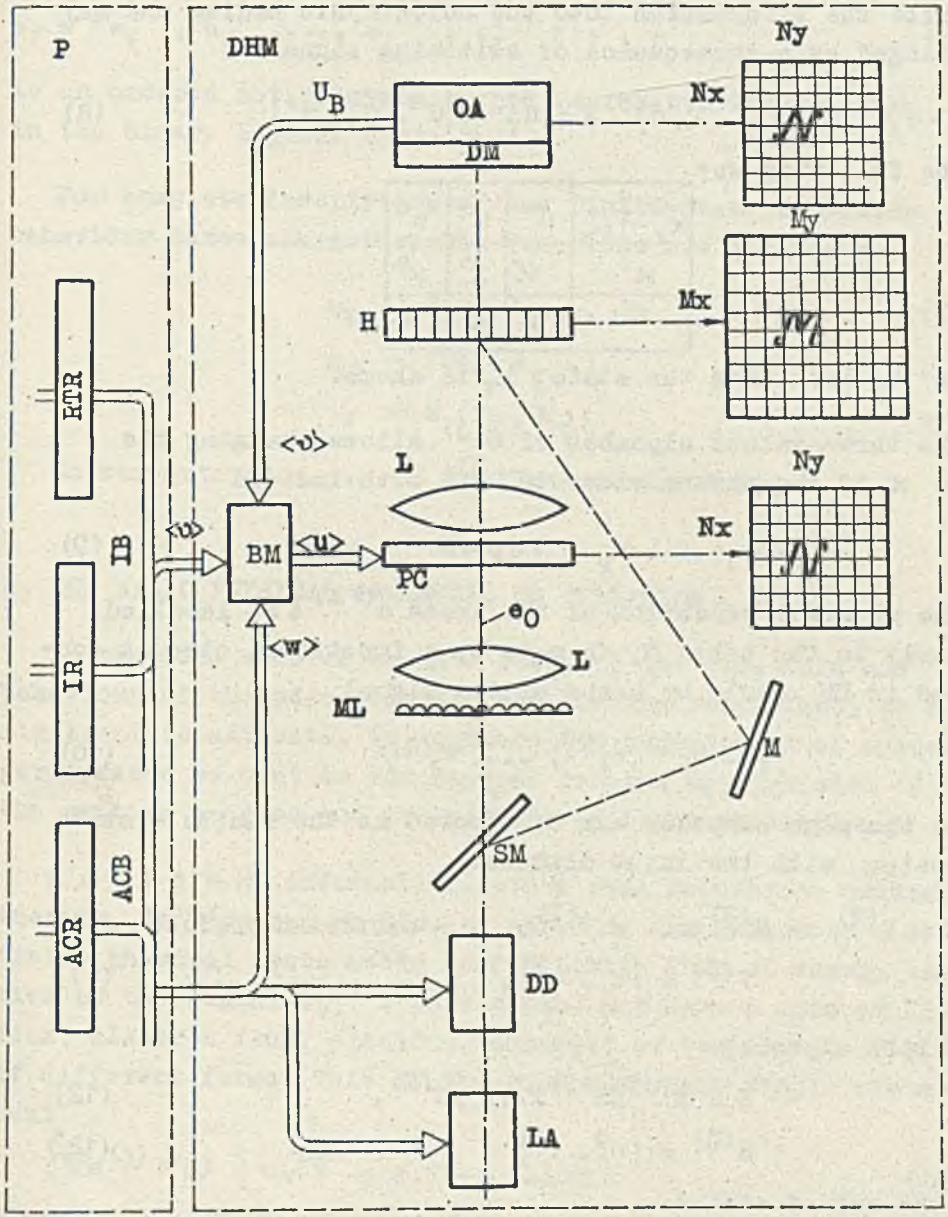


Fig. 1. An example of processor (P) - digital holographic memory (DHM) cooperation

ACB - address and control bus, IB - information bus, ACR - address and control register, IR - information register, RTR - real time register, BM - buffer module, <v> - input word signal, <V> - output word signal, OA - output amplifier, H - hologram, L - lens, PC - page composer, ML - matrix lens, DD - deflecting device, LA - laser, SM - splitting mirror

The state $\varkappa_\gamma$ of the PC is lasting τ_γ, e.g. long enough to write the information into the holographic medium and may be changed as a consequence of switching signals

$$u_\gamma = \langle u_\gamma^{(1)}, u_\gamma^{(2)}, \ldots, u_\gamma^{(N)} \rangle \; ; \; u_\gamma^{(j)} \in \{u^o, u^-, u^+\} \tag{8}$$

in the following way

$\varkappa_\gamma$ \ u	u^+	u^-	u^o
$\varkappa^o$	$\varkappa'$	$\varkappa^o$	$\varkappa^o$
$\varkappa'$	$\varkappa'$	$\varkappa^o$	$\varkappa'$

In the crossings the state $\varkappa_\gamma$ is shown.

The three-valued alphabet of $U^{(j)}$ allows changing the state $\varkappa$ of the chosen elements $s^{(i)}$ with indices

$$\alpha_o, \alpha_1, \ldots, \alpha_p \; ; \; 1 < p \leqslant N \tag{9}$$

The physical selection of the areas $s^{(\alpha i)} \in S$, labelled properly in the table T, is made by a functional circuit contained in BM module by means of the signal

$$w_\gamma = \langle w_\gamma^{(1)}, w_\gamma^{(2)}, \ldots, w_\gamma^{(q)} \rangle \tag{10}$$

So the page composer may be treated as the finite - state automaton with the input alphabet

$$V = V^{(1)} \times V^{(2)} \times \ldots \times V^{(P)} \; ; \; \times \text{ - cartesian product} \tag{11}$$

$$V^{(j)} = \{v^o, v^2\}, \tag{11'}$$

Output alphabet

$$E = E^{(1)} \times E^{(2)} \times \ldots \times E^{(N)}, \tag{12}$$

$$E^{(j)} = \{e^o, e^1\} \tag{12'}$$

and the set of states

$$K = K^{(1)} \times K^{(2)} \times \ldots \times K^{(N)}, \tag{13}$$

$$K^{(j)} = (\varkappa^o, \varkappa^1). \tag{13'}$$

Now, the output variable

$$e_\gamma = \langle e_\gamma^{(1)}, e_\gamma^{(2)}, \ldots, e_\gamma^{(N)} \rangle; \; e_\gamma^{(j)} \in \{e^0, e^1\} \tag{14}$$

is an ordered set of symbols and represents the point $\vec{b}$ in the binary logical space $\mathfrak{B}$.

For complete description of the finite-state automaton behaviour three characteristic functions are necessary

$$e_\gamma = f(v_\gamma, \varkappa_\gamma), \tag{15}$$

$$\varkappa_\gamma = F(u_\gamma, \varkappa_{\gamma-1}), \tag{15'}$$

$$u_\gamma = g(v_\gamma, w_\gamma) \tag{15"}$$

So our automaton is of the "Present-Past" type.

3. THE BASIC PHYSICAL PROCESSES OF OPERATION

In previous chapter we have outlined the structure and behaviour of the page composer showing its resemblance to a finite-state automata. To complete the description of system performance we want to add several remarks upon physics of the writing process and the process of light modulation.

W r i t i n g information into a real modulation medium, that is, driving the particular parts of the medium to a definite physical state needs some definite form of energy carried by the signal $u_\gamma^{(j)}$. This signal may have a form of light flux, electron flux, electric, magnetic or temperature field of different forms. This is the reason why the simple statement

$$(\forall s^{(j)} \in S) \left[u_\gamma^{(j)} = g(v_\gamma, w_{\gamma o}) \right]; \tag{16}$$

where

$\forall$ - universal quantifier

$\exists$ - existential quantifier

is not satisfactory for the existence of the stable state representing (1). In each case the signal $u_\gamma^{(j)}$ must be described by some ordered set of parameters $\xi^{(j)}$

$$\xi^{(i)}, \xi^{(2)}, \ldots, \xi^{(r)} \rangle; \; \forall u^{(j)} \tag{17}$$

If, for instance, the modulation medium is of photochromic type with so-called M_A colour centers [18], where $u_\gamma^{(j)}$ takes form of a light beam, the following set of parameters has to be controlled

$$\{\xi\} = \{\text{light intensity, space gradient of light intensity, crossection area of the beam, wavelength, polarization, direction of flow, pulse duration}\}$$

Besides the proper choice of the set $\{\xi\}$ representative one has to resolve the intervals of permitted changes of the parameters $\xi^{(1)}$

$$\Xi^{(1)}, \Xi^{(2)}, \ldots, \Xi^{(r)} \tag{18}$$

Thus knowing the relation

$$R_u(\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(r)}) \tag{19}$$

with its domains

$$D^{(1)}(R_u) = \Xi^{(1)}, D^{(2)}(R_u) = \Xi^{(2)}, \ldots, D^{(r)}(R_u) = \Xi^{(r)} \tag{19'}$$

we can give finally a formula

$$(\forall u^{(j)})(\exists <\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(r)}>)(\forall \xi^{(l)} \in \Xi^{(l)} \quad ; \tag{20}$$

$$l = 1, 2, \ldots, r) [R_u(\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(r)}) \Longrightarrow$$

$$\Longrightarrow <x^{(\alpha_0)}, x^{(\alpha_1)}, \ldots, x^{(\alpha_r)}> = <b^{(1)}, b^{(2)}, \ldots, b^{(p)}>]$$

which describes reliable writing the information (1) into the modulation medium.

M o d u l a t i o n of a beam of light passing through physical medium, or reflecting from it, is the form of transfering the information contained in state x_γ of the PC and the input v_γ to the output e_γ. The formula

$$e(x, y) = m(x, y)\, e_o; \quad \forall (x, y) \in S(x, y) \tag{21}$$

reflects this information transfer if

$$e_o(x, y) = \text{const}; \quad \forall (x, y) \in S(x, y)$$

and the space modulation factor $m(x, y)$ obtains values either m^0 or m^1 for the points $(x, y) \in S(x, y)$ in the same way as the state $\varkappa$ is supposed to do. According to our model of the information page ((6), (6'), (7)) the output $e_\gamma(x, y)$ is measured only in the elementary areas $s^{(j)}(x, y)$. Supposing

$$e(x, y) = \text{const}; \quad \forall (x, y) \in s^{(j)}(x, y) \tag{22}$$

one may change the (21) to the following

$$e^{(j)} = m^{(j)} e_o, \quad m^{(j)} \in \{m^o, m^1\}; \quad \forall j \in T \tag{23}$$

The space modulation factor $m^{(j)}$ takes a particular form depending on the physical modulation medium.

Several forms of modulation may be noticed but they may be linked into two distinguished groups:

(A) polarization modulation where modulation factor is in a matrix form

$$\begin{bmatrix} \hat{e}_x^{(j)} \\ \hat{e}_y^{(j)} \end{bmatrix} = \begin{bmatrix} m_{11}^{(j)}, & m_{12}^{(j)} \\ m_{21}^{(j)}, & m_{22}^{(j)} \end{bmatrix} \cdot \begin{bmatrix} \hat{e}_{o\,x} \\ \hat{e}_{o_y} \end{bmatrix}, \tag{24}$$

where each element of the matrix has its modulus and phase to be modulated

$$\hat{m}_{\mu,\nu}^{(j)} = m_{\mu,\nu}^{(j)} \exp\left(i \Gamma_{\mu\nu}^{(j)}\right) \tag{25}$$

and (B) phase and amplitude modulation without change of polarization

$$\hat{m}^{(j)} = m^{(j)} \exp\left(i \Gamma^{(j)}\right) \tag{26}$$

In both types of modulation there exists a predominance of amplitude or phase change.

The particular type of modulation is dictated by physics of the chosen modulation medium. The latter is built using

special materials with the ability of dynamic change of optical activity. Because of historical reasons we first quote the electrooptic crystals [3], [4], [5]. There exists a possibility to change in a local way the birefringence by driving an external electric field of a defined space shape in XY plane. In this case [8] the modulation is mostly of A^p - type.

C. Land [9] has discussed the page composer application of electrooptic ceramics which is divided according to the grain size: small grained ceramic is electrically controlled birefringent material and thick grained-electrically controlled scatterers.

Ferromagnetic materials such as: Mn Bi, EuO: Fe, NiFe, FeF_3, $FeBO_3$ [6] [10] possess the well known magnetooptic mechanisms.

The list of materials is far from completion to quote only the numerous family of liquid crystals [11], [12]. A well defined table of materials and related modulation mechanisms is included in I.N. Roberts paper [13]. We want only to emphasize that analysis and/or synthesis of the element $s^{(j)}$ microstructure and behaviour mechanisms is a selfcontained and rather large problem, nevertheless we hope the starting point for solving it may stem from the presented paper.

4. TECHNICAL PARAMETERS

Some analogy may be observed between the described page composer system and a memory matrix containing ferrite memory cores:

- there exists a similarity of structures,
- the elements $s^{(j)}$ in PC possess two discrete physical states as the ferrite core,
- procedures of addressing are similar,
- the information from the modulation medium is transfered to the output beam as from the core to the output voltage response.

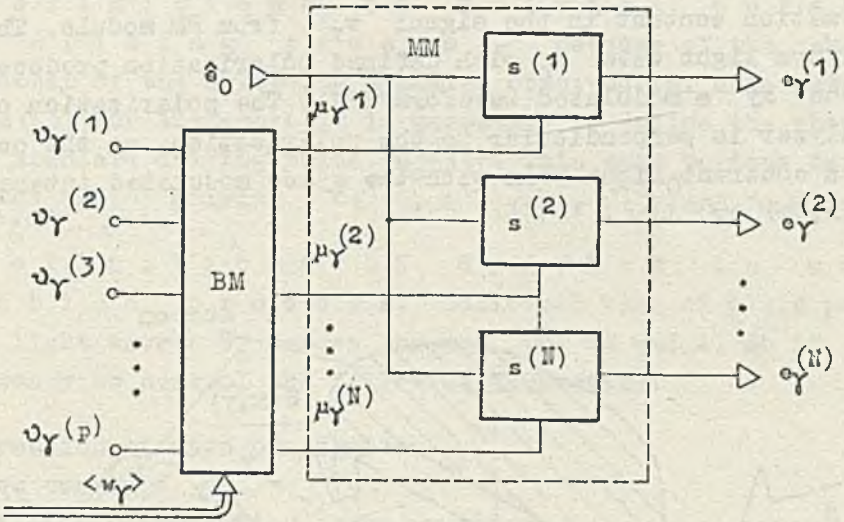


Fig. 2. Scheme of the page composer system

The problem of technical definition and measurements of core response parameters has a finite and well done theoretical form, there is also a long period of practice proved by many publications, e.g. [14], [15]. In conclusion we want to propose to use the same philosophy in methodology of measurement of page composer response parameters. So, after cited papers [14], [15] the page composer response parameter measuring methodology should include:

1/ measuring circuit,
2/ definitions of driving in writing process,
3/ definitions of driving in modulation process,
4/ response definitions,
5/ ambient conditions.

