

BIBLIOTEKA NARODOWA
DZIAŁ PRZETWARZANIA DANYCH

PAKIET MAK

EDYCJA I DRUKOWANIE

Wydanie piąte rozszerzone i poprawione

Stan na dzień 1 kwietnia 2004 r.

Opracował Jerzy Swianiewicz

WARSZAWA 2004
BIBLIOTEKA NARODOWA

CIP – Biblioteka Narodowa

Swianiewicz Jerzy

Pakiet MAK – edycja i drukowanie / oprac. Jerzy Swianiewicz ;

Biblioteka Narodowa. Dział Przetwarzania Danych. – Wyd. 5

rozszerz. i popr., stan na dzień 1 kwietnia 2004. – Warszawa : BN, 2004

ISBN 83-7009-594-1

BIBLIOTEKA NARODOWA • WARSZAWA 2004

Druk: Wydawnictwo Biblioteki Narodowej

Warszawa, al. Niepodległości 213

Spis treści

1	Język opisu drukowania dokumentów z bazy MAK	7
1.1	Struktura opisu drukowania	7
1.2	Struktura opisu drukowania dokumentu	8
1.3	Instrukcje opisu drukowania	9
1.3.1	Instrukcje pobierania danych z bazy	9
1.3.2	Instrukcje sterujące	10
1.4	Kodowanie znaków specjalnych	16
1.5	Warunkowa translacja	17
1.6	Budowa DEFINICJI	17
1.7	Wywołanie programu MAKD	17
1.7.1	Szczegółowy opis dekodatorów	19
1.8	Przykład formatu 1	20
1.9	Generowanie formatu wzorcowego	21
2	Edytor EDYTO	22
2.1	Instrukcje EDYTO	22
2.2	Wywołanie programu EDYTO	22
2.3	Przykład formatu 2	23
3	Edytor TABLI	24
3.1	Struktura danych dla TABLI	24
3.2	Instrukcje TABLI	24
3.3	Wywołanie programu TABLI	25
3.4	Przykład formatu 3	26
4	Edytor SEGKK	28
4.1	Struktura danych dla SEGKK	28
4.2	Instrukcje, znaczniki SEGKK	28
4.3	Wywołanie programu SEGKK	29
5	Edytor YEDYT	31
5.1	Instrukcje YEDYT	31
5.1.1	Instrukcje konstruowania akapitów	31
5.1.2	Instrukcje ustawiania współrzędnych	33
5.1.3	Instrukcje konstrukcji strony	33
5.1.4	Instrukcja definiowania paginy	33
5.1.5	Instrukcje działań na rejestrach	33
5.1.6	Instrukcja definiowania kolumn na stronie	34
5.2	Wywołanie programu YEDYT	34
5.3	Przykład formatu 4	35
6	Program SKSORT	37
6.1	Wywołanie programu SKSORT	40
6.1.1	Zbiór zmian w tablicy kodującej	40
6.2	Przykład formatu 5	41

7	Programy pomocnicze	43
7.1	Program ZAMIANY	43
7.1.1	Wywołanie programu ZAMIANY	45
7.2	Program MAZAM	45
7.2.1	Wywołanie programu MAZAM	45
7.3	Program TSKR	46
7.3.1	Wywołanie programu TSKR	46
7.4	Program REDINDD	46
7.4.1	Wywołanie programu REDINDD	46
7.5	Program STYTABB	47
7.5.1	Wywołanie programu STYTABB	47
7.6	Program STATY	47
7.6.1	Wywołanie programu STATY	48
7.7	Program ADAPT	48
7.7.1	Wywołanie programu ADAPT	49
7.7.2	Przykład dekodera	50
A	Wykorzystanie znaków ASCII > 127	52
B	Procedury drukowania (ba' y)	53
C	Formaty dla indeksów	55
D	Tablice kodujące programu SKSORT	59

Opisane tu programy realizują edycję danych z baz typu MAK. Wszystkie one działają na zasadzie strumieniowej (batch). Oznacza to, że każdy z nich, aby wykonać swoje zadanie, pobiera ze wskazanego zbioru strumień danych i informacji sterujących, aby przetworzyć je na inny zbiór danych. Dla uzyskania różnego typu edycji używa się różnych programów łącząc je w *łańcuch* poprzez skonstruowanie zbioru o rozszerzeniu **.BAT**. Pierwszym programem w takim łańcuchu jest zawsze program **MAKD**, który kontaktuje się z bazą typu MAK "wyjmując" z niej dane.

Dane sterujące programem **MAKD** zebrane są w oddzielnym zbiorze, który nazywamy *formatem*. Format steruje całym procesem drukowania. Zawiera on zarówno instrukcje sterujące programem **MAKD** jak i – przekazywane wraz z danymi – instrukcje dla innych programów edycyjnych realizujących dalsze etapy przetwarzania. Głównym zadaniem niniejszego opracowania jest opisać zasady konstruowania formatów.

Niniejszy opis może się okazać trudnym do zrozumienia dla nie przygotowanego czytelnika. Pewną pomocą może wtedy być przyjrzenie się zamieszczonym przykładom formatów. Jest ich pięć. Są możliwie proste i w praktyce mało przydatne, ilustrują jednak podstawowe mechanizmy stosowane przy konstruowaniu formatów bardziej złożonych.

W pierwszym przykładzie pokazany jest format, który steruje programem **MAKD** tak aby wynik jego działania został umieszczony w zbiorze tekstowym na dysku. W następnych demonstruje się kolejno: współdziałanie programu **MAKD** z programem **EDYTO** (przykład 2), z programem **TABLI** (przykład 3), z programem **YEDYT** (przykład 4) i z programami **SKSORT** i **YEDYT** (przykład 5). Przykłady formatów wraz z odpowiadającymi im zbiorami *bat* są udostępniane użytkownikom w zbiorach **OPXXPP_n.SFO** i **PRZYK_n.BAT**, gdzie *n* – numer przykładu.