The m e a s u r i n g c i r c u i t. This circuit must be designed as a consequence of a particular physical system of PC; such one is given in fig. 3 as an example. The circuit consists of modulation medium (MM) of electrooptic type, with the modulation characteristics $m_{uv}^{(j)}$, as in [25] and field effect devices (FD) which produce the signal $U_\gamma^{(j)}$ in the form

of an electric field in the medium MM according to the information content in the signal v_γ from BM module. The uniform light wave $\hat{e}_0$ with defined polarization produces in plane xy a modulated waveform (21). The polarization of the analyzer is perpendicular to the polarization, so the output is a coherent light beam with the space modulated intensity $J_\gamma(x, y)$ or $J_\gamma^{(J)}$.

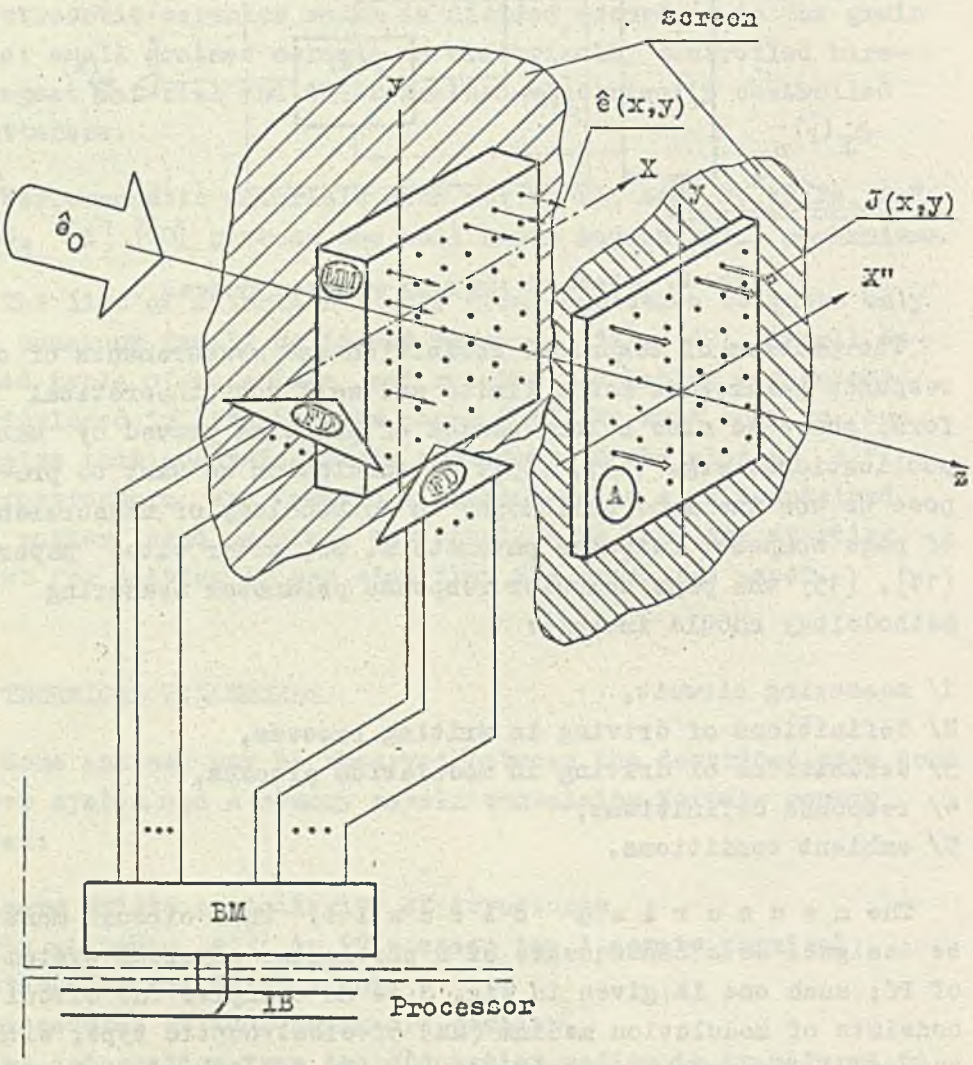


Fig. 3. Physical form of page composer system

D e f i n i t i o n s o f d r i v i n g p u l s e i n w r i t i n g p r o c e s s. Because of the charge character of the switching dynamics observed for some materials ([3] for instance) it is necessary to define the shape of a standard driving pulse $u_o^{(j)}(t)$. This must be done defining the "zero" driving $<\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(r)}>^0$ and "one" driving $<\xi^{(1)}\xi^{(2)}, \ldots, \xi^{(r)}>^1$.

D e f i n i t i o n s o f d r i v i n g i n m o d u l a t i o n p r o c e s s. Colimated beam of plain polarized light wave $\hat{e}_o$ passes through the MM and A. So it is necessary to control the following parameters:

- direction of wave propagation,
- wave constant,
- angle of polarization plane position,
- amplitude of input wave intensity,
- time duration of modulation process (of measurement).

All above mentioned parameters should be constant for each point $(x, y) \in S(x, y)$ in time $t \in \tau_\gamma$.

R e s p o n s e d e f i n i t i o n s. It comes out from system description that the page composer system response has the light intensity character similar to driving in modulation process. So it seems that the measurements of $J^{(j)}$, would be satisfying. But to help the decision in an estimation procedure it is feasible to use some relative parameter called c o n t r a s t q u o t i e n t $J^{(j)}(x^1) / J^{(j)}(x^o)$ in a similar way to the one-to-zero signal ratio measurements for cores. We shall also include the notion of disturbed and undisturbed responses for a given number of disturbances.

A m b i e n t c o n d i t i o n s. The choice of ambient condition parameters depends on the character of modulation medium material. Generally speaking it is necessary to include not only the influence of temperature, humidity pressure, shocks, vibrations but also different types of radiations.

5. CONCLUSIONS

In this paper a certain system philosophy was applied to describe the performance of page composer - an important submodule of a holographic memory.

This performance description forms, in our opinion, some base for eventual technical synthesis of the page composer. The description itself is made by means of mathematical formulae mostly, what enables to define the basic features of the PC and to show the relations among them in a compact way. This fact is very important for the future continuation of the problem. All mathematical notions and many of notations used throughout this paper do not exceed the material included in the book [19].

Acknowledgement

The author is greatly indebted to professor A. Janicki for his support of the work and much helpful discussion.

References

[1] AJZERMAN M.A. et al.: Logika, avtomaty, algoritmy, Moskwa, 1963.

[2] WRZESZCZ Z.: Układy optyczne w przetwarzaniu informacji, ETO Nowości, nr 1, 1973, s. 31.

[3] PULVARI C.F., DE LA PAZ A.S.: Phenomenological Theory of Polarization Reversal in Ferrielectric $Bi_4Ti_3O_{12}$ Single Crystals, Journal of Applied Physics, t. 37, nr 4, 1754.

[4] CUMMINS S.E.: Switching Behaviour of Ferroelectric $Bi_4Ti_3O_{12}$. Journal of Applied Physics, v. 36, no. 6, 1958.

[5] TAYLOR G.W.: A Method of Matrix Addressing Polarization Rotating or Recording Light-Valve Arrays. Proc. IEEE, v. 58, no. 11, 1812.

[6] RAJCHMAN J.A.: Promise of Optical Memories. Journal of Applied Physics, v. 41, no. 3, 1376.

[7] CUMMINS S.E., HILL B.H.: Electron-Beam Writing of Ferroelectric Domains in $Bi_4Ti_3O_{12}$ Sigle Crystals. Proc. IEEE (letters), v. 58, no. 6, 938.

[8] SHABANA M.M., JONES R.V.: Electrooptical Activity of Localized Perpendicularly Switched Domains in Ferroelectric Crystals. Proc. IEEE. v. 54, no. 1, 85.

[9] LAND C.E., THACHER P.D.: Ferroelectric Ceramics Electrooptic Materials and Devices. Proc. IEEE, vol. 57, no. 5, 751.

[10] ESHENFELDER A.A.: Promise of Magnetooptic Storage Systems Compared to Conventional Magnetic Technology. Journal of Applied Physics, v. 41, no. 3, 1372.

[11] SOREF R.A.: Liquid-Crystals Light-Control Experiments. The Physics of Opto-Electronic Materials. Plenum Press, N. York, 1971.

[12] JACOBSON A.D.et al.: The Liquid Crystals Light-Valve, an Optical to Optical Interface Device. Pattern Recognition, v. 5, no. 1, 13 (1973).

[13] ROBERTS I.N.: Strain-Biased PLZT Input Devices (Page Composers) for Holographic Memories and Optical Data Processing. Applied Optics, v. 11, no. 2, 397.

[14] Tentative Methods of Test for Nonmetallic Magnetic Cores to Be Used in a Coincident Current Memory with a Two-to-one Selection Ratio Operating under Full Switching Conditions. ASTM c526-63T, 1963.

[15] WRZESZCZ Z. et al.: Projekt standardu dotyczącego materiałów, rdzeni i ramek pamięci operacyjnej EMC, ETO Nowości, nr 1, 1968, s. 45.

[16] GILL A.: Introduction to the Theory of Finite-State Machines, Russian translation, Moskwa 1966.

[17] BURT J. et al.: Experimental High Density Optical Memory Using the Dichroic Absorption of the M_A Color Centres, Appl. Opt., t. 12, nr 6, 1213.

[18] KOTOVŠČIKOV G.S. et al.: Odpojannyj skanirujuščij laser vozbuzdajemyj elektronnym puškom. Kvantova Elektronika, nr 2, 428, 1974.

[19] RASIOWA H.: Wstęp do matematyki współczesnej. PWN, Warszawa.

O CECHACH DZIAŁANIA TWORNIKA STRONICY INFORMACJI W CYFROWEJ PAMIĘCI HOLOGRAFICZNEJ

Streszczenie

W niniejszej pracy zawarto podstawowe elementy opisu działania jednego z głównych podzespołów cyfrowej pamięci holograficznej - twornika stronicy informacji /TSI/. Mając na względzie zasadniczą funkcję TSI, tj. przeniesienie informacji z nośnika elektronicznego na nośnik optyczny, podano zapis formalny sygnału elektronicznego, wiązki światła, stronicy informacji, tablicy adresowej, sygnału optycznego oraz stanu. Wymienione elementy umożliwiły podanie opisu działania TSI w sposób zbliżony do opisu automatu skończonego. Stan TSI jako automatu reprezentuje stan fizyczny ośrodka modulującego wiązkę świetlną, stan fizyczny zaś powstaje w wyniku procesu pisania do tego ośrodka; podano więc opis warunków niezbędnych do prawidłowego przebiegu tego procesu. Po zaistnieniu odpowiedniego stanu /reprezentującego żądaną informację/ w ośrodku modulacyjnym zachodzi proces modulacji parametrów wiązki. Stan fizyczny ośrodka w tym procesie reprezentuje czynnik modulacji, uzyskuje się więc zapis pozwalający na badanie parametrów fizycznych zmodulowanej wiązki światła.

Na zakończenie zwrócono uwagę na podobieństwo cech fizycznych TSI, a zwłaszcza cyfrowego ośrodka modulującego, do płatu pamięciowego, np. płatu pamięci ferrytowej. Z wymienionego podobieństwa wyciągnięto wnioski odnośnie ogólnej metodyki badania fizycznych parametrów TSI.

О СВОЙСТВАХ ДЕЙСТВИЯ СОЗДАТЕЛЯ СТРАНИЦЫ ИНФОРМАЦИИ В ЦИФРОВОЙ ГОЛОГРАФИЧЕСКОЙ ПАМЯТИ

Резюме

В настоящем труде содержатся основные элементы описания действия одного из главных узлов цифровой голографической памяти – создателя страницы информации (ССИ) . Принимая во внимание основную функцию ССИ, т.е. перенос информации с электронного носителя на оптический носитель, приведена формальная запись электронного сигнала светового пучка, страницы информации, адресной таблицы оптического сигнала и состояния. Перечисленные элементы позволили приблизить описание действия ССИ к описанию конечного автомата. Состояние ССИ в качестве автомата представляет собой физическое состояние центра, модилирующего световой пучок, а физическое состояние появляется в результате процесса писания в этот центр; таким образом приведено описание условий, необходимых для правильного происхождения этого процесса. После появления соответствующего состояния (представляющего собой требуемую информацию) в модулирующем центре происходит процесс модуляции параметров пучка. Физическое состояние центра в этом процессе представляет собой фактор модуляции и таким образом получается запись, позволяющая исследовать физические параметры модулированного светового пучка.

В заключение обращено внимание на подобие физических свойств ССИ, а особенно цифрового модулирующего центра, плату памяти, например плату ферритной памяти. На основе этого подобия сделаны выводы относительно общей методики исследования физических параметров ССИ.

Prace IMM
Zeszyt 1

681.322.06:800.92

LL - REGULAR GRAMMARS

Stanisław JARZĄBEK
Tomasz KRAWCZYK

Received 11th, 07.1974

LLS(k) grammars defined by Lewis and Stearns [6] are the class of CF grammars that can be parsed in a top-down manner without backtrack. Following the example of K.Culik and R.Cohen [2] in the case of Knuth's LR(k) grammars, we introduce LL-regular grammars which are an extension of LLS(k) grammars. Several properties of these grammars are given in this paper, as well as some necessary conditions for a grammar to be LL-regular are derived.

1. LLR GRAMMARS

Before defining these grammars some notations concerning context - free grammars and LLS(k) grammars should be introduced.

We represent a context - free grammar G by a four - touple (V_N, V_T, P, S), where V_N and V_T are finite disjoint sets of symbols called nonterminal and terminal alphabets respectively, P is a finite set of the productions of the form $A \rightarrow \alpha$ with $A \in V_N$ and $\alpha \in (V_N \cup V_T)^*$ and $S \in V_N$ is the starting symbol. By V we denote $V_N \cup V_T$.

For γ_1 and $\gamma_2 \in V^*$ $\gamma_1 \overset{}{\underset{G}{\Rightarrow}} \gamma_2$ iff there exist $\beta, \delta \in V_T^*$ and production $A \rightarrow \alpha$ in P such that $\gamma_1 = \beta A \delta$ and $\gamma_2 = \beta \alpha \delta$.

We write $\gamma_1 \overset{}{\underset{G}{\Rightarrow}}_L \gamma_2$ if β is in V_T^*.

We assume that $\overset{*}{\underset{G}{\Longrightarrow}}$ ($\overset{*}{\underset{GL}{\Longrightarrow}}$) represents a reflexive transitive completion of $\underset{G}{\Longrightarrow}$ ($\underset{GL}{\Longrightarrow}$). The string $\alpha \in V^*$ such that $S \overset{*}{\underset{G}{\Longrightarrow}} \alpha$ will be called the sentential form.

The language generated by G is the following set $L(G) = \{x \in V_T^* \mid S \overset{*}{\underset{G}{\Longrightarrow}} x\}$, where S is the starting symbol. The set of productions with the same nonterminal symbol on the left side is called a rule.

Upper case Latin letters will denote nonterminals,
letters: a, b, c, d, ... - terminals,
letters: u, v, w, x ... will denote strings belonging to V_T^*
and $\alpha, \beta, \gamma, \delta$ ... will denote strings belonging to V^*.

For any string δ we define $k{:}\,\delta$ to be the string consisting of the first k symbols of δ, if δ has more than k symbols or δ in the other case.