1 Język opisu drukowania dokumentów z bazy MAK

Wybrane dokumenty z bazy MAK można wydrukować lub przygotować do wydrukowania za pomocą programu **MAKD**. Przez „przygotowanie do wydrukowania” rozumie się tutaj utworzenie zbioru tekstowego zawierającego sekwencje sterujące określonego edytora tekstu. Przetworzenie takiego tekstu przez odpowiedni edytor da w wyniku tekst drukowany.

Wyróżnionym edytorem dla **MAKD** jest edytor $\text{T}_{\text{E}}\text{X}$. Wyróżnienie to polega na tym, że program **MAKD** wśród swoich dekodery posiada taki (dekoder 2), który zamienia pewne znaki specjalne z zapisu stosowanego w bazach MAK na zapis edytora $\text{T}_{\text{E}}\text{X}$.

Podstawowym parametrem programu **MAKD** jest **format**. Z punktu widzenia programu **MAKD**, **format** jest — odpowiednio zbudowanym — zbiorem danych sterującym formatowaniem danych wyjściowych programu **MAKD**. Mówimy, że **format** zawiera *opis drukowania* dokumentów bazy MAK.

W dalszym ciągu opisane są zasady konstruowania *opisów drukowania* czyli **formatów**. Podany opis dotyczy postaci *źródłowej* formatu. Przed zastosowaniem, format powinien być przetworzony na postać *wynikową* za pomocą programu **TFORM**.

Nazwy zbiorów zawierających formaty w postaci źródłowej winny mieć postać następującą:

OPxxlln.SFO

gdzie **lln** jest oznaczeniem formatu (2 litery + liczba do dwóch cyfr) podawanym jako parametr **MAKD**, a **xx** — dowolne dwa znaki.

Postać wynikową otrzymujemy przez wywołanie programu **TFORM** w następujący sposób:

TFORM OPxxlln dd k

gdzie:

dd jest oznaczeniem typu drukarki (lub edytora tekstu), dla której został przygotowany format.

Dozwolone wartości parametru **dd**: **X1**, **EP**, **CI**, **TX** i **EQ**. Przyjęto stosować **EP** dla formatów przygotowujących dane dla drukarki 9-igłowej, pracującej w standardzie EPSON, — **EQ** dla formatów przygotowujących dane dla drukarki 24-igłowej, pracującej w standardzie EPSON, a **TX** — dla formatów produkujących dane dla edytora $\text{T}_{\text{E}}\text{X}$. Jest to jednak czysta umowa. Programy **MAKD** i **TFORM** ani nie kontrolują, ani nie przystosowują formatów do odpowiednich drukarek, jakkolwiek tzw. definicje warunkowe (patrz. 1.6) dostarczają pewnego automatyzmu.

k może przyjmować wartość **0** (dla formatów w kodzie MAZOVIA) lub **1** (dla formatów w kodzie LATIN 2)

Wynikiem działania programu **TFORM** jest zbiór o nazwie:

OPddlln.TTT

zawierający postać *wynikową* formatu.

1.1 Struktura opisu drukowania

Opis drukowania zawiera teksty i operacje wykonywane w trakcie drukowania dokumentów z bazy MAK.

Ogólna struktura opisu drukowania:

```
DEFINICJE
$$$
PROLOG<K>komentarz
<PD>Opis Drukowania Dokumentu<K>komentarz
<EP>EPILOG
```

DEFINICJE — jest to ciąg wierszy definiujących symbole oraz ich wartości. Symbole te program TFORM zastąpi przez ich wartości przed przystąpieniem do przetwarzania opisu na postać wynikową. Dokładny opis tej części opisu podany jest w p. 1.6.

PROLOG jest zapisem tekstu inicjującego wydruk (tytuł, instrukcje definiujące parametry strony itp.).

Opis Drukowania Dokumentu zawiera instrukcje określające sposób drukowania poszczególnych pól i podpól dokumentu.

EPILOG jest zapisem tekstu kończącego wydruk.

1.2 Struktura opisu drukowania dokumentu

Ogólna struktura opisu drukowania dokumentu:

```
<PD>tekst drukowany na początku dokumentu<K>komentarz
1<Np>nazwa pola<K>komentarz
2<Np>nazwa pola<Pob>klasa porządku pól<K>komentarz
3<Np>nazwa pola<Po>nazwa podpola porządkującego<Po>klasa porządku pól<K>komentarz
<PjI>tekst inicjujący pola jednokrotnego<K>komentarz
<PjK>tekst kończący pola jednokrotnego<K>komentarz
<PwpI>tekst inicjujący pierwsze wystąpienie pola wielokrotnego<K>komentarz
<PwpK>tekst kończący pierwsze wystąpienie pola wielokrotnego<K>komentarz
<PwsI>tekst inicjujący środkowe wystąpienie pola wielokrotnego<K>komentarz
<PwsK>tekst kończący środkowe wystąpienie pola wielokrotnego<K>komentarz
<PwoI>tekst inicjujący ostatnie wystąpienie pola wielokrotnego<K>komentarz
<PwoK>tekst kończący ostatnie wystąpienie pola wielokrotnego<K>komentarz
<Pbr>tekst wypisywany przy braku pola w dokumencie <K>komentarz
4<Npp>nazwa podpola<K>komentarz
5<Npp>nazwa podpola<PPo>klasa porządku podpól<K>komentarz
<PPj>tekst dla podpola jednokrotnego<K>komentarz
<PPwp>tekst dla pierwszego wystąpienia podpola wielokrotnego<K>komentarz
<PPws>tekst dla środkowego wystąpienia podpola wielokrotnego<K>komentarz
<PPwo>tekst dla ostatniego wystąpienia podpola wielokrotnego<K>komentarz
<PPbr>tekst wypisywany przy braku podpola w polu <K>komentarz
...
*** opisy drukowania pozostałych podpól pola ***
...
...
*** opisy drukowania pozostałych pól dokumentu ***
...
<KD>tekst drukowany na końcu dokumentu<K>komentarz
```