Definition 1

Let $k > 0$. A grammar $G = (V_N, V_T, P, S)$ is said to be LLS(k) grammar iff for any strings $u_1, u_2, x_1, x_2 \in V_T^*$ and any strings $\alpha_1, \alpha_2, \delta_1, \delta_2 \in V^*$ from

$$S \overset{*}{\underset{G}{\Longrightarrow}}_L u_1 A \delta_1 \underset{G}{\Longrightarrow}_L u_1 \alpha_1 \delta_1 \overset{*}{\underset{G}{\Longrightarrow}}_L u_1 x_1$$

$$S \overset{*}{\underset{G}{\Longrightarrow}}_L u_2 A \delta_2 \underset{G}{\Longrightarrow}_L u_2 \alpha_2 \delta_2 \overset{*}{\underset{G}{\Longrightarrow}}_L u_2 x_2$$

and

$k{:}x_1 = k{:}x_2$ we have $\alpha_1 = \alpha_2$

Let $p = A \longrightarrow \alpha$ and $k > 0$. We introduce "look-ahead" sets

$$H_p^k : H_p^k = \left\{k{:}x \mid S \overset{*}{\underset{G}{\Longrightarrow}}_L uA\delta,\ A\delta \underset{G}{\Longrightarrow}_L \alpha\delta \overset{*}{\underset{G}{\Longrightarrow}}_L x\right\},$$

where $u, x \in V_T^*$, $\delta \in V^*$.

Theorem 1

A grammar G is LLS (k) grammar iff for any rule $U = \{p_1, p_2, \dots, p_n\}$ $H^k_{pi} \cap H^k_{pj} = \emptyset$ for any $1 \leqslant i < j \leqslant n$ ($\emptyset$ denotes the empty set).

Let $\pi = \{R_1, \dots, R_n\}$ be a partition of V_T^* into n disjoint sets R_i. Set π is called a regular partition if all sets R_i are regular.

Now, we extend the definition of LLS(k) grammars in such a way that the production used in any sentential form will be uniquely determined not by the first k symbols of the analysed string but by set R_i containing this part of that string.

Definition 2

Let $G = (V_N, V_T, P, S)$ be any CF grammar with k rules and let U denote a rule in P:

$$U = \{A \longrightarrow \alpha_1, \dots, A \longrightarrow \alpha_n\}.$$

Let $\pi = \{R_1, \dots, R_n\}$ denote a regular partition of V_T^*.

The rule U is called a LL(π)rule iff for any two derivations of the form

$$S \overset{*}{\underset{G}{\Longrightarrow}} L \; u_1 A \delta_1 \underset{G}{\Longrightarrow} L \; u_1 \alpha' \delta_1 \overset{*}{\underset{G}{\Longrightarrow}} L \; u_1 x_1$$

$$S \overset{*}{\underset{G}{\Longrightarrow}} L \; u_2 A \delta_2 \underset{G}{\Longrightarrow} L \; u_2 \alpha'' \delta_2 \overset{*}{\underset{G}{\Longrightarrow}} L \; u_2 x_2$$

where $u_1, u_2, x_1, x_2 \in V_T^*$, $\delta_1, \delta_2 \in V^*$, from the existence of $l : 1 \leqslant l \leqslant n$ such that x_1 is in R_l and x_2 is in R_l, we have $\alpha' = \alpha''$.

A grammar G is called LL(π_1, ... , π_k) if for any $1 \leq i \leq k$ the rule U_i is LL(π_i) rule.

A grammar G is called LL - regular (LLR) iff G is LL(π_1, ... , π_k) for some fixed regular partitions π_1,..., π_k of V_T^*.

A language L is LLR iff there exists a LLR grammar G such that L = L(G).

Example 1

Let $G = (\{S, A, B\}, \{a, b\}, P, S)$ where P contains

$S \longrightarrow a\ A\ b$, $S \longrightarrow a\ B\ a$

$A \longrightarrow a\ A\ a$, $A \longrightarrow b$

$B \longrightarrow a\ B\ a$, $B \longrightarrow b$.

This grammar isn't LLS(k) grammar for any k because of the analysed string $x = a\ a^k b$ ... and of two derivations of the form:

$$S \longrightarrow a\ A\ b \overset{*}{\underset{G}{\Longrightarrow}}_L a\ a^k b\ a^k b$$

$$S \longrightarrow a\ B\ a \overset{*}{\underset{G}{\Longrightarrow}}_L a\ a^k b\ a^k a\ .$$

The production to be used in the first step of the analysis cannot be determined by using the first k symbols of the string x. But rule $U = \{S \longrightarrow aAb,\ S \longrightarrow aBa\}$ is clearly LL(π) for the regular partition $\pi = (V_T^* b,\ V_T^* a)$.

Theorem 1.1

Every LLS(k) grammar is LLR grammar.

Proof

To show this, it suffices to take for every rule $U = \{A \longrightarrow \alpha_1, \ldots, A \longrightarrow \alpha_n\}$ the regular partition $\pi_A = \{R_1, \ldots, R_n\}$, where $R_i = H_{pi}^k V_T^*$.

2. THE PARSING ALGORITHM

The parsing algorithm works similary to the parsing algorithm for LLS(k) grammars (see [7]).

Let G be a given $LL(\pi_1, \ldots, \pi_k)$ grammar for some regular partitions of V_T^* : $\pi_1, \ldots, \pi_k$. The analyser consists of a stack and some finite-state acceptors which accept all sets R_i from the regular partitions $\pi_1, \ldots, \pi_k$. The parsing algorithm works in the following manner:
Let the nonterminal symbol A be on the top of the stack and let U be a rule of the form $A \to \alpha_1, \ldots, A \to \alpha_n$ and $\pi_i = \{R_1, \ldots, R_n\}$ be the regular partitions such that U is $LL(\pi_i)$. Then we check if the currently part of the analysed string belongs to some R_i $1 \leqslant i \leqslant n$ (it is if this string is accepted by suitable acceptor). If so, we put on the stack the right side of the production $A \to \alpha_i$ and consider the symbol on the top of the stack. If no, then an error occurs. If a terminal symbol appears on the top of the stack, then we delete it from the stack and from the analysed string.

3. SOLVABLE AND UNSOLVABLE PROBLEMS FOR LLR GRAMMARS

The two theorems presented below concern the decidability of the following problems: whether CF grammar is a LLR grammar for some given regular partitions and whether the problem is solvable for any partitions.

First, we give some lemmas and theorems.
Let $G = (V_N, V_T, P, S)$ be a CF grammar and $U = \{p_1, \ldots, p_n\}$, be a rule where $p_i = A \to \alpha_i$. Let $S \overset{*}{\underset{G}{\Longrightarrow}} {}_L\, uA\delta \underset{G}{\Longrightarrow} {}_L\, u\alpha_i\delta$.

We define the languages:

$$L_A^{p_i} = \left\{ x \in V_T^* \mid A \to \alpha_i \overset{*}{\underset{G}{=\!=}} x \right\}$$

$$L_U^{p_i} = L_A^{p_1} \cup \ldots \cup L_A^{p_{i-1}} \cup L_A^{p_{i+1}} \cup \ldots \cup L_A^{p_n}$$

$$L^{A}_{\delta} = \{ x \in V_T^* \mid \delta \overset{*}{\underset{G}{\Longrightarrow}} x, \text{ for each string } \delta \text{ such}$$

that exist derivation $S \overset{*}{\underset{G}{\Longrightarrow}}_L uA\delta \}$

$$L_{p_i\delta} = \{ \alpha_i \delta \in V^* \mid S \overset{*}{\underset{G}{\Longrightarrow}}_L uA\delta \underset{G}{\Longrightarrow}_L u\alpha_i\delta \} .$$

Obviously, the languages $L_A^{p_i}$ and $L_U^{p_i}$ are context - free languages and can be effectively determined. We want to show that the language $L_A^{p_i} L^{A}_{\delta}$ is a context - free language and that it can be effectively determined.

Let $G = (V_N, V_T, P, S)$ be a given CF grammar. First we prove the following lemma.

Lemma 3.1

The language $L_{p_i\delta}$ is a regular language.

Proof

We take the production $p_i = A \longrightarrow \alpha_i = X_1 \dots X_k$ and create the grammar $G' = (V'_N, V'_T, P', S')$ in the following way:

$$V'_N = \{ X' \mid X \in V_N \}$$

$$V'_T = V$$

$$P' = \{ B' \longrightarrow Y'_j\, Y_{j+1} \dots Y_1 \quad \text{for every production } B \longrightarrow Y_1 \dots Y_1 \text{ in P and for each } 1 \leq j \leq 1 \text{ so that } Y_j \in V_N .$$

$$A' \longrightarrow X_1 \dots X_k \quad \text{for} \quad p_i : A \longrightarrow X_1 \dots X_k \}$$

It's obvious that the grammar just created is a regular grammar. Now we show that G' generates the language $L_{p_i\delta}$.

We must prove that

$$S' \overset{*}{\underset{G'}{\Longrightarrow}} A'\delta \text{ iff } S \overset{*}{\underset{G}{\Longrightarrow}}_L uA\delta \text{ for some } u \in V_T^* .$$

Let $S' \overset{*}{\underset{G'}{\Longrightarrow}} A'\delta$. We have to show that $S \overset{*}{\underset{G}{\Longrightarrow}}_L uA\delta$ for

some $u \in V_T^*$.

We prove that by induction on n, where n denotes the length of the derivation of the string $A\delta$.

For $n = 1$ this implication is evidently true.

Let $n \geqslant 2$ and assume the implication true for $1, \ldots, n-1$.

Let us take the derivation $S' \overset{*}{\underset{G'}{\Longrightarrow}} A'\delta$, the length of which is n. This derivation can be written in the form:

$$S' \overset{*}{\underset{G'}{\Longrightarrow}} B'\delta_2 \Longrightarrow A'\delta_1\delta_2 \quad \text{for some } \delta_1, \delta_2 \in V^*$$

such that $\delta = \delta_1\delta_2$ and for some production $B' \longrightarrow A'\delta_1$. Since the length of the derivation $B'\delta_2$ is $n-1$, there exists, by the induction hypothesis, the derivation in G: $S \overset{*}{\underset{G}{\Longrightarrow}}_L u_1 B\delta_2$ for some $u_1 \in V_T^*$. As for the production $B' \longrightarrow A'\delta_1$ there must exist production $B \longrightarrow \beta A\delta_1$ in G, where $\beta \overset{*}{\underset{G}{\Longrightarrow}}_L u_2$, then the following derivation exists in G:

$$S \overset{*}{\underset{G}{\Longrightarrow}}_L u_1 B\delta_2 \underset{G}{\Longrightarrow}_L u_1\beta A\delta_1\delta_2 \overset{*}{\underset{G}{\Longrightarrow}}_L u_1 u_2 A\delta_1\delta_2 = uA\delta,$$

where $u_1 u_2 = u$.

Now we prove the converse implication. We prove it by induction on the length of the derivation of the string $uA\delta$.

Let $S \overset{*}{\Longrightarrow}_L uA\delta$. For $n = 1$ the implication is obviously true. Let $n \geqslant 2$ and assume the implications true for $1, \ldots, n-1$. Let the length of the derivation $S \overset{*}{\underset{G}{\Longrightarrow}}_L uA\delta$ be n.
It can be written in the form

$$S \overset{*}{\underset{G}{\Longrightarrow}}_L u_1 B\delta' \underset{G}{\Longrightarrow}_L u\beta\delta' = uA\delta \qquad \text{for some } \delta'$$

and for some production $B \longrightarrow \beta$.

Two cases are possible here:
Firstly, there exists a production $B \longrightarrow u_2 A\delta_1$ such that $u_1 u_2 = u$,

Secondly, there exists a production $B \longrightarrow u_3$, where $u_3 \in V_T^*$ and A belongs to δ'.

In the first case, since the length of the derivation $u_1 B \delta'$ is shorter than n, then $S' \overset{*}{\underset{G'}{\Longrightarrow}} B'\delta'$.

For the production $B \longrightarrow u_2 A \delta_1$ there exists a production $B' \longrightarrow A'\delta_1$ in G'. Thus there exists derivation:

$$S' \overset{*}{\underset{G'}{\Longrightarrow}} B'\delta' \underset{G'}{\Longrightarrow} A'\delta_1\delta' = A'\delta$$

In the other case there has to exist a derivation:

$$S \overset{*}{\underset{G}{\Longrightarrow}}_L u_2 C\delta_2 \underset{G}{\Longrightarrow}_L u_2\beta' A\delta_1\delta_2 \quad \text{and} \quad u_2\beta' \overset{*}{\underset{G}{\Longrightarrow}}_L u,$$

$$\delta_1\delta_2 = \delta.$$

Since production $C' \longrightarrow A'\delta_1$ in G' exists for the production $C \longrightarrow \beta' A \delta_1$ and by the induction hypothesis $S' \overset{*}{\underset{G'}{\Longrightarrow}} C'\delta_2$, therefore

$$S' \overset{*}{\underset{G'}{\Longrightarrow}} A'\delta_1\delta_2 = A'\delta$$

So, we have proved that:

$$S' \overset{*}{\underset{G'}{\Longrightarrow}} A'\delta \quad \text{iff} \quad S \overset{*}{\underset{G}{\Longrightarrow}}_L uA\delta \quad \text{for some} \quad u \in V_T^*.$$

Since the production $A \longrightarrow \alpha_i$ exists in G and the production $A' \longrightarrow \alpha_i$ exists in G, therefore

$$S' \overset{*}{\underset{G'}{\Longrightarrow}} \alpha_i\delta \quad \text{iff} \quad S \overset{*}{\underset{G}{\Longrightarrow}}_L u\alpha_i\delta \quad \text{for some} \quad u \in V_T^*.$$

Theorem 3.1

A language $L_A^{p_i} L_\delta$ is a context - free language and it can be effectively determined.

Proof

We create a grammar $G'' = (V_N'', V_T'', P'', S'')$, where:

$V_N'' = V_N \cup V_N'$

$V_T'' = V_T$

$S'' = S'$

$P = P \cup P'$ and V_N', S', P' are the elements of the grammar G' from Lemma 3.1.

We must prove that $L(G'') = L_A^{p_i} L_\delta$.

Let $x \in L(G'')$, i.e. there exists a derivation $S' \overset{*}{\underset{G''}{\Longrightarrow}} x$. Since the production $A' \to \alpha_i$ is the only production which has a symbol from V_N' on its left side and hasn't a symbol from V_N' on its right side, then this production must be used in the derivation of the string x, i.e.

$$S'' = S' \overset{*}{\underset{G''L}{\Longrightarrow}} A'\delta \overset{*}{\underset{G''L}{\Longrightarrow}} \alpha_i \delta \overset{*}{\underset{G''L}{\Longrightarrow}} x , \qquad \delta \in V^*$$

In the derivation of $A'\delta$ we use only the productions from P'. From the first part of the proof of Lemma 1 we receive

$$S \overset{*}{\underset{GL}{\Longrightarrow}} uA\delta \underset{GL}{\Longrightarrow} u\alpha_i \delta \qquad \text{for some } u \in V_T^*.$$

It is obvious that in the derivation of x from $\alpha_i \delta$ we use only productions from P. Thus, $x = yz$, where $y \in L_A^{p_i}$ and $z \in L_\delta$, so

$$x \in L_A^{p_i} L_\delta .$$

Let $x \in L_A^{p_i} L_\delta$, i.e. $x = yz$, where $y \in L_A^{p_i}$ and $z \in L_\delta$. Since $P'' = P \cup P'$, for the derivations

$$S \overset{*}{\underset{GL}{\Longrightarrow}} uA\delta, \quad A \underset{GL}{\Longrightarrow} \alpha_i \overset{*}{\underset{GL}{\Longrightarrow}} y \quad \text{and} \quad \delta \overset{*}{\underset{GL}{\Longrightarrow}} z \quad \text{in } G,$$

there exist the following derivations in G'':

$$S'' \overset{*}{\underset{GL}{\Longrightarrow}} A'\delta \quad \text{(from Lemma 1)}, \quad A' \to \alpha_i \overset{*}{\underset{GL}{\Longrightarrow}} y$$

and $\delta \overset{*}{\underset{G''L}{\Longrightarrow}} z$. Thus

$S'' \overset{*}{\underset{G''_L}{\Longrightarrow}} A' \delta \overset{*}{\underset{G''_L}{\Longrightarrow}} \alpha i \delta \overset{*}{\underset{G''_L}{\Longrightarrow}} y \delta \overset{*}{\underset{G''}{\Longrightarrow}} yz = x$, that is $x \in L(G')$.

Theorem 3.2

Let G be a CF grammar.

For a given rule $U = \{A \to \alpha_1, A \to \alpha_2, \ldots A \to \alpha_n\}$ and a fixed regular partition $\pi = \{R_1, \ldots, R_n\}$ it is decidable whether this rule is a $LL(\pi)$ rule.

Proof

It is obvious that U is a $LL(\pi)$ rule iff for each $i \leq n$, $L_A^{p_i} L_\delta \subset R_i$ and $L_U^{p_i} L_\delta \subset V_T^* - R_{i_{p_i}}$ it is iff for each i $L = (R_i \cap L_U^{p_i} L_\delta) \cup ((V_T^* - R_i) \cap L_A^{p_i} L_\delta) = \emptyset$.

Since $L_A^{p_i} L_\delta$ and $L_U^{p_i} L_\delta$ are CF languages (theorem 3.1) $R_i \cap L_U^{p_i} L_\delta$ and $(V_T^* - R_i) \cap L_A^{p_i} L_\delta$ are also CF languages, so L is CF language which can be effectively determined, and the problem whether L is empty is decidable.

As it was indicated above, one can effectively decide if G is $LL(\pi_1, \ldots, \pi_k)$ in the case when the regular partitions $\pi_1, \ldots, \pi_k$ are given, but in the general case the problem whether a CF grammar is LLR is unsolvable

Theorem 3.3 (following Culik and Cohen [2]).

It is not decidable for the two given CF languages $L_1 = L_A^{p_i} L_\delta$ and $L_2 = L_U^{p_i} L_\delta$ whether there exists any regular set R containing L_1 and disjoint from L_2.

4. NECESSARY CONDITIONS FOR A CF GRAMMAR TO BE A LLR GRAMMAR

Let $G = (V_N, V_T, P, S)$ be a CF grammar with the rule U:

$U = \{p_1 = A \to \alpha_1, \ldots, p_n = A \to \alpha_n\}$ and let

$\pi = \{R_1, \ldots, R_n\}$ be some regular partition.

Theorem 4.1

If U is a LL(π) rule then for every i,j: $1 \leqslant i < j \leqslant n$

$$L_A^{p_i} L_\delta^A \cap L_A^{p_j} L_\delta^A = \emptyset$$

The proof is quite obvious.

Theorem 4.2

If $L_A^{p_i} L_\delta^A \cap L_A^{p_j} L_\delta^A = \emptyset$ then $L_A^{p_i} \cap L_A^{p_j} = \emptyset$

Proof

We shall prove this condition by leading to contradiction. Let us assume that $L_A^{p_i} \cap L_A^{p_j} \neq \emptyset$, i.e. there exists a string x, such that $x \in L_A^{p_i}$ and $x \in L_A^{p_j}$.

Then there exists a string y, such that $y = x, z$, $z \in L_\delta$, and $y \in L_A^{p_i} L_\delta^A \cap L_A^{p_j} L_\delta^A$.

If there exist i, j for which $L_A^{p_i} \cap L_A^{p_j} \neq 0$, then it follows from Theorem 4.1 and Theorem 4.2 that the rule U is not LL(π) rule any π.

So, we receive the following conclusion:

Conclusion

Ambiguous languages aren't LLR languages.

Theorem 4.3

If $L_A^{p_i} \cap L_A^{p_j} = \emptyset$ then $L_{p_i}\delta \cap L_{p_j}\delta = \emptyset$

Proof

The proof of this condition is similar to the proof used in Theorem 4.2.

Let us assume that there exists such α that $\alpha \in L_{p_i}\delta$ and $\alpha \in L_{p_j}\delta$, so

$$S \underset{G}{\overset{*}{\Longrightarrow}}_L u\alpha = u\alpha_i\,\delta \underset{G}{\overset{*}{\Longrightarrow}}_L y \quad \text{and}$$

$$S \overset{*}{\underset{G_L}{\Rightarrow}} u\alpha = u\alpha_j \delta \overset{*}{\underset{G_L}{\Rightarrow}} y \qquad \text{thus} \quad L_A^{p_i} \cap L_A^{p_j} \neq 0:$$

In the class of CF languages it is not decidable whether any intersection of two languages is a CF language. But any intersection of two regular languages is a regular language and the problem of emptiness in this class of languages is decidable. So from the Theorem 4.3 we receive the decidable necessary condition for the given rule U to be LLR(π) for some regular partition π. Unfortunately the above condition is not sufficient. Further studies are being performed to find this condition as well as practical methods of construction of regular π partitions.

References

[1] CULIK K.: Syntactical Analysis, Department of Applied Analysis and Computer Science, University of Waterloo, 1971.

[2] CULIK K., COHEN R.: LR-Regular Grammars - an Extension of LR(k) Grammars, Journal of Computer and System Sciences 7, 1973, s. 66-96.

[3] GRIES D.: Compiler Construction for Digital Computers, John Wiley and Sons., Inc, 1971.

[4] KNUTH D.E.: Top-Down Syntax Analysis, Acta Informatica, t. 17, nr 3, 1970, s. 226-256.

[5] KURKI-SUONIA R.: Notes on Top-Down Languages - BIT 9, 1969, s. 225-238.

[6] LEWIS P.M., STEARNS R.E.: Syntax-Directed Transductions, J.ACM 15, 1969, s. 465-488.

[7] ROSENKRANTZ D.J., STEARNS R.E.: Properties of Deterministic Top-Down Grammars, Information and Control 17, 1970, s. 226-236.

LL - REGULARNE GRAMATYKI

Streszczenie

Zdefiniowane przez Lewisa i Stearnse'a [6] LLS(k) gramatyki mogą być analizowane "bez powrotów" w sposób generacyjny. Wprowadzamy LL-regularne gramatyki, które są rozszerzeniem LLS(k) gramatyk, podobnie jak to zrobił K.Culik i R.Cohen [2] dla LR(k) gramatyk Knutha. W pracy podane są pewne własności LLR gramatyk oraz warunki konieczne jakie musi spełniać gramatyka, aby była LLR gramatyką.

ЛЛ - РЕГУЛЯРНЫЕ ГРАММАТИКИ

Резюме

Граммати LLS/k/,введенные Левисом и Стернсом [6] , принадлежат к классу контекстно свободных грамматик,анализ которых производится методом "top-down" без поворотов. В работе определены регулярные грамматики LLR , которые являются расширением грамматик "LLS/k/, подобно тому, как грамматики LRR, введенные К.Чуликом и Р.Кохеном [2] , являются расширением грамматик Кнута LR/k/. Доказывается несколько свойств грамматик LLR , а также необходимые условия, которым должна отвечать грамматика, для того, чтобы она была LLR грамматикой.

Prace IEM
Zeszyt 1
© 1974.08

621.315.612:
681.327.1.613

BADANIA MATERIAŁÓW CERAMICZNYCH MOGĄCYCH ZNALEŹĆ ZASTOSOWANIE W GŁOWICACH MAGNETYCZNYCH

Joanna SYNAK
Wacław WELIK

Pracę złożono 17.08.1973

Omówiono wymagania stawiane materiałom ceramicznym do głowic magnetycznych, dokonano przeglądu tych materiałów. Przeprowadzono badania ścieralności, obrabialności oraz podano wyniki docierania gładkościowego, wytypowano najbardziej odpowiedni materiał na elementy nośne (obudowy) głowic.

Spis treści

1. WSTĘP

Rozwój EMC stawia coraz większe wymagania odnośnie jakości elementów i podzespołów, a więc i materiałów, z których są one wykonywane. Wymagania te powodują poszukiwania nowych, coraz lepszych i coraz bardziej niezawodnych materiałów.

Spośród znanych materiałów korzystnymi właściwościami wyróżniają się materiały ceramiczne. Odznaczają się one doskonałymi i ściśle określonymi właściwościami fizycznymi, wykazują dużą odporność na działanie erozyjne powietrza i działanie temperatury, wykazują dużą odporność na zmęczenie powierzchniowe i starzenie się, a elementy z nich wykonywane zaliczane są do najbardziej stabilnych geometrycznie i niezawodnych. Badania zjawisk i intensywności zużywania ściernego są jednym z wielu procesów towarzyszących konstruowaniu i produkowaniu głowic magnetycznych. Przy wyborze materiałów powinna być także brana pod uwagę ich obrabialność, gdyż większość elementów głowic poddawana jest bardzo pracochłonnej obróbce gładkościowej. Dokładność wykonania (wymiarowa) zawarta jest najczęściej w przedziale klas 1 ÷ 3, a wymagana gładkość powierzchni dotyczy przeważnie klas chropowatości 11 ÷ 14.

Celem artykułu jest rozpatrzenie możliwości zastosowania materiałów ceramicznych na elementy nośne (obudowy) głowic magnetycznych oraz wytypowanie najbardziej odpowiedniego materiału.

2. WYMAGANIA STAWIANE MATERIAŁOM CERAMICZNYM PRZEZNACZONYM DO GŁOWIC MAGNETYCZNYCH

Specyficzne warunki pracy głowic magnetycznych, a przede wszystkim odległość elementu nośnego (obudowy) głowicy od ruchomej warstwy magnetycznej wynosząca 0 ÷ 5 μm powoduje, że materiałom stosowanym na obudowy głowic stawiane są specjalne wymagania. Materiały te powinny mieć takie cechy, jak:

- wysoka odporność na zużycie ścierne,
- wysoka odporność na zatarcie,
- dobra obrabialność, pozwalająca uzyskać wysoką gładkość powierzchni,
- jednorodna struktura,
- mała porowatość,

- duża wytrzymałość mechaniczna na zginanie i na udary,
- stabilność geometryczna,
- odporność na korozję,
- mała oporność elektryczna,
- dobra przewodność cieplna,
- niska cena.

Znaczny wpływ na wartości użytkowe głowic, poza samym materiałem, mają także własności jego warstwy wierzchniej. Spośród różnych cech użytkowych głowic najczęściej wymaganymi cechami związanymi ze stanem warstwy wierzchniej elementów nośnych (obudowy) są:

- odporność na zatarcie,
- zdolność przejmowania obciążeń bez plastycznych odkształceń powierzchniowych,
- wysoka odporność na zużycie ścierne,
- duża wytrzymałość zmęczeniowa powierzchniowa,
- mały opór tarcia współpracujących powierzchni.

Odporność na zatarcie jest podstawowym warunkiem długotrwałej i bezawaryjnej współpracy głowicy latającej z nośnikiem informacji. Jest to odporność głowicy na tworzenie się na jej warstwie wierzchniej (nośnej) narostów pochodzących ze startej warstwy nośnika. Badania zjawiska zacierania się głowic prowadzone przez czołowe firmy produkujące pakiety dyskowe (Memorex, Scotch) wykazały, iż zatarcie zapoczątkowane jest uderzeniem głowicy o nośnik (bezpośrednio lub pośrednio przez ciało obce). Warstwa wierzchnia zostaje uszkodzona - odkształcona plastycznie. Starte cząstki nośnika informacji (tlenki żelaza) osadzają się na uszkodzonej powierzchni i na skutek tarcia w mikroobszarach uszkodzonej warstwy wierzchniej głowicy wytwarza się wysoka temperatura powodująca spiekanie osadzonych cząstek. Prowadzi to do zniszczenia nośnika informacji.

Złożony charakter zjawiska zacierania się głowic latających powoduje, że materiałom przeznaczonym na obudowy stawia

się wysokie wymagania. Również porowatość warstwy wierzchniej elementu nośnego głowicy może być jedną z przyczyn zacierania się głowic, gdyż w mikrowgłębieniach mogą się osadzać cząstki ciał stałych z otoczenia oraz starte cząstki nośnika powodując powstawanie narostów. Należy w tym przypadku uwzględnić działanie skrawające ostrych krawędzi porów.

Porowatość rzutuje na wytrzymałość mechaniczną na zginanie i udarność elementów wykonanych z danego materiału. Duża wytrzymałość mechaniczna wymagana jest także ze względów technologicznych. Jednorodna struktura materiału pod względem składu chemicznego i fazowego warunkuje uzyskanie żądanych własności w całym przekroju elementu.

Stabilność geometryczna a także odporność materiału na korozję i erozję warunkuje utrzymanie stałej odległości między współpracującymi elementami oraz zapewnia trwałość połączeń między elementami głowicy.

Dotychczas nie ma materiału o wszystkich wyżej wymienionych cechach. Dlatego też określa się cechy decydujące i ze względu na nie dobiera się materiał na głowice, starając się o jak najlepszy kompromis innych cech.

Wymagania stawiane materiałom na obudowy głowic najpełniej spełniają nowe materiały ceramiczne (quasiceramiczne).

3. PRZEGLĄD WYBRANYCH MATERIAŁÓW CERAMICZNYCH

Strukturalnie ceramika stanowi układ dwu lub więcej faz polikrystalicznych i bezpostaciowych oraz pewnej ilości wtrąceń gazowych - porów. Wzajemny stosunek fazy krystalicznej i bezpostaciowej decyduje o jej właściwościach fizykochemicznych i elektrycznych, przy czym główną rolę gra tu faza krystaliczna. W ostatnich latach zwiększa się pole zastosowań ceramiki (także w EMC) oraz zakres i poziom wymagań. Opracowywane są nowe technologie wytwarzania, jak prasowanie na gorąco pod ciśnieniem ok. 500 kG/cm^2 i w próżni w temperatu-

Tabela 1

Zestawienie niektórych właściwości wybranych materiałów ceramicznych oraz ferrytu i stali

Nazwa Właściwość	Ultraporcelana Al 80	Alund Al 19	Steatyt	Ceramika cyrkonowa	Agalit	Ferryt gęsty	Stal "55" (niehartowana)
Składniki podstawowe	BaO, Al_2O_3, SiO_2	BaO, Al_2O_3, SiO_2	MgO, Al_2O_3, SiO_2	ZrO_2, SiO_2	SiO_2, Al_2O_3, LiO_2, K_2O	Fe_2O_3, NiO, ZnO	Fe, C, Mn, Si
Wytrzymałość na zginanie [Rg] kG/mm^2	10 ÷ 15	⩾ 20	14 ÷ 16	ok. 19	ok. 45	ok. 3	66 (Rm) *)
Udarność [Rn] $kGcm/cm^2$	3 ÷ 5	4 ÷ 6	3 ÷ 5	-	-	-	-
Twardość w skali Mohsa	8 ÷ 9	9	7 ÷ 8	8	6	6	ok. 4
Nasiąkliwość wodna,%	0 ÷ 0,05	⩽ 0,02	0,0	0,0	-	-	-
Odporność na zmiany temperatury Δt^oC	150	180	110	-	360	-	-
Współczynnik rozszerzalności cieplnej α $20 \div 600^oC$ $1/^oC \times 10^{-6}$	4 ÷ 6	6 ÷ 8	7 ÷ 9	8 ÷ 9 dla zakresu temp. $20 \div 100^oC$	12	9	14

*) Rm - wytrzymałość doraźna na rozciąganie

rze od 1000 do 2000°C, czy proces kontrolowanej krystalizacji (dewitryfikacji). Pozwoliły one na uzyskanie materiałów ceramicznych o minimalnej porowatości i drobnokrystalicznej strukturze.