Objaśnienia:

Grupy wierszy poprzedzone grubą kreską pionową (¶) stanowią opcje, z których należy wybrać jedną. Opcje te zostały ponumerowane i objaśnienia niektórych z nich podane są poniżej:

opcja 2 wpływa na kolejność przetwarzania pól i podpól z danej klasy. Wystąpienie klasy porządku pól w kilku polach powoduje, że pola o tej samej klasie porządku pól oraz ich podpola będą przetwarzane w kolejności ich zapisu w bazie.

opcja 3 wpływa na kolejność przetwarzania pól i podpól z danej klasy. Klasa porządku pól musi być liczbą całkowitą. Nazwa podpola porządkującego wskazuje na podpole, którego zawartością też musi być liczba całkowita. Pola o tej samej klasie porządku pól wraz z ich

podpolami, będą przetwarzane w kolejności określonej przez zawartość podpola porządkującego. A więc, jeżeli pola *A* i *B* mają identyczną klasę porządku pól i oba są wielokrotne, to najpierw będą przetwarzane wystąpienia pól *A* i *B* nie zawierające podpola porządkującego, następnie będą przetworzone wystąpienia tych pól, w których podpole porządkujące zawiera liczbę 0, później — liczbę 1, itd...

opcja 5 Wystąpienie *klasy porządku podpól* w kilku podpolach tego samego pola powoduje, że wystąpienia podpól o tej samej *klasie porządku podpól* będą przetwarzane w kolejności ich zapisu w bazie.

Osobliwości:

Pole o zastrzeżonej nazwie **@brak@**, użyte w opisie drukowania, zachowuje się tak, jak gdyby było określone w każdej bazie, ale w żadnym dokumencie nie występuje (sygnalizuje 'brak')

Pole o zastrzeżonej nazwie **@kazde@**, użyte w opisie drukowania, zachowuje się tak, jak gdyby było równe każdej nazwie pola występującej w dokumencie. Dokument jest wtedy traktowany tak, jak gdyby zawierał jedno wielokrotne pole. Prawdziwą nazwę pola, w każdym wystąpieniu, można wtedy uzyskać za pomocą instrukcji **<Onazp>**.

Analogicznie: podpole o zastrzeżonej nazwie **@kazde@**, użyte w opisie drukowania, zachowuje się tak, jak gdyby było równe każdej nazwie podpola występującej w polu. Pole jest wtedy traktowane tak, jak gdyby zawierało jedno wielokrotne podpole. Prawdziwą nazwę podpola, w każdym wystąpieniu, można wtedy uzyskać za pomocą instrukcji **<Onazpp>**.

1.3 Instrukcje opisu drukowania

Teksty występujące w opisie drukowania mogą zawierać instrukcje, które są realizowane w trakcie kopiowania odpowiedniego tekstu. Dzielą się one na dwie kategorie:

1. Instrukcje pobierania danych z bazy,
2. Instrukcje sterujące.

1.3.1 Instrukcje pobierania danych z bazy

Dane z bazy (tzn. wartości podpól) mogą być kopiowane bezpośrednio na wyjście (tzn. do drukarki lub do zbioru wyjściowego na dysku) lub też mogą być kopiowane do bufora **DBbuf**, gdzie mogą być poddane dalszemu przetwarzaniu. W obu przypadkach dane pobierane z bazy są dekodowane za pomocą uprzednio wybranego dekodera (patrz parametr *dek* w 1.7 oraz p. 1.7.1). Znak '«' (wartość dzies. 174) powoduje wyłączenie dekodowania, a znak '»' (wartość dzies. 175) - przywrócenie dekodowania wg ostatnio aktywnego dekodera.

Funkcje pobierania danych z bazy realizują instrukcje:

<Dpp> – przepis� zawartość podpola na wyjście

<Bpp> – przepis� zawartość podpola do bufora **DBbuf**

Powyższe instrukcje mogą występować tylko w tekstach dla podpól i dotyczą tychże podpól.

Pojemność bufora **DBbuf** - 1600 znaków (bajtów).

1.3.2 Instrukcje sterujące

Instrukcje sterujące mogą dotyczyć rejestrów, dokonywania operacji na buforze **DBbuf** oraz przełączania wyjścia na jeden z 10 tymczasowych buforów **TBuf₀, ..., TBuf₉**. Bufory **TBuf_i** mają różne pojemności: 0 – 3000, 1 – 12000, 2 – 2700, 3 – 2700, 4 – 600, 5 – 150, 6 – 150, 7 – 150, 8 – 150 i 9 – 200 znaków (bajtów). Instrukcja **<OzmT>** pozwala zmienić pojemności buforów **TBuf_i**.