Ze względu na wymagania stawiane materiałom do budowy głowic na uwagę zasługują takie materiały jak ceramika alundowa (korundowa) i quasiceramy (np. agalit). Do pełniejszej oceny i wyboru materiału wzięto pod uwagę także i inne materiały ceramiczne (tabela 1).

4. METODYKA I ZAKRES PRAC DOŚWIADCZALNYCH

Metodykę pracy oparto w głównej mierze na doświadczeniach z obróbki ferrytów w zakresie ścieralności i obróbki gładkościowej. Jako podstawę do oceny i wyboru programu badań próbek przyjęto twardość, odporność na ścieranie oraz możliwość uzyskania gładkości powierzchni odpowiadającej 14 klasie chropowatości na drodze docierania.

4.1. Mikrotwardość

Pomiary mikrotwardości próbek materiałów ceramicznych przeprowadzono metodą Vickersa za pomocą mikrotwardościomierza typu PMT-3. Gładkość powierzchni pomiarowych wykonano w 8 ÷ 9 klasie chropowatości. Na każdej próbce przeprowadzono 5 pomiarów. Wyniki pomiarów przedstawiono w tabeli 2.

Tabela 2

Wyniki pomiarów mikrotwardości badanych próbek materiałów

Materiał	Średnia mikrotwardość (μHV) kG/mm^2
Al 80	800
Al 19	1500
Steatyt	770
Ceramika cyrkonowa	1080
Agalit	700
Ferryt gęsty	750
Stal "55" niehartowana	390

4.2. Odporność na zużycie

Bardzo ważną cechą materiału jest jego wytrzymałość powierzchniowa rzutująca na odporność na zużycie wskutek tarcia. Cecha ta związana jest z odrywaniem się cząstek od powierzchni materiału w procesie tarcia pod działaniem ściernym materiału współpracującego np. w wyniku działania wyraźnie twardszych mikronierówności jednego z ciał współpracującej pary głowica - nośnik. W związku z tym przeprowadzono próby ścieralności 6 materiałów ceramicznych oraz stali "55" niehartowanej. Ścieralność powierzchni próbek badano metodą Skoda-Savin oraz dodatkowo przeprowadzono porównanie próby ścieralności za pomocą tulejki.

4.2.1. Metoda Skoda-Savin

Próby ścieralności przygotowanych próbek materiałów przeprowadzono na maszynie Skoda-Savin wg typowej metodyki ujętej normą PN-67/M-04306. Próbę prowadzono przy obciążeniu krążka z węglików spiekanych, P = 2 kG, przy prędkości obrotowej krążka n = 1820 obr/min i całkowitej liczbie obrotów n_c = 2000 obr. Jako chłodziwo w czasie próby ścierania stosowano 0,5% roztwór wodny K_2CrO_3. Na każdej próbce przeprowadzono po 6 wytarć-śladów. Wyniki pomiarów przedstawiono w tabeli 3.

4.2.2. Metoda wykonywania wytarć tulejką

W doświadczeniach własnych, oprócz prób odporności na zużycie ścierne na maszynie Skoda-Savin, przeprowadzono również próby odporności na zużycie ścierne za pomocą prostego oprzyrządowania przedstawionego na rys. 1.

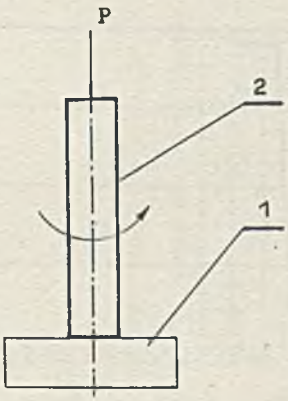
Próbki w kształcie płaskich płytek były obrabiane maszynowo na docieraczce w celu wykonania powierzchni pomiarowych - bazy. Badanie polegało na wykonaniu wytarć na płaskiej po-

Tabela 3

Wyniki pomiarów ścieralności płytek ceramicznych i stali "55" na maszynie Skoda-Savin

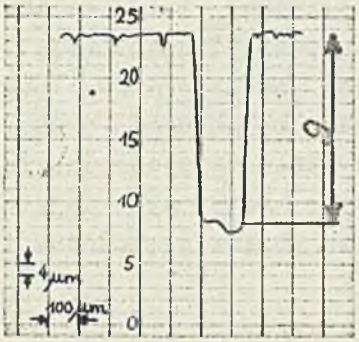
Materiał	Objętość wytarcia $V_L/mm^3 \cdot 10^{-3}$/						Średnia objętość wytarcia $V_L śr/mm^3 \cdot 10^{-3}$/
	1	2	3	4	5	6	
Al 80	100,5	123,9	96,9	121,2	138,3	-	116,1
Al 19	10,76	8,689	11,65	13,37	8,689	8,689	10,03
Steatyt	144,80	167,50	167,50	244,1	152,2	183,0	176,05
ZrO_2	13,37	11,16	7,11	7,49	5,195	5,195	8,27
Agalit	52,97	35,48	23,46	31,36	59,37	87,75	48,5
Ferryt gęsty	77,58	107,1	247,5	67,91	318,3	49,15	144,6
Stal"55"	109,30	59,37	66,6	88,54	140,5	171,5	106,0

wierzchni płytki za pomocą cienkościennej tulei ze stali "55" hartowanej. Jako ścierniwa użyto pasty diamentowej 1 ÷ 5 µm. Warunki badania: obciążenie tulei stałe - 1 kG, prędkość obrotowa tulei - 1400 obr/min, czas pojedynczej próby - 0,5 min. Dla każdego materiału wykonano 4 wytarcia. Tulejki były używane jednorazowo dla każdego materiału. Objętość startego materiału wyznaczono przez pomiar głębokości i szerokości starcia.

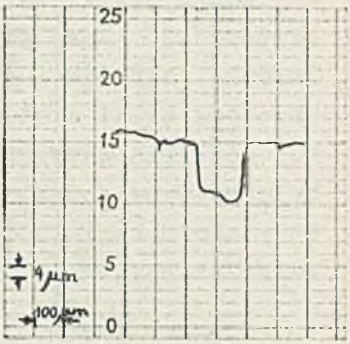


Rys. 1. Schemat badania zużycia za pomocą tulei
1 - próbka badana,
2 - docierak-tulejka,
P - obciążenie

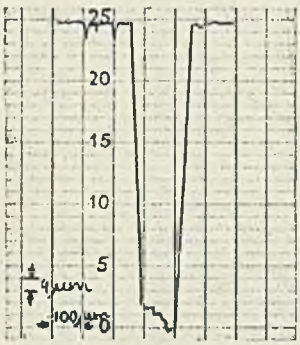
Pomiary głębokości wytarć wykonywano na profilografometrze Talysurf (rys. 2). Za wyniki przyjęto średnią arytmetyczną wykonanych pomiarów wytarć (tabela 4). Obciążenie i drogę tarcia dobrano w taki sposób, że głębokość starcia g (rys. 2) zawierała się w granicach 0,008 ÷ 0,1 mm.



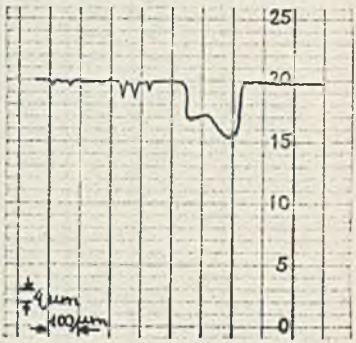
Al 80



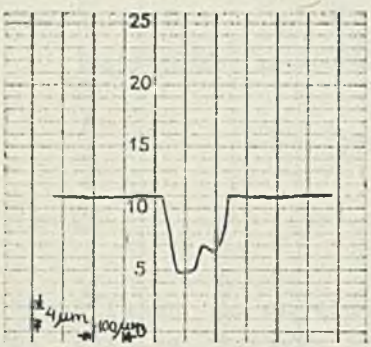
Al 19



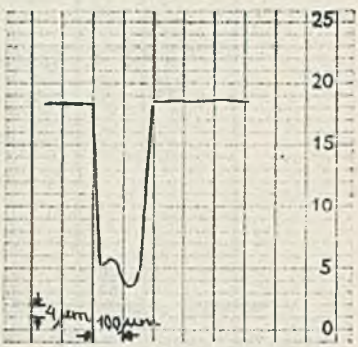
Steatyt



ZrO_2



Agalit



Ferryt

Rys. 2. Profilogramy nacięć

Tabela 4

Wyniki pomiarów ścieralności płytek ceramicznych stali "55" tulejką

Materiał	Średnia objętość wytarcia - śladu ($mm^3 \cdot 10^{-3}$)
Ultraporcelana Al 80	575
Alund Al 19	152
Steatyt	910
ZrO_2	128
Agalit	205
Ferryt gęsty	520

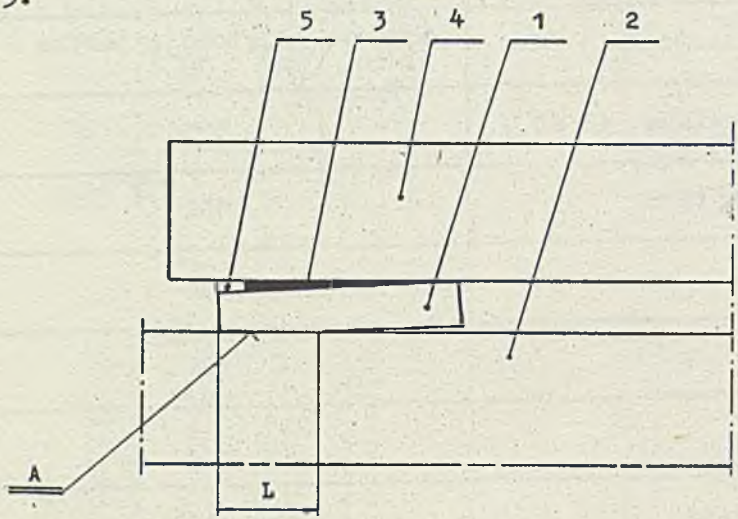
4.3. Obrabialność materiałów ceramicznych

Stopień obrabialności oceniano według wskaźnika jakim jest ubytek materiału w czasie docierania oraz uzyskana gładkość powierzchni obrobionej. Docieranie jest podstawowym sposobem obróbki gładkościowej ceramiki i ferrytów (materiałów twardych). Głównymi czynnikami mającymi wpływ na przebieg i wyniki procesu docierania są: materiał docieraka, skład pasty ściernej, wielkość nacisku powierzchniowego oraz szybkość ruchu - skrawania. Dla jednoznaczności wyników zastosowano obróbkę maszynową, w czasie której proces docierania łącznie z wywieraniem nacisku powierzchniowego jest wykonywany przez obrabiarkę - docierarkę.

Sposób przeprowadzania badań:

Materiał wyjściowy pocięto na płytki prostopadłościenne o jednakowej grubości piłą z tarczą diamentową. Zmierzono chro-

powatość powierzchni obrobionych płytek (tab. 6). Następnie docierano płytki na docierarce z tarczą metalową (żeliwną). Sposób mocowania płytek do tarczy przyciskowej pokazano na rys. 3.



Rys. 3. Sposób mocowania płytek przy docieraniu

1 - przedmiot obrabiany - płytka, 2 - docierak, 3 - spoiwo, 4 - tarcza przyciskowa, 5 - folia dystansowa

A - powierzchnia dotarta, L - długość powierzchni dotartej

Dla każdego materiału obrabiano jednocześnie 3 płytki umocowane do tarczy dociskowej co 120°. Jako kryterium obrabialności przez docieranie przyjęto wielkość powierzchni dotartej A (rys. 3) A = Lx b - gdzie b jest szerokością płytki. Za wyniki przyjęto średnie arytmetyczne wartości wykonanych pomiarów powierzchni dotartej (tabela 5).

Zmierzono chropowatość dotartych powierzchni. Wyniki pomiarów na profilografometrze podano w tabeli 6. Profilogramy powierzchni dotartych podano na rys. 4 ÷ 6.

Tabela 5

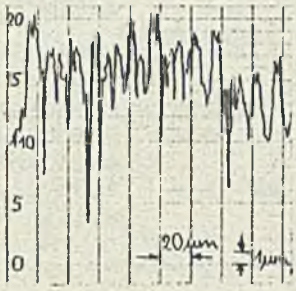
Obrabialność powierzchni przez docieranie

Materiał	Powierzchnia dotarta Lxb mm^2	Czas docierania min	Obrabialność mm^2/min	Obrabialność przy założeniu 100% dla stali "55" %
Ultraporcelana Al 80	57,4	7,5	7,7	68,5
Alund Al 19	33,0	30	1,1	9,8
Steatyt	86,0	7,5	11,5	10,3
Ceramika cyrkonowa	20,7	30	0,69	6,2
Agalit	36,0	15	2,4	21,4
Ferryt gęsty	159,6	7,5	21,3	190
Stal "55" niehartowana	83,8	7,5	11,2	100

Rys. 4. Profilogramy powierzchni próbek

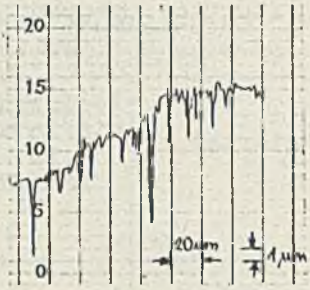
a - obrobionych tarczą diamentową, b - dotartych na docierarce z tarczą metalową (żeliwną)

a)

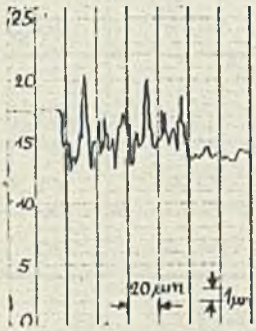


Al 80

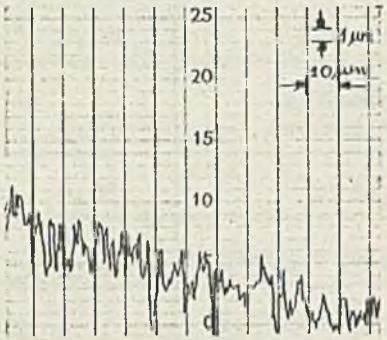
b)



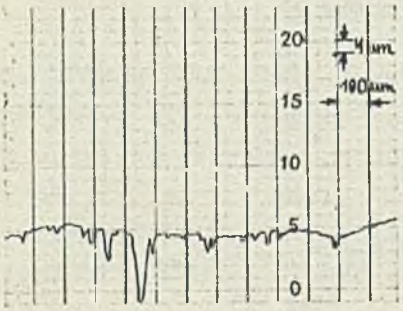
Al 80



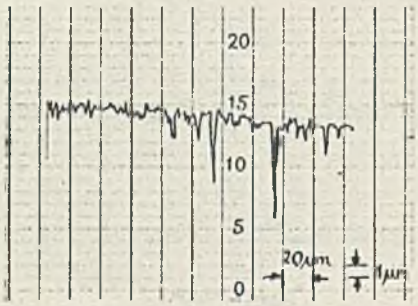
Al 19



Al 19



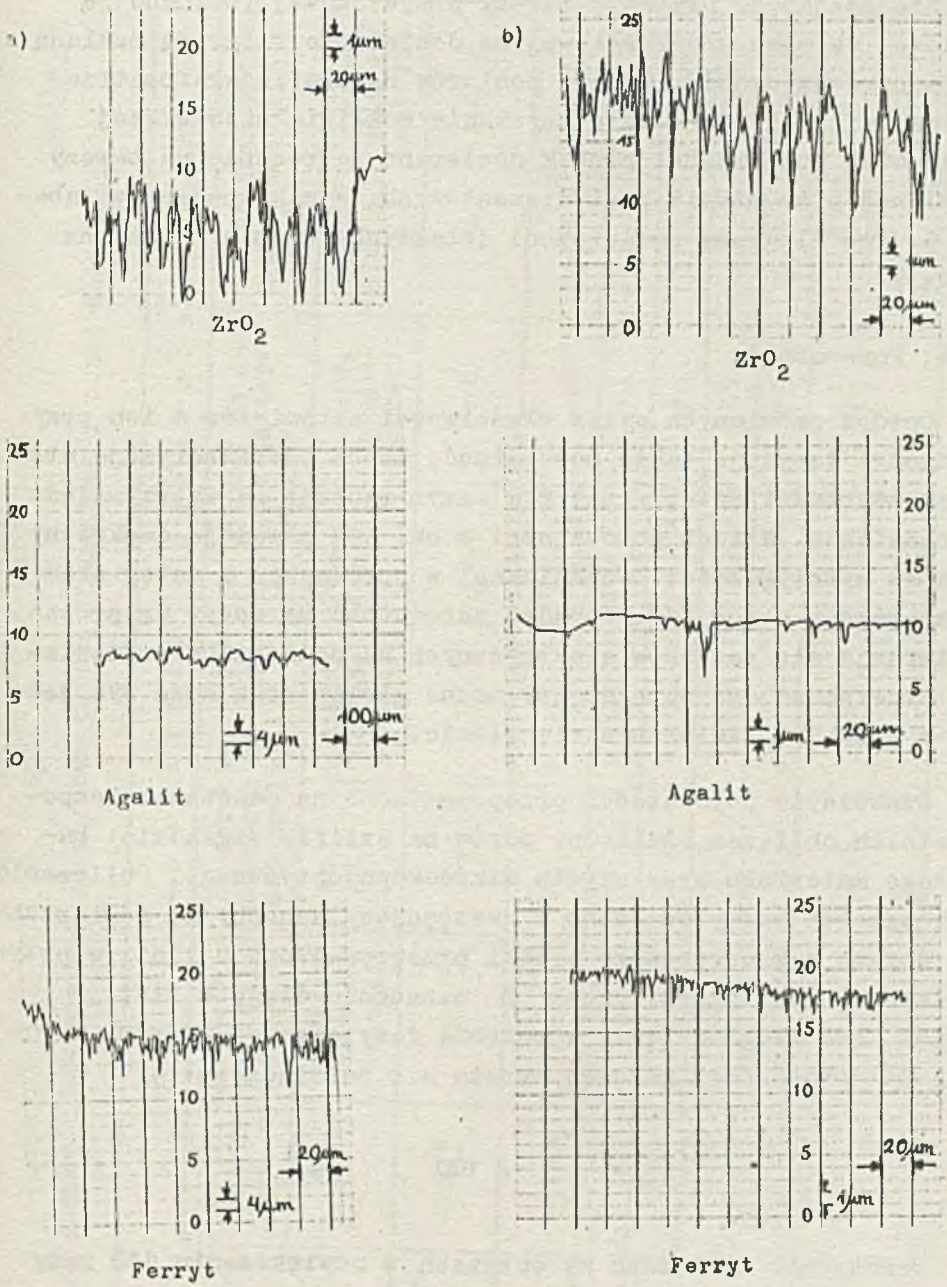
Steatyt



Steatyt

Rys. 5. Profilogramy powierzchni próbek

a - obrobionych tarczą diamentową, b - dotartych na docierarce z tarczą metalową (żeliwną)



Rys. 6. Profilogramy powierzchni próbek

a - obrobionych tarczą diamentową, b - dotartych na docierarce z tarczą metalową (żeliwną)

Nie zdejmując płytek z tarczy przyciskowej, poddano je dalszej obróbce gładkościowej na docierarce z tarczą szklaną w jednakowym czasie. Wyniki pomiarów na profilografometrze podano w tabeli 6. W celu uzyskania możliwie największej gładkości powierzchni próbek docierano je ręcznie na tarczy szklanej przy użyciu past diamentowych. Wyniki podano w tabeli 6. Profilogramy powierzchni dotartych ręcznie podano na rys. 7.

4.4. Porowatość

Oprócz omówionych wyżej właściwości materiałów o ich przydatności decyduje także porowatość, która charakteryzuje stopień zwartości materiału i tym samym rzutuje na wytrzymałość mechaniczną. Wzrost porowatości o ok, 10% powoduje dwukrotny spadek wytrzymałości mechanicznej w porównaniu z materiałem pozbawionym porów [1]. Ponadto porowatość ma wpływ na proces zbierania się ładunków elektrycznych na powierzchni materiału przeznaczonego na elementy nośne głowic oraz może być jedną z przyczyn zacierania się głowic.

Oznaczenie porowatości przeprowadzono na podstawie bezpośrednich obliczeń wielkości porów na szlifie (zgładzie) badanego materiału przy użyciu mikroskopu optycznego. Obliczenie porowatości przeprowadzono w następujący sposób: na powierzchni obrazu mikroskopowego próbki przeprowadzono 8 linii w przypadkowych kierunkach. Przez L oznaczono długość linii, przez La długość linii w obrębie fazy gazowej (porów) obrazu. Zawartość fazy gazowej wyraża się zależnością:

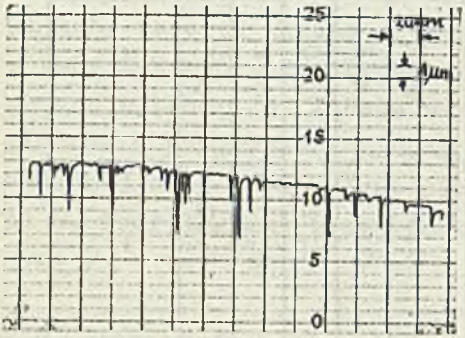
$$V = \frac{L_a}{L} \cdot 100 \qquad (2)$$

Porowatość wyliczono na obrazach w powiększeniu 130 razy (rys. 8), wyniki obliczeń przedstawiono w tabeli 7.

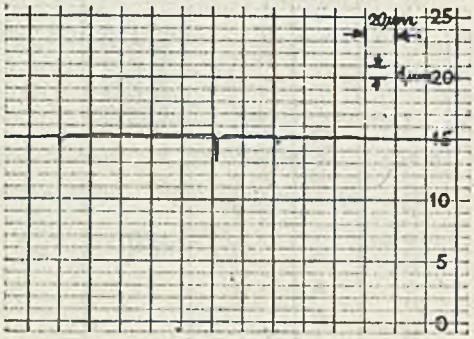
Tabela 6

Chropowatość powierzchni po obróbce tarczą diamentową i przez docieranie

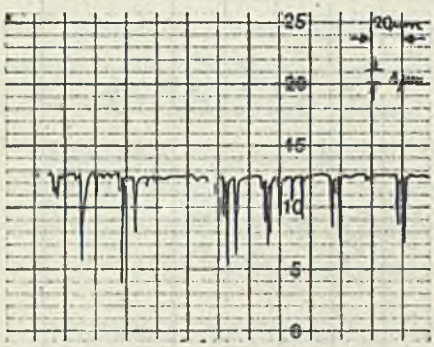
Materiał	Chropowatość powierzchni wg PN							
	Po obróbce tarczą diamentową pow. wyjściowa		Po obróbce na docierarce na tarczy żeliwnej		Po obróbce na docierarce na tarczy szklanej		Po docieraniu ręcznym ostatecznym	
	Ra µm	oznacz. chropowatości	Ra µm	oznacz. chropowatości	Ra µm	oznacz. chropowatości	Ra µm	oznacz. chropowatości
Ultraporcelana Al 80	1,7	▽ 6	0,9	▽7	0,6	▽8	0,14	▽10
Alund Al 19	1,3	▽ 6	0,7	▽7	0,13	▽10	0,03	▽ 12
Steatyt	1,3	▽ 6	1,1	▽7	0,5	▽ 8	0,4	▽ 8
ZrO_2	1,9	▽6	1,3	▽6	0,33	▽ 8	0,12	▽10
Agalit	1,5	▽6	0,1	▽10	0,05	▽11	0,01	▽14
Ferryt	1,4	▽6	0,5	▽ 8	0,31	▽9	0,02	▽13
Stal "55" nieharto-wana	1,6	▽6	0,3	▽ 9	0,2	▽9	-	-



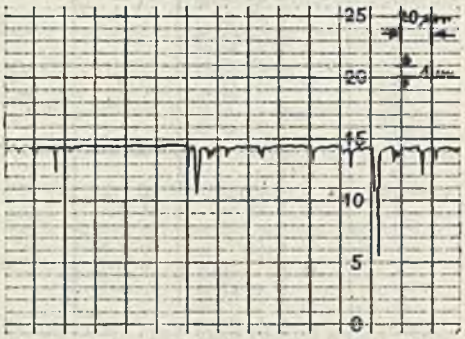
Al 80



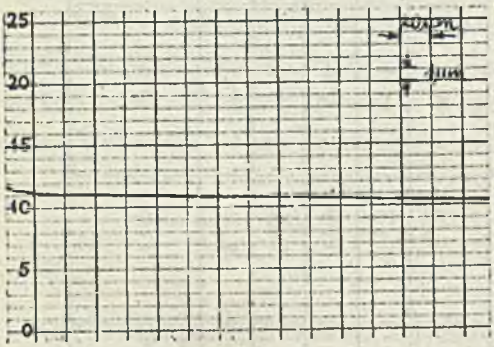
Al 19



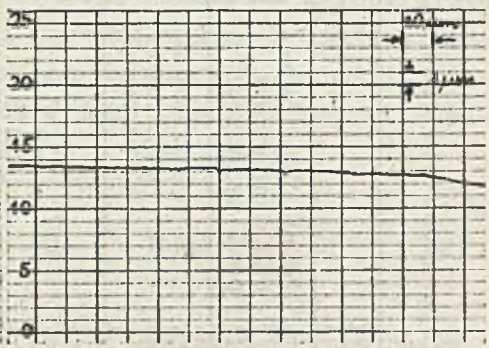
Steatyt



ZrO_2

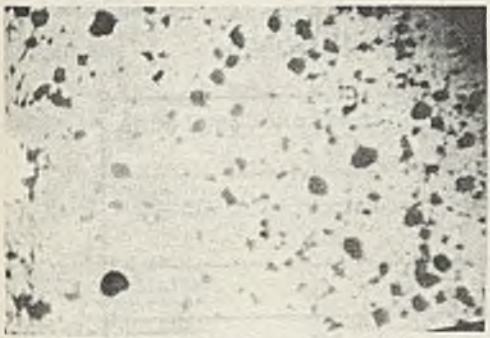


Agalit

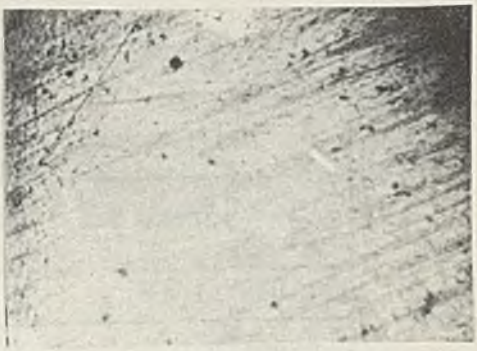


Ferryt

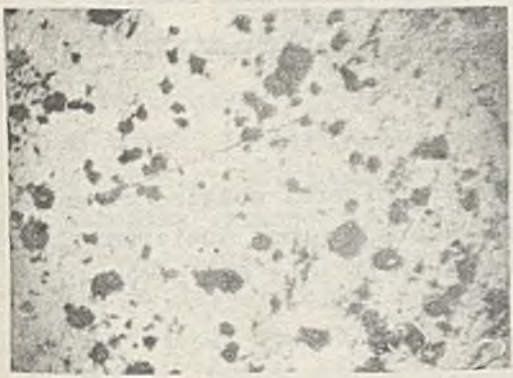
Rys. 7. Profilogramy powierzchni dotartej ręcznie (po obróbce ostatecznej) 1 działka = 1 µm



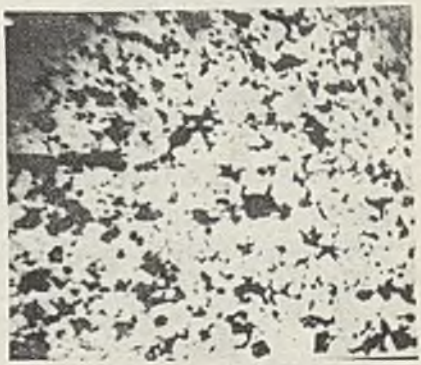
AL 80



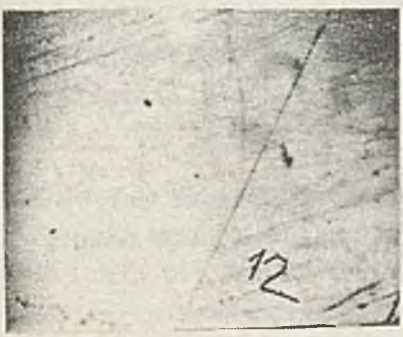
AL 19



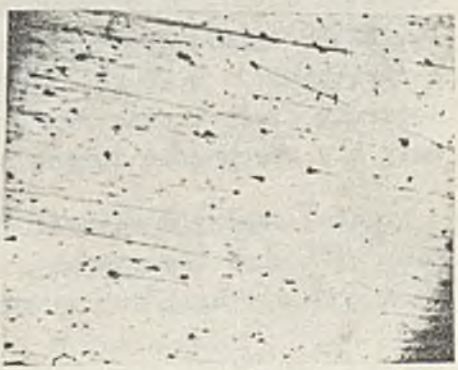
Steatyt



ZrO_2



Agalit



Ferryt

Rys. 8. Zdjęcia próbek otrzymane w mikroskopie metalograficznym przy powiększeniu 130 razy

Tabela 7

Porowatość materiałów ceramicznych

Materiał	Średnia porowatość względna (%)
Ultraporcelana Al 80	20
Alund Al 19	5
Steatyt	40
Ceramika cyrkonowa	39
Agalit	2
Ferryt gęsty	8

5. OMÓWIENIE WYNIKÓW BADAŃ

Badaniom ścieralności, obrabialności oraz możliwości uzyskania powierzchni ▽ 14 (zwierciadlanej) przez docieranie, poddanych było 5 materiałów ceramicznych oraz dla porównania ferryt gęsty i stal "55" niehartowana.