Dwubajtowe rejestry r_0, r_1, \dots, r_{33} mogą zawierać liczby z przedziału $(-32768, +32767)$. Zawartość niektórych rejestrów jest automatycznie inicjowana w sposób następujący:

- r_0 — numer systemowy aktualnego dokumentu
- r_1 — numer powtórzenia bieżącego pola
- r_2 — numer powtórzenia bieżącego podpola
- r_3 — numer kolejny drukowanego dokumentu
- r_{16} — indeks opisu pola/podpola w OPIDOK
- r_{18} — numer systemowy dokumentu nadrzędnego
- r_{19} — indeks opisu bieżącego pola w OPIDOK
- r_{20} — długość tekstu w buforze **DBbuf** (liczba znaków)
- r_{30} — 15 wskaźników (bity 0–14) dotyczących sposobu wyświetlania dokumentu przez MAK
- r_{31} — liczba określająca kolejność wyświetlania pól/podpól w bieżącej masce (wartości 0–512)
- r_{32} — 8 wskaźników (bity 0–7) dotyczących sposobu wyświetlania bieżącego pola/podpola, w szczególności:
 - bit 1 – (0): wyświetlać dane pole/podpole,
 - (1): nie wyświetlać danego pola/podpola
- r_{33} — liczba – wskaźnik wyświetlania (w MAKU) nazw pól/podpól:
 - 0 – nie wyświetlać,
 - 1 – wyświetlać nazwy pełne,
 - 2 – wyświetlać nazwy skrócone.

Czterobajtowe rejestry $rd_0, rd_1, \dots, rd_{39}$ mogą zawierać liczby z przedziału $(-2^{31}, +2^{31} - 1)$. Dwa z nich dostają automatycznie wartość początkową:

- rd_0 – numer systemowy aktualnego dokumentu
- rd_1 – numer systemowy dokumentu nadrzędnego

32-bajtowe rejestry tekstowe rt_0, \dots, rt_{11} mogą zawierać teksty nie przekraczające 31 bajtów długości. Do rejestrów rt_4 i rt_5 (bajty 0–11) wstawiane są automatycznie odpowiednio: nazwa bieżącego pola i nazwa bieżącego podpola. Ponadto, do bajtów 20–21 rejestru rt_5 , wpisywana jest nazwa skrócona bieżącego pola/podpola. Do rejestru rt_6 wpisywane są bieżące nazwa pola i nazwa podpola, oddzielone dwukropkiem.

STOS

STOS jest pamięcią, do której można wkładać teksty.

Instrukcja **<Otbst><n>** przenosi zawartość bufora **TBuf_n** na **STOS**.

Instrukcja **<Orsto><p>** *schemat* wypisuje wskazane w *schemacie* bufory **TBuf** tyle razy, ile jest tekstów na **STOS**ie, wstawiając za każdym razem we wskazane miejsce kolejny tekst ze **STOS**u.

STOS można wyzerować (instrukcja **<Ozsto><0>**).

Liczniki DMIch i DMOgr.

Podczas wysyłania tekstu na drukarkę licznik **DMIch** jest zwiększany o 1 po każdym znaku, z wyjątkiem *nowej linii* **<lf>** (wartość 10). Wysłanie znaku **<lf>** powoduje automatyczne wyzerowanie licznika **DMIch**. **DMIch** wskazuje zatem *bieżącą liczbę znaków w wierszu drukarki*. Instrukcja **<Odmc><p>**, stosowana na ogół w połączeniu z funkcją **<tr_{2p}>**, pozwala korygować liczbę znaków w wierszu w przypadku, gdy wysyłane są znaki sterujące drukarki. Wartość początkowa: $DMIch = 0$.

Licznik **DMOgr** pełni funkcję ograniczania długości wiersza drukarskiego. Gdy *DMlch* > *DMOgr*, najbliższa spacja jest automatycznie zamieniana na *crlf*, a **DMlch** zostaje wyzerowany. Wartość początkowa: *DMOgr* = 32767 (praktycznie oznacza to brak ograniczenia długości wiersza).

Instrukcje sterujące:

- <Ozer>< r > — zeruj rejestr *r*;
- <Ozerd>< rd > — zeruj rejestr długi *rd*;
- <Odr>< r > — wypisz zawartość rejestru *r* w zapisie dziesiętnym;
- <Oddr>< rd > — wypisz zawartość rejestru długiego *rd* w zapisie dziesiętnym;
- <Oddrx>< rd > — wypisz zawartość rejestru długiego *rd* w zapisie dziesiętnym wstawiając przecinek (",") przed dwoma ostatnimi cyframi;
- <Onast>< r > — zwiększ zawartość rejestru *r* o 1;
- <Osre>< r >< p > — wstaw *p* do rejestru *r*;
- <Osbit>< r >< b > — ustaw bit *b* w rejestrze *r*. Stosowane jest dodawanie logiczne. Bity rejestru numerowane są od końca — ostatni bit ma numer 0. Bit *b*, liczbowo, odpowiada wartości 2^b ;
- <Opr>< r₁ >< r₂ > — przenieś zawartość *r₂* do *r₁*;
- <Omax>< r > — wstaw do rejestru *r* liczbę 16380;
- <Owzn>< z > — jeżeli ostatni wypisany znak jest różny od *z* – wypisz *z*;
- <Ofun>< n >< p > — wykonaj funkcję *tr_n* z parametrem *p* (opis funkcji *tr_i* – niżej);
- <Ocbuf>< 0 > — wypisz całą zawartość bufora **DBbuf**; zawartość **DBbuf** nie ulega zmianie;
- <Ozdb>< rt > — przepisuj zawartość rejestru tekstowego *rt* do bufora **DBbuf**;
- <Obrd>< rd > — wybierz cyfry (od lewej do prawej) z bufora **DBbuf**, utwórz z nich liczbę i jej wartość wpisz do rejestru długiego *rd*. Jeżeli otrzymana liczba jest większa od 999 999 999 to do rejestru *rd* zostanie wpisana liczba 2 000 000 000;
- <Ozbuf>< rt₁ >< rt₂ > — w buforze **DBbuf** wszystkie wystąpienia tekstu zawartego w rejestrze tekstowym *rt₁* zamień na tekst zawarty w rejestrze tekstowym *rt₂*;
- <Odmc>< p > — jeżeli *p* < 255, zmniejsz licznik *DMlch* o *p*, jeżeli *p* = 255, nadaj *DMlch* wartość -32767;
- <Otbz>< n > — zeruj **TBuf_n**;
- <Otst>< n > — przełącz wyjście na **TBuf_n**;
- <Otko>< 0 > — przerwij wypisywanie do **TBuf_n** (przełącz wyjście na standardowe);
- <Odt>< n > — wypisz zawartość **TBuf_n**;
- <Odtb>< n > — przepisuj zawartość **TBuf_n** do **DBbuf**. Zawartość **TBuf_n** nie ulega zmianie. Jeżeli przepisany tekst nie mieści się w **DBbuf** to umieszczona w nim będzie część początkowa tekstu z **TBuf_n**;
- <Odtp>< n >< p > — wypisz *p* początkowych znaków z **TBuf_n** uzupełniając spacjami (gdy trzeba);
- <Otad>< n >< p >< z > — uzupełnij bufor **TBuf_n** do *p* znaków znakami *z*. Gdy bufor **TBuf_n** zawiera więcej niż *p* znaków, pozostaw w nim *p* początkowych znaków, a pozostałe usuń;
- <Otp>< n >< p >< z > — uzupełnij bufor **TBuf_n** do *p* znaków znakami *z*, przesuując tekst zawarty w buforze w prawo (adjustacja *do prawej*). Gdy bufor **TBuf_n** zawiera więcej niż *p* znaków, pozostaw w nim *p* początkowych znaków, a pozostałe usuń;
- <Otkon>< n >< z > — w buforze **TBuf_n** usuń końcowe znaki *z*;
- <Odek>< n > — ustaw dekodery *n*;
- <Omas>< n > — ustaw maskę *n*; po tej instrukcji instrukcje <Onazp> i <Onazpp> będą produkowały nazwy wg maski *n* i zawartość rejestrów *rt₄* i *rt₅* będzie ustalana, wg maski *n* (format zawsze przygotowuje się wg maski 0);
- <Oskip>< p > — wstaw *p* do wskaźnika pomijania *Dskip*;
Dskip = 0 – nie pomijać
Dskip = 1 – pomijać, gdy *r₁₆* > *r₁₇* (z wyj. pół/podpół „brak”)

Dskip = 2 – pomijać, gdy $r_{16} < r_{17}$ (z wyj. pól/podpól „brak”)
Dskip = 3 – pomijać wszystkie z wyj. pól/podpól „brak”
Dskip = 4 – pomijać wszystkie (zaniechać drukowania dokumentu).
Na początku dokumentu, Dskip = 0;

- <Oskp>< r >< p > — jeżeli zawartość rejestru $r = p$ – zaniechaj wypisywania bieżącego pola, podporządkowanych mu podpól oraz tekstu kończącego pole;
- <Oskpp>< r >< p > — jeżeli zawartość rejestru $r = p$ – zaniechaj wypisywania bieżącego podpola;
- <Orskpp>< r₁ >< r₂ > — jeżeli zawartość rej. r_1 jest różna od zawartości rej. r_2 – zaniechaj wypisywania bieżącego podpola;
- <Oogr>< p > — jeżeli $p < 255$, wstaw p do DMOgr,
jeżeli $p = 255$, nadaj DMOgr wartość 32767;
- <Odata>< p > — wypisz datę w postaci p ;
 $p = 0$ – data bieżąca, postać: *dd/mm/rr*
 $p = 1$ – data bieżąca, postać: *dd* miesiąca 19*rr*
 $p = 2$ – data wprowadzenia dokumentu do bazy, postać: *rr.mm.dd*
 $p = 3$ – data ostatniej modyfikacji dokumentu w bazie, postać: *rr.mm.dd*;
- <Ozzer>< rt > — wpisz tekst pusty do rejestru tekstowego rt ;
- <Ozlad>< rt >< p >tekst — wstaw do rejestru rt , złożony z p znaków, tekst występujący bezpośrednio po instrukcji;
- <Ozladb>< rt > — przenieś do rejestru rt zawartość bufora **DBbuf**, lub pierwszych 31 bajtów **DBbuf**;
- <Ozladp>< rt >< p > — wstaw do rejestru rt tekst odpowiedzi operatora MAKD na pytanie p
 $p = 0$ – odpowiedź na pytanie: „Czy pytać o liczbę kopii”;
 $p = 1$ – nazwa bazy (wraz z ew. ścieżką dostępu);
 $p = 2$ – nazwa indeksu lub zbioru zaznaczonych dokumentów;
- <Ondok>< n > — ustal numer następnego dokumentu do przetwarzania
 $n = 0$ – następnym dokumentem będzie dokument nadrzędny;
- <Ostop>< 0 > — przerwij przetwarzanie dokumentów;
- <Owysw>< n > — wyświetl na ekranie (w obszarze sygnalizacji błędów) zawartość bufora **TBuf_n** do pierwszego znaku < 0 > lub pierwsze 79 znaków. Tekst wyświetlany na ekranie zostaje również dopisany do zbioru MAKD.LOG;
- <Opause>< 0 > — zatrzymaj pracę programu **MAKD** do chwili naciśnięcia klawisza Esc (np. w celu umożliwienia przeczytania komunikatu wysłanego przy pomocy instrukcji <Owysw>< n >;
- <Oind>< 0 > — przenieś bieżącą wartość z indeksu do **DBbuf** oraz przejdź do następnej wartości w indeksie;
- <Ocekr>< p >tekst — pobierz tekst z ekranu monitora. Na ekranie pojawi się polecenie ‘podaj tekst’ uzupełnione tekstem **tekst** o długości p znaków. Użytkownik winien wprowadzić wiersz tekstu (nie więcej niż 75 znaków) i nacisnąć ENTER. We wprowadzonym tekście znaki ‘spacja’ należy zastępować znakami ‘podkreślenie’ (‘_’);
- <Ocrt>< rt >< d >tekst — pobierz tekst z ekranu monitora do rejestru tekstowego rt . Na ekranie pojawi tekst **tekst** (d – jest jego długością), który ma poinstruować użytkownika, o co chodzi. Użytkownik winien wprowadzić z klawiatury tekst (nie dłuższy niż 30 znaków) i nacisnąć ENTER.
- <Oczb>< d >nazwa zbioru — pobierz tekst ze zbioru **nazwa zbioru** i wyślij go na wyjście.
 d jest liczbą znaków tekstu **nazwa zbioru**;
- <Onazp>< 0 > — wypisz nazwę bieżącego pola;
- <Onazpp>< 0 > — wypisz nazwę bieżącego podpola;
- <Onazsk>< 0 > — wypisz nazwę skróconą pola/podpola (z bajtów 20 i 21 rejestru rt_5);

<Owsk>< n >< k > — nadaj wskaźnikowi W_n wartość k .

Wskaźniki:

W_0 ($W_0 = 0$ – przy pobieraniu danych z bazy nie pomijaj znaku '|'; $W_0 = 1$ (wartość domyślna) – pomijaj znak '|'),

W_1 – określa wartość liczbową znaku *sep1* wykorzystowanego przez funkcję < $tr_{14} n$ > (wartość domyślna: 93 (znak '|')).

<Opetl>< r >sekwencja< fi > — powtarzaj sekwencję zawartą między <Opetla>< r > i < fi > $r + 1$ razy (r – zawartość rejestru r). Sekwencja może zawierać instrukcje, ale instrukcje warunkowe muszą zamykać się w ramach sekwencji. W trakcie realizacji pętli zawartość rejestru r zmienia się (maleje do 0 przez odejmowanie 1).

Zmiana pojemności buforów **TBuf_i**:

<OzmT>< 0 > $l_0sl_1sl_2sl_3sl_4sl_5sl_6sl_7sl_8sl_9$ K — Instrukcja ta określa nowe długości buforów **TBuf_i**; bufor **TBuf_i** będzie miał długość l_i . Należy określać długości wszystkich buforów naraz przy czym suma $l_0 + l_1 + \dots + l_9$ musi wynosić 21800 (w przeciwnym przypadku — błąd). W wyniku tej instrukcji wszystkie **TBuf** zostaną wyzerowane.

Przykład:

<OzmT>< 0 >2180s2180s2180s2180s2180s2180s2180s2180s2180s2180K —

w wyniku powyższej instrukcji wszystkie **TBuf** będą miały pojemność 2180 bajtów.

Instrukcje dotyczące **STOSu**:

<Otblst>< n > — Dopisz do **STOSu** tekst stanowiący zawartość bufora **TBuf_n**;

<Ozsto>< 0 > — Zeruj **STOS**. Po tej instrukcji **STOS** nie zawiera żadnego tekstu;

<Orsto>< p >schemat — Instrukcja określa *schemat*, który jest ciągiem znaków będących cyframi 0, 1, 2, 3 lub literą 'x'. Cyfry w *schemacie* są numerami buforów **TBuf**, a litera 'x' jest oznaczeniem „kolejnego” tekstu na **STOSie**. Wykonanie instrukcji polega na interpretacji *schematu* tyle razy, ile trzeba aby wykorzystać wszystkie teksty ze **STOSu** (za każdą interpretacją litery 'x' wykorzystywany jest kolejny tekst ze **STOSu**). Interpretacja *schematu* polega na zinterpretowaniu kolejno wszystkich jego znaków. Interpretacja cyfry c powoduje wypisanie bufora **TBuf_c**. Interpretacja litery 'x' powoduje wypisanie kolejnego tekstu ze **STOSu**. Teksty ze **STOSu** wypisywane są w kolejności takiej, jak były zapisywane (FIFO). W *schemacie* może być kilka liter 'x', ale *schemat* bez litery 'x' jest nieprawidłowy

Działania arytmetyczne na rejestrach:

<Oaryt>< 0 ><ciąg operacji>K — Wykonaj *ciąg operacji* na rejestrach.

W *ciągu operacji* mogą występować następujące:

(poniżej, *liczba* jest ciągiem cyfr zamienianym na wartość liczbową.)