Porowatość

Najmniejszą liczbę porów z badanych materiałów posiadał agalit i alund. Na obserwowanych przekrojach próbek z agalitu, Al 19 i ferrytu widoczne są pory o kształcie kulistym i o wielkości 10 ÷ 20 μm. W przypadku ferrytu powierzchnie po dotarciu miały liczne rysy. Pozostałe materiały charakteryzowały się obecnością dużych porów (ok. 50 μm).

Wyniki pomiarów porowatości mogą być obarczone pewnym błędem, gdyż obok właściwych porów mogą znajdować się dziury po wyrwanych w czasie docierania ziarnach.

Twardość

Szczególnie wysoką twardością odznaczają się dwa spośród badanych materiałów, tj. Al 19 i ceramika cyrkonowa. Pozostałe materiały ceramiczne mają twardość porównywalną z ferrytem gęstym. Wyniki pomiarów mogą być obarczone błędem spowodowanym trudnym odczytem śladów wgłębnika mikrotwardościomierza na twardych materiałach jak: Al 19 i ceramika cyrkonowa.

Ścieralność

Spośród badanych materiałów wyjątkowo wysoką odpornością na zużycie przez tarcie odznacza się ceramika cyrkonowa i Al 19, są one 10-krotnie odporniejsze na ścieranie od ferrytu i stali "55" niehartowanej. Także agalit wykazuje dużą odporność na ścieranie - ok. trzykrotnie większą od ferrytu.

Wyniki uzyskane przy ścieraniu na maszynie Skoda-Savin i przy użyciu tulejki wskazują największą odporność na ścieranie ceramiki cyrkonowej, Al 19 oraz agalitu, przy tym Al 19 i ceramika cyrkonowa miały największą twardość. Według badań W. Bottenberga [5] istnieje pewna tendencja do wzrostu odporności na zużycie przez tarcie w miarę wzrostu twardości materiału, jednak twardość nie określa jednoznacznie odporności na zużycie przez tarcie. Tłumaczy się to tym, że wraz ze wzrostem twardości zmniejsza się zagłębienie w zużywany materiał ziaren lub chropowatości ściernych, dzięki czemu maleje zużycie spowodowane skrawaniem ostrzami ziaren lub chropowatości oraz zmniejsza się odpowiednio odkształcenie plastyczne towarzyszące skrawaniu i wygniataniu bruzd.

Obrabialność i gładkość

Materiały najtwardsze tj. Al 19 i ceramika cyrkonowa są trudno obrabialne. W przypadku Al 19 uzyskano wysoką gładkość obrobionej powierzchni odpowiadającej 12 klasie chropowatości, a dla ceramiki cyrkonowej 10 klasę chropowatości. Także trudny do obróbki okazał się agalit. Wydajność jego obróbki jest o

rząd wielkości mniejsza od ferrytu, mimo porównywalnych twardości. Jednakże na próbkach agalitu uzyskano najwyższą gładkość powierzchni obrobionej odpowiadającą 14 klasie chropowatości. Dla ferrytu uzyskano gładkość odpowiadającą klasie 13.

Stosunkowo łatwo obrabialne materiały jak steatyt i Al 80 nie pozwalały na uzyskanie powierzchni o dobrej jakości, co związane jest z ich znaczną porowatością. W czasie obróbki ściernej tych materiałów następuje otwieranie się porów, co wpływa na zwiększenie chropowatości powierzchni.

6. WNIOSKI

Na podstawie omówionych wyżej wyników badań można przyjąć, że najbardziej odpowiednim materiałem na elementy nośne względnie obudowy głowic jest tworzywo szklanokrystaliczne - agalit, które odznacza się pożądanymi właściwościami, a mianowicie: dużą odpornością na ścieranie, bardzo małą porowatością, łatwością uzyskania powierzchni o bardzo wysokiej gładkości (∇14) (przez docieranie). Materiałem o jeszcze większej (ok. 4-krotnie) odporności na zużycie ścierne niż agalit jest Al 19. Jednak w porównaniu z agalitem jest on bardziej porowaty oraz trudniejszy do obróbki i pozwala na uzyskanie powierzchni o chropowatości odpowiadającej klasie ∇12. Gdyby udało się zmniejszyć jego porowatość, co jest częściowo możliwe przez zaostrzenie procesu wytwarzania, to można by uzyskać jeszcze lepszą jakość obrobionej powierzchni.

Literatura

[1] KINGERY V.D.: Vvedenije v keramiku, Moskwa, 1964.

[2] JACKSON B., FORD W.F., WHITE J.: The Influence of Cr_2O_3 on the Wetting of Periclase Grains by Liquid Silicate, Trans. Brit. Cer. Soc. (7), nr 62, 1963.

[3] RYSHKEEWITSCH E.: Keramographie - Begriff und Anwendung. Sprechsal fur Keramik. Glass, Email, nr 20, 1955.

[4] GÓRSKI E.: Obróbka gładkościowa, WNT, 1970

[5] BOTTENBERG W.: Die Neue Gisserei, nr 36, 1949, s. 39.

ИССЛЕДОВАНИЕ КЕРАМИЧЕСКИХ МАТЕРИАЛОВ, ПРИМЕНЯЕМЫХ В МАГНИТНЫХ ГОЛОВКАХ

Резюме

В работе обсуждаются исследования группы керамических материалов с точки зрения их применения в магнитных головках. Исследовались такие свойства материалов, как сопротивляемость абразивному износу, обрабатываемость, твердость, пористость, а также чистота поверхности, обрабатываемой притиркой. Испытания сопротивляемости износу были проведены по методу Севина при использовании стальной втулки. Оценка обрабатываемости была проведена согласно указателю убытка материала во время притирки и чистоте обработанной поверхности.

Измерение твердости было проведено на микроизмерителе твердости ПМТ-3.

Пористость была обозначена на основе непосредственного вычисления величины пор на шлифе исследуемого материала с помощью оптического микроскопа.

В исследуемом материале особенно хорошие качества были обнаружены в агалите, который обладает большой сопротивляемостью абразивному износу, что позволяет получить очень чистую поверхность, отвечающую 14 классу шероховатости.

INVESTIGATION OF POTTERY MATERIALS THAT CAN BE APPLIED TO MAGNETIC HEADS

Summary

The paper discusses the investigations carried out in a group of pottery materials from the point of view of their application to magnetic heads.

The investigations embraced such material properties like resistance to abrasive wear, workability, hardness, porosity and smoothness of the surface worked, by rubbing up. Resistance tests to wear were carried out by Savin's method using a steel sleeve. Workability was estimated according to the index of material loss while rubbing up, and the smoothness of the worked surface. Hardness measurements were made on a PMT-3 microhardness measurement device. Porosity was defined on the basis of pore magnitude direct computations carried out in the tested material cut by means of an optic microscope.

Agalite proved its special value in the examined material because of its great resistance to abrasion and the surface smoothness corresponding to 14 class of coarseness.

Prace IMM
Zeszyt 1

681.322.06.004.14:
621.38.019.34

SZACOWANIE I ANALIZA PROGRAMÓW NIEZAWODNOŚCIOWYCH ZA POMOCĄ PROGRAMU "LAMBDA".

Jerzy DĄBROWSKI

Pracę złożono 7.11.1973

Omówiono wykorzystanie maszyny cyfrowej do szacowania niektórych parametrów niezawodnościowych urządzeń elektronicznych. Zrealizowany na EMC ODRA 1304 i ODRA 1305 program "LAMBDA" pozwala na oszacowanie średniego okresu międzyawaryjnego oraz analizę urządzenia i jego podzespołów pod względem niezawodnościowym. Algorytm programu "LAMBDA" stanowi rozszerzenie znanej i stosowanej od dawna metody szacowania niezawodności urządzeń elektronicznych, a wykorzystanie programu "LAMBDA" pozwala na uproszczenie obliczeń oraz przeprowadzenie analizy niezawodnościowej.

Program "LAMBDA" powstał w celu usprawnienia procesu obliczania średniego okresu międzyawaryjnego układów oraz urządzeń elektroniczno-mechanicznych o szeregowym modelu niezawodnościowym. Ze względu na prostotę algorytmu i możliwość szybkiego automatycznego obliczania, program cechuje duża skala zastosowań. Najistotniejsza jest możliwość wykorzystania programu w procesie konstrukcyjnym urządzeń lub fragmentów urządzeń w celu zbadania ich parametrów niezawodnościowych. Daje to możliwość przeanalizowania w krótkim czasie wielu wersji i wybrania najodpowiedniejszej. Wyniki programu można również wykorzystać do porównania z wynikami badań eksploatacyjnych.

W programie obliczane są współczynniki intensywności uszkodzeń i średnie okresy międzyawaryjne układów na podstawie zadanych współczynników intensywności uszkodzeń elementów podstawowych oraz znajomości struktury układów (z uwzględnieniem obciążenia elementów) [1].

Algorytm programu nie ogranicza typów elementów i urządzeń, natomiast ich charakter elektroniczny lub mechaniczny może być uwidoczniony w identyfikujących je nazwach.

Program źródłowy, napisany w języku ALGOL 60, został przetłumaczony za pomocą translatora XALM MK.15C na EMC ODRA 1305. Opracowano dwie wersje programu do wprowadzania danych przez czytnik kart perforowanych albo czytnik taśmy papierowej. Wyniki wyprowadzane są na drukarkę wierszową. Przedstawiony dalej przykładowy program zajmuje w pamięci operacyjnej ok. 10k słów, natomiast efektywny czas jego wykonywania wynosi poniżej 1 min.

Definicje

Przypomnimy teraz kilka pojęć, z których będziemy w dalszym ciągu korzystać.

Współczynnik intensywności uszkodzeń urządzenia (λ) :

$$\lambda = \frac{m}{\sum_{i=1}^{m} t_i} \qquad (1)$$

gdzie m - suma uszkodzeń urządzenia, t_i - czas między kolejnymi uszkodzeniami urządzenia.

Średni okres międzyawaryjny urządzenia (T):

$$T = \frac{1}{\lambda} \qquad (2)$$

Współczynnik obciążenia urządzenia (α):

$$\alpha = \frac{\lambda o}{\lambda_n} \tag{3}$$

gdzie λ_o - współczynnik intensywności uszkodzeń w rzeczywistych warunkach pracy urządzenia, λ_n - współczynnik intensywności uszkodzeń w nominalnych warunkach pracy urządzenia.

Współczynnik intensywności uszkodzeń zestawu N urządzeń (λ_z):

$$\lambda_z = \sum_{j=1}^{N} \lambda_j \delta_j \alpha_j \tag{4}$$

gdzie δ_j - liczba jednakowych urządzeń j, α_j - współczynnik obciążenia urządzenia j

Wprowadzimy również nowe pojęcie.

Procentowy wskaźnik intensywności uszkodzeń urządzenia j w zestawie ($P\{U_j, Z\}$):

$$P\{U_j, Z\} = 100 \ \delta_j \alpha_j \lambda_j \frac{1}{\lambda_z} \tag{5}$$

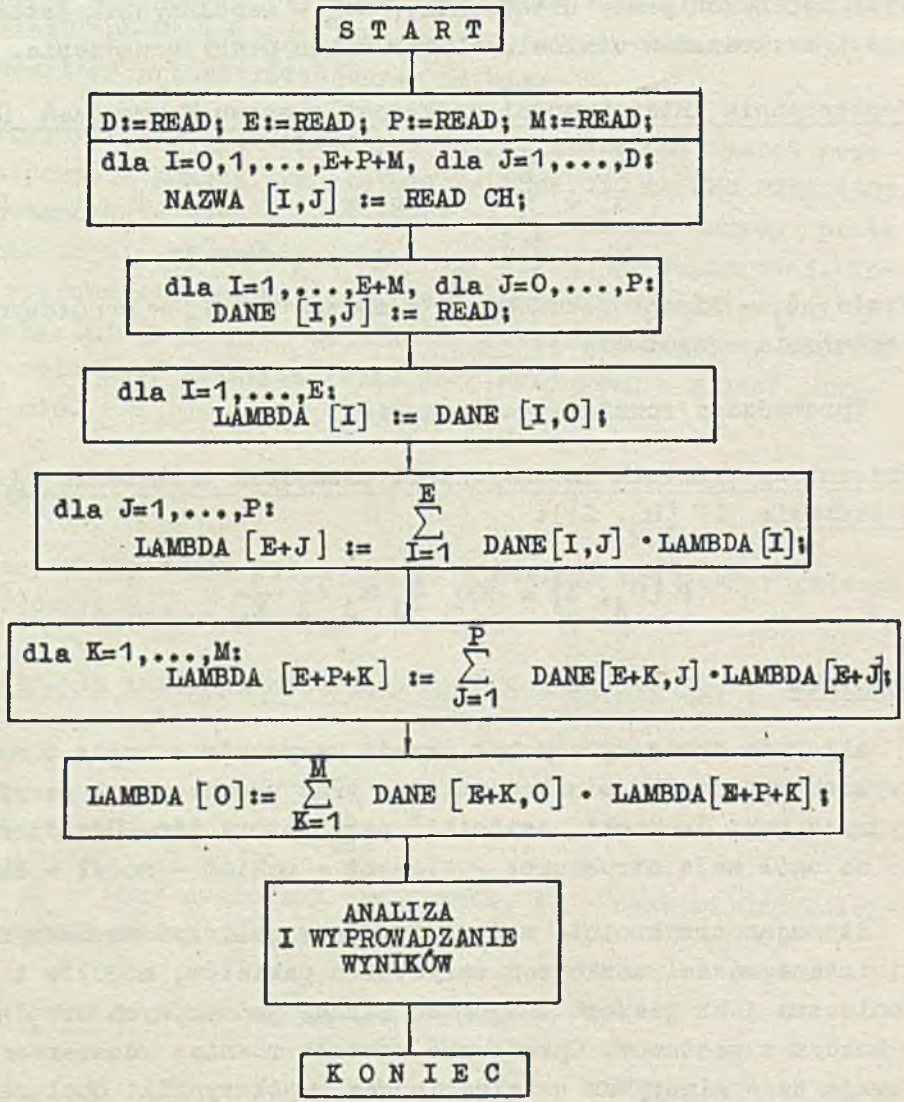
Algorytm

Algorytm opracowywany był przede wszystkim z myślą o maszynach cyfrowych, minikomputerach oraz urządzeniach peryferyjnych jak: drukarki, czytniki, perforatory itp. Urządzenia te na ogół mają strukturę: - element - pakiet - moduł - EMC.