D< n >*liczba* — dodaj do zawartości rejestru r_n liczbę *liczba*

O< n >*liczba* — odejmij od zawartości rejestru r_n liczbę *liczba*

M< n >*liczba* — pomnóż zawartość rejestru r_n przez liczbę *liczba*

Q< n >*liczba* — podziel zawartość rejestru r_n przez liczbę *liczba*, iloraz zostaw w r_n , a resztę w r_{n+1}

R< n1 >< n2 > — przenieś zawartość rejestru r_{n1} do r_{n2}

B< n1 >< n2 > — do zawartości rejestru r_{n1} dodaj zawartość r_{n2} , wynik w r_{n1}

C< n1 >< n2 > — od zawartości rejestru r_{n1} odejmij zawartość r_{n2} , wynik w r_{n1}

A< rd₁ >< rd₂ > — dodaj do zawartości rejestru długiego rd_1 liczbę zawartą w rd_2 . Wynik umieść w rd_1

X< rd₁ >< rd₂ > — pomnóż zawartość rejestru długiego rd_1 liczbę zawartą w rd_2 . Wynik umieść w rd_1

E< rd₁ >< rd₂ > — podziel zawartość rejestru długiego rd_1 liczbę zawartą w rd_2 . Wynik (liczba całkowita) umieść w rd_1 a resztę w rd_2

S< n >< rd > — przenieś zawartość rejestru r_n do rejestru długiego rd

K — zakończ wykonywanie *ciągu operacji*. Jest to również oznaczenie końca instrukcji <Oaryt>

Instrukcje sterujące warunkowe

Poniższe instrukcje warunkowe są realizowane w następujący sposób: badany jest właściwy dla danej instrukcji warunek, po czym:

- jeżeli warunek jest spełniony dalszy tekst jest kopiowany aż do separatora `<el>` lub `<sep>`. Jeżeli pierwszym napotkanym separatorem jest `<el>` to dalszy tekst aż do separatora `<sep>` jest pomijany.
- jeżeli warunek nie jest spełniony dalszy tekst jest pomijany aż do separatora `<el>` lub `<sep>`. Każdy z tych separatorów przerywa pomijanie tekstu.

Instrukcje warunkowe mogą być zagnieżdżane.

- `<Owie>< r >< p >` — zbadaj czy zawartość rejestru r spełnia warunek $r > p$, następnie kopiuje tekst według opisanej wyżej zasady;
- `<Orow>< r >< p >` — zbadaj czy zawartość rejestru r spełnia warunek $r = p$, następnie kopiuje tekst według opisanej wyżej zasady;
- `<Omni>< r >< p >` — zbadaj czy zawartość rejestru r spełnia warunek $r < p$, następnie kopiuje tekst według opisanej wyżej zasady;
- `<Obit>< r >< b >` — zbadaj czy bit b w rejestrze r jest jedynką, następnie kopiuje tekst według opisanej wyżej zasady;
- `<Orrow>< r1 >< r2 >` — zbadaj czy zawartości rejestrów r_1 i r_2 są równe, następnie kopiuje tekst według opisanej wyżej zasady;
- `<Obbuf>< 0 >` — zbadaj czy bufor **DBbuf** jest niepusty, następnie kopiuje tekst według opisanej wyżej zasady;
- `<Oozn>< z >` — zbadaj czy ostatni wypisany znak jest różny od z , następnie kopiuje tekst według opisanej wyżej zasady;
- `<Ozirow>< rt >< p >`tekst — zbadaj czy tekst zawarty w rejestrze rt jest równy, złożonemu z p znaków, tekstowi występującemu bezpośrednio po instrukcji, następnie kopiuje tekst z opisu drukowania według opisanej wyżej zasady;
- `<Ozbrow>< rt >` — zbadaj czy tekst zawarty w rejestrze rt jest równy tekstowi na początku bufora **DBbuf**, następnie kopiuje tekst z opisu drukowania według opisanej wyżej zasady;
- `<Ozbrod>< rt >` — zbadaj czy tekst zawarty w buforze **DBbuf**, jest równy (dokładnie) tekstowi zawartemu w rejestrze rt , następnie kopiuje tekst z opisu drukowania według opisanej wyżej zasady;
- `<Ozbuie>< rt >` — zbadaj czy tekst zawarty w buforze **DBbuf** jest większy od tekstu zawartego w rejestrze rt , następnie kopiuje tekst z opisu drukowania według opisanej wyżej zasady.
- `<Ozbmni>< rt >` — zbadaj czy tekst zawarty w buforze **DBbuf** jest mniejszy lub równy tekstowi zawartemu w rejestrze rt , następnie kopiuje tekst z opisu drukowania według opisanej wyżej zasady.
- `<Obstr>< r >< d >`tekst — sprawdź czy bufor **DBbuf** zawiera tekst 'tekst' o długości d , wpisz do rejestru r lokalizację znalezionej tekstu (numer znaku lub 0, gdy nie ma tekstu), następnie kopiuje tekst z opisu drukowania według opisanej wyżej zasady.
- `<Ozbstr>< r >< rt >` — sprawdź czy bufor **DBbuf** zawiera tekst zawarty w rejestrze rt , wpisz do rejestru r lokalizację znalezionej tekstu (numer znaku lub 0, gdy nie ma tekstu), następnie kopiuje tekst z opisu drukowania według opisanej wyżej zasady.
- `<Orwie>< r1 >< r2 >` — sprawdź czy zawartość rejestru r_1 jest większa od zawartości rejestru r_2 , następnie kopiuje tekst z opisu drukowania według opisanej wyżej zasady.
- `<Okwie>< r >< d >`liczba — zbadaj czy liczba zawarta w rejestrze r spełnia warunek: $r > liczba$ (d określa liczbę cyfr parametru $liczba$), następnie kopiuje tekst z opisu drukowania według opisanej wyżej zasady.
- `<Odwie>< rd >< d >`liczba — zbadaj czy liczba zawarta w rejestrze rd spełnia warunek: $rd > liczba$ (d określa liczbę cyfr parametru $liczba$), następnie kopiuje tekst z opisu drukowania według opisanej wyżej zasady.