Stosując trzykrotnie wzór (4) możemy obliczyć współczynniki intensywności uszkodzeń wszystkich pakietów, modułów i EMC. Konieczna jest jeszcze znajomość liczby jednakowych urządzeń w każdym z zestawów. Opracowana została również rozszerzona wersja tego algorytmu uwzględniająca współczynniki obciążenia elementów, pakietów i modułów. Zadawane są one z zewnątrz jako parametry (dane). Ze względu na trudność określenia współczynnika obciążenia modułów, a przede wszystkim pakietów,

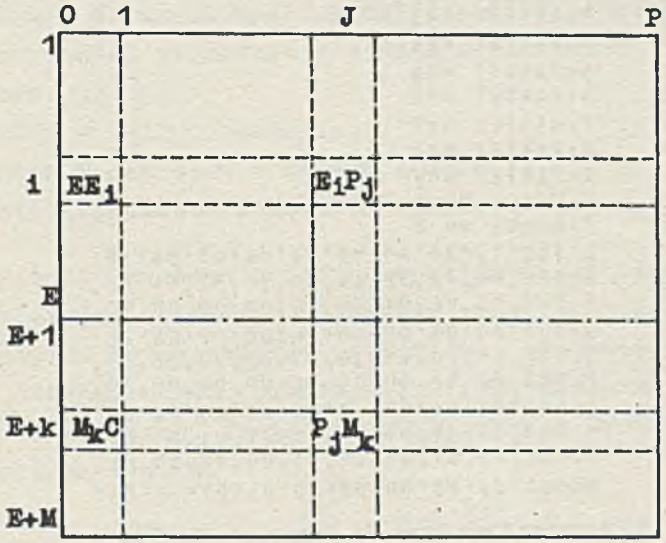
większe zastosowanie będzie miała zapewne przedstawiona tu wersja podstawowa, w której przyjęto α równe 1.

Schemat blokowy algorytmu przedstawia rys. 1.



Rys. 1. Schemat blokowy algorytmu programu "LAMBDA"

Algorytm pozwala na identyfikowanie elementów, pakietów, modułów i EMC przez nazwy. Liczbę symboli w nazwie (D) stanowi parametr wprowadzany do programu jako część danych. Równocześnie wprowadzane są: liczba elementów (E), pakietów (P) i modułów (M). Po wczytaniu tych parametrów zapamiętywane są nazwy wszystkich elementów, pakietów, modułów oraz EMC. Następnie wczytywane są parametry do tablicy DANE, która schematycznie przedstawiona jest na rys. 2.



Rys. 2. Schematyczna postać tablicy DANE [1:E+M, 0:P]

Oznaczenia:

EE$_i$ - przeskalowany współczynnik intensywności uszkodzeń elementu i; E$_i$P$_j$ - liczba elementów i w pakiecie j; P$_j$M$_k$ - liczba pakietów j w module k; M$_k$C - liczba modułów k w EMC

Ze względu na to, że współczynniki intensywności uszkodzeń elementów są na ogół liczbami z przedziału $\langle 10^{-6}, 10^{-10} \rangle$ wszelkie działania dokonywane są na liczbach $EE_i = \lambda_i \alpha_i 10^6$. Pozwala to na uniknięcie fałszowania wyników spowodowanego błędami zaokrągleń i reprezentacji liczb w słowie maszynowym.

Zestaw przykładowych danych przedstawia rys. 3.

```
 1    10,8,9,2,
 2    1,KOMPUTER X
 3    1,UK.SCALONY
 4    2,TRANZYSTOR
 5    3,DIODA DOG
 6    4,DIODA DZG
 7    5,OPORNIK
 8    6,KONDENSAT.
 9    7,STYK
10    8,OBW. DRUK.
11    1,PAKIET NR1
12    2,PAKIET NR2
13    3,PAKIET NR3
14    4,PAKIET NR4
15    5,PAKIET NR5
16    6,PAKIET NR6
17    7,PAKIET NR7
18    8,PAKIET NR8
19    9,PAKIET NR9
20    1,MODUL NR 1
21    2,MODUL NR 2
22    0.100,12,24,48,96,92,84,68,36,18,
23    0.050,88,76,52,04,08,16,32,64,82,
24    0.020,08,00,02,20,10,00,00,00,10,
25    0.030,00,08,00,00,00,00,00,00,20,
26    0.010,16,00,04,10,05,00,00,00,05,
27    0.020,00,16,00,00,00,00,00,00,40,
28    0.004,64,64,64,64,64,64,64,64,64,
29    0.800,01,01,01,01,01,01,01,01,01,
30    00002,01,01,01,01,01,00,00,00,00,
31    00001,00,00,00,00,00,01,01,01,01,
```

Rys. 3. Przykładowe dane do programu "LAMBDA"

Współczynniki intensywności uszkodzeń wszystkich pakietów, modułów oraz EMC obliczane są według wzorów:

współczynnik intensywności uszkodzeń pakietu j

$$PP_j = \sum_{i=1}^{E} E_i P_j \cdot EE_i \qquad (\text{dla } j = 1,\ldots, P),$$

współczynnik intensywności uszkodzeń modułu k

$$MM_k = \sum_{j=1}^{P} P_j M_k \cdot PP_j \qquad (\text{dla } k = 1,\ldots, M),$$

współczynnik intensywności uszkodzeń EMC

$$MC = \sum_{k=1}^{M} M_k C \cdot MM_k.$$

Równolegle obliczane są średnie okresy międzyawaryjne pakietów, modułów i EMC, na podstawie wzoru (2).

Analiza otrzymanych wyników prowadzona jest jednocześnie z wydrukami. W ramach analizy obliczane są procentowe wskaźniki intensywności uszkodzeń urządzeń w zestawach ($P\{U_j, Z\}$ - wg wzoru (5)) dla:

- elementów w każdym pakiecie,
- elementów i pakietów w każdym module,
- elementów, pakietów i modułów w EMC.

Dla każdego zestawu urządzeń (pakietów, modułów i EMC) wyniki zawierają informację o liczbie urządzeń i ich procentowym wskaźniku intensywności uszkodzeń w zestawie oraz współczynnik intensywności uszkodzeń i średni okres międzyawaryjny zestawu urządzeń. Fragment przykładowych wyników przedstawiony jest na rys. 4.

Uwagi z eksploatacji programu

Za pomocą programu "LAMBDA" dokonano wielu obliczeń dla niektórych polskich urządzeń cyfrowych. Wyniki programu pozwoliły ocenić, które z elementów, pakietów i modułów miały decydujący wpływ na niezawodność urządzenia, które należało zamienić lub zmodyfikować. Przydatność programu potwierdza również krótki czasokres przygotowania danych i dokonania obliczeń. Opracowanie i perforacja danych wraz z obliczeniami zajmują praktycznie 1 do 2 godz. Pozwala to na szybką i dokładną analizę urządzenia pod względem niezawodnościowym.

```
***********
KOMPUTER (
***********   ILOSC   INTENS. %

***********

MODUL NR 1      2     71.6  %

MODUL NR 2      1     28.4  %

***********

PAKIET NR1      2     11.0  %

PAKIET NR2      2     12.3  %

PAKIET NR3      2     13.4  %

PAKIET NR4      2     17.8  %

PAKIET NR5      2     17.1  %

PAKIET NR6      1      8.1  %

PAKIET NR7      1      7.4  %

PAKIET NR8      1      6.2  %

PAKIET NR9      1      6.9  %

***********

UK.SCALONY    750     58.9  %

TRANZYSTOR    650     25.5  %

DIODA  DOG     90      1.4  %

DIODA  DZG     35      0.8  %

OPORNIK        75      0.6  %

KONDENSAT.     72      1.1  %

STYK          896      2.8  %

DRW. DRUK.     14      5.8  %

***********

************************************************************

INTENSYWNOSC  USZKODZEN  L.M.C.  LAMBDA =  127 35400000  E-6

****************************
 S.C.M.U.   T = 7852   GODZIN
****************************

***********
UK.SCALONY   LAMBDA =    0.10000000  E-6
***********
TRANZYSTOR   LAMBDA =    0.05000000  E-6
***********
DIODA  DOG   LAMBDA =    0.04000000  E-6
***********
DIODA  DZG   LAMBDA =    0.03000000  E-6
***********
OPORNIK      LAMBDA =    0.01000000  E-6
***********
KONDENSAT.   LAMBDA =    0.04000000  E-6
***********
STYK         LAMBDA =    0.00400000  E-6
***********
DRW. DRUK.   LAMBDA =    0.30000000  E-6
***********
```

Rys. 4. Fragment przykładowych wyników programu "LAMBDA"

Literatura

[1] CALABRO S.R.: Reliability Principles and Practices, McGraw Hill Book Company, Inc., New York, Toronto, Londyn 1962.

[2] JURKIEWICZ A., MADEY J., PALUSZKIEWICZ A.: ALGOL 60, Wydawnictwa Uniwersytetu Warszawskiego, Warszawa 1969.

[3] ALGOL Magnetic Tape Compiler - 1900 Series, International Computers Limited - Technical Publications Service, Reading, 1969.

РАСЧЕТ И АНАЛИЗ ПРОГРАММ НАДЕЖНОСТИ ПРИ ПОМОЩИ ПРОГРАММЫ "LAMBDA"

Резюме

В труде описано использование цифровой вычислительной машины для расчета некоторых параметров надежности электронных устройств. Реализована на ЭВМ ODRA 1304 i ODRA 1305 программа "LAMBDA" позволяет рассчитать средний межаварийный период и проанализировать устройство и его узлы с точки зрения их надежности. Алгоритм программы "LAMBDA" представляет собой расширение уже известного и давно применяемого метода расчета надежности электронных устройств, а использование программы "LAMBDA" позволяет упростить расчет и провести анализ надежности.

ESTIMATION AND ANALYSIS OF RELIABILITY PROGRAMS BY "LAMBDA" PROGRAM

Summary

Using digital computer for estimation of some reliability parameters of electronic instruments is discussed. "LAMBDA" program, realized on computers ODRA 1304 and ODRA 1305, allows reliability estimation of mean time between failures and analysis of instrument and its elements. Algorithm of "LAMBDA" program is the extension of know and used long ago method of reliability estimation of electronic instruments. The use of "LAMBDA" program allows simplification of computations and performance of reliability analysis.

AUTORZY ARTYKUŁÓW ZAMIESZCZONYCH W ZESZYCIE 1 "PRAC IMM"

WYBRANE ZAGADNIENIA ZWIĄZANE ZE STRUKTURAMI DANYCH

Danuta KOŁACKA ukończyła studia na Wydziale Matematyki UW. Od roku 1972 pracuje w Instytucie Maszyn Matematycznych na stanowisku starszego asystenta. Zajmuje się strukturami danych oraz organizacją dużych zbiorów danych.

Anna WIECZOREK ukończyła studia na Wydziale Matematyki UW w roku 1968. W latach 1965-1974 pracowała w Instytucie Maszyn Matematycznych, obecnie pracuje w Instytucie Organizacji i Kierowania na stanowisku starszego asystenta. Zajmuje się zagadnieniami projektowania systemów przetwarzania danych, a zwłaszcza organizacją dużych zbiorów danych.

Jan WIERZBOWSKI ukończył studia na Wydziale Matematyczno-Fizycznym UW w 1955 roku. W Instytucie Maszyn Matematycznych pracuje od 1957 roku. Pracując, a następnie kierując Zakładem Przetwarzania Danych zajmował się projektowaniem i realizacją systemów przetwarzania danych. Obecnie zajmuje się bankami danych. Jest autorem wielu publikacji.

Julian WINIEWSKI, ukończył studia na Wydziale Matematyki UW w 1971 roku. W Instytucie Maszyn Matematycznych pracuje od 1971 roku, obecnie na stanowisku starszego asystenta. Zajmuje się teoretycznymi aspektami organizacji dużych zbiorów danych oraz semantyką języków programowania.

ON PERFORMANCE CHARACTERISTICS OF INFORMATION PAGE COMPOSER IN DIGITAL HOLOGRAPHIC MEMORY

Zdzisław WRZESZCZ ukończył studia na Wydziale Elektroniki Politechniki Warszawskiej. Od roku 1959 pracuje w Instytucie Maszyn Matematycznych, zajmując się początkowo projektowaniem

i badaniami układów techniki analogowej. W latach 1962-65 pod jego kierunkiem opracowano tranzystorowe układy logiczne S-400, zespoły zasilania oraz operacyjną pamięć ferrytową PAO-5 do maszyny cyfrowej ZAM 41; w okresie późniejszym powstała pamięć ferrytowa PAO-6 nagrodzona przez Przewodniczącego KNiT. Obecnie jest kierownikiem Zakładu Optocyfroniki i pracuje nad podstawami budowy układów i systemów liczących zawierających elementy optoelektroniczne. Jest autorem wielu opracowań i publikacji, głównie na temat pamięci wewnętrznych maszyn cyfrowych.

LL - REGULAR GRAMMARS

Stanisław JARZĄBEK ukończył w 1972 roku studia na Wydziale Matematyki Uniwersytetu Warszawskiego. W tym samym roku podjął pracę w Instytucie Maszyn Matematycznych w Zakładzie Metod Translacji. Obecnie pracuje w Zakładzie Teorii Translatorów, gdzie zajmuje się problemami optymalizacji programów.

Tomasz KRAWCZYK ukończył studia na Wydziale Matematyki Uniwersytetu Warszawskiego w 1968 roku. W tym samym roku podjął pracę w Instytucie Maszyn Matematycznych w Zakładzie Metod Programowania. Obecnie jest kierownikiem pracowni w Zakładzie Teorii Translatorów. Zajmuje się metodami analizy składniowej i z tej tematyki posiada kilka publikacji.

BADANIA MATERIAŁÓW CERAMICZNYCH MOGĄCYCH ZNALEŹĆ ZASTOSOWANIE W GŁOWICACH MAGNETYCZNYCH

Joanna SYNAK. W 1968 roku ukończyła Wydział Chemii na Uniwersytecie Warszawskim. W latach 1968-1972 pracowała w Zakładach Ceramiki Radiowej. Od 1972 roku pracuje w Instytucie Maszyn Matematycznych, początkowo w Zakładzie Pamięci Wirujących, obecnie w Zakładzie Głowic Magnetycznych, gdzie zajmuje się materiałami do głowic.

Wacław WELIK. Ukończył w 1956 roku Wydział Mechaniczno-Technologiczny Politechniki Warszawskiej. W latach 1956-63 pracował w Doświadczalnych Zakładach Lampowych "Lamina". W 1963 roku ukończył Wydział Łączności Politechniki Warszawskiej. W latach 1964-69 pracował w Biurze Rozwojowo-Konstrukcyjnym Z.D.B.A.N. "Unipan". Od 1970 roku pracuje w Instytucie Maszyn Matematycznych. Jest kierownikiem Pracowni Materiałów i Mikromontażu w Zakładzie Głowic Magnetycznych. Posiada kilka publikacji.

SZACOWANIE I ANALIZA PROGRAMÓW NIEZAWODNOŚCIOWYCH ZA POMOCĄ PROGRAMU "LAMBDA"

Jerzy DĄBROWSKI ukończył w 1972 roku studia matematyczne na Uniwersytecie Warszawskim i w tym samym roku rozpoczął pracę w Instytucie Maszyn Matematycznych, zajmując się zastosowaniem metod numerycznych w problematyce niezawodnościowej. Obecnie zajmuje się tematyką banków danych.

Cena 60,-