Funkcje tr_n :

- < $tr_0 p$ > — Wypisz nie więcej niż p znaków z bufora **DBbuf** (starając się skończyć na spacji). Pozostały tekst dosuń do lewej, aktualizując liczbę znaków pozostałych w **DBbuf**. Wypisany tekst uzupełnij spacjami do długości p ;
- < $tr_1 p$ > — Wypisz z bufora **DBbuf** p kolejnych słów; jeżeli **DBbuf** zawiera mniej niż p słów — wypisz wszystkie; dosuń do lewej pozostały tekst, aktualizując liczbę znaków pozostałych w **DBbuf**. Gdy pierwsze słowo jest krótsze niż 3 znaki zostaje połączone w jedno ze słowem następnym (bez likwidowania spacji);
- < $tr_2 p$ > — Wypisz całą zawartość bufora **DBbuf** tak, aby nie przekroczyć p znaków wierszu. Licznik **DMLch** określa aktualną liczbę znaków w wierszu; wypisywany tekst jest uzupełniany *crlf* w miejsce spacji poprzedzającej słowo przekraczające ograniczenie (p);
- < $tr_3 0$ > — Wypisz zawartość bufora **DBbuf** pomijając początkowe znaki '0';
- < $tr_4 0$ > — Wypisz całą zawartość bufora **DBbuf** zamieniając '\l' na '\L' i '\g' na '\c' (dla '\uppercase' w \TeX);
- < $tr_5 0$ > — Ustaw licznik **DGklop** (liczba kopii) według zawartości **DBbuf**;
- < $tr_6 p$ > — Wypisz p początkowych znaków z bufora **DBbuf** uzupełniając spacjami (jeśli trzeba). Zawartość **DBbuf** pozostaje bez zmian;
- < $tr_7 0$ > — Zawartość bufora **DBbuf** zamień na „uppercase” z uwzględnieniem znaków polskich według Mazovii oraz ä, ç, ö, ü w kodzie IBM Character Set #2;
- < $tr_8 p$ > — Ustaw zawartość rejestru p według zawartości **DBbuf**;
- < $tr_9 p$ > — Usuń z **DBbuf** p pierwszych znaków;
- < $tr_{10} z$ > — Jeżeli **DBbuf** zawiera znak z lub dwa po sobie występujące znaki z , usuń wszystkie poprzedzające znaki wraz z tym(i) znakiem(ami). W przeciwnym przypadku — nic nie rób.
- < $tr_{11} z$ > — Wypisz z **DBbuf** wszystkie znaki do znaku z , wyłącznie. Następnie usuń z **DBbuf** wypisany tekst wraz z kończącym lub kończącymi (jeżeli jest ich kilka pod rząd) znakami z . Jeżeli znakiem z jest spacja (wartość: 32), to funkcję tę można opisać: „wypisz z **DBbuf** pierwsze słowo usuwając je z **DBbuf** wraz z kończącymi spacjami”.
- < $tr_{12} n$ > — Zakoduj „do sortowania” rejestr tekstowy n (jeżeli $n < 12$) a jeżeli $n = 64$ zakoduj analogicznie **DBbuf**. W wyniku powyższego kodowania tekst zakodowany ma długość nie większą niż 30 znaków, ewentualnie ucięty do 30 znaków. Ewentualne końcowe spacje zostają usunięte. Kodowanie „do sortowania” tak przenumerowuje znaki, aby polskie litery i znaki akcentowane znalazły się we właściwych miejscach.
- < $tr_{13} z$ > — Usuń z **DBbuf** wszystkie końcowe znaki z . Usuwane są znaki z od końca **DBbuf** aż do natrafienia na znak różny od z .
- < $tr_{14} n$ > — Funkcja zakłada, że bufor **TBuf n** zawiera tekst będący ciągiem tekstów o następującej budowie:
 $klucz < sep1 > tekst < 0 >$
Funkcja porządkuje *tekst*'y według *klucz*'y i w tej kolejności je wyprowadza. Porządkowanie według kluczy polega na tym, że z każdego klucza wybiera się cyfry, tworzy z nich liczby, które porządkuje się rosnąco. Domyślną wartością *sep1* jest 93 (znak ']'). W trakcie realizacji tej funkcji zostają zmienione *klucze* w **TBuf n** wobec czego nie może być ona wykonywana wielokrotnie.
- < $tr_{15} n$ > — Funkcja powoduje przetworzenie zawartości **DBbuf** w sposób następujący: jeżeli **DBbuf** zawiera tekst postaci:

$liczba_1.liczba_2. \dots .liczba_p$

to każda z tych liczb zostanie zamieniona na liczbę 3-cyfrową z ewentualnymi cyframi 0 na początku (jeżeli oryginalna liczba miała więcej niż 3 cyfry to zostaną 3 ostatnie), kropki zostaną usunięte. Wynikowy tekst w **DBbuf** będzie ciągiem $3n$ cyfr (n – parametr funkcji). Jeżeli $p < n$ to na końcu zostanie dopisana odpowiednia liczba cyfr 0.

Przykład:

DBbuf zawiera tekst: 3.216.5

po wykonaniu **<Ofun>< 15 >< 5 >**

DBbuf będzie zawierał tekst: 003216005000000

- < tr₁₆ n >** — Sortuj i wypisz **TBuf_n**. Funkcja zakłada, że bufor **TBuf_n** zawiera tekst będący ciągiem tekstów o następującej budowie:
tekst klucza **< sep1 >** *tekst do wyprowadzenia* **< 0 >**
"teksty kluczy" zostają uporządkowane rosnąco co, z kolei, wyznacza kolejność wyprowadzania odpowiednich "tekstów do wyprowadzenia". Domyślną wartością *sep1* jest 93 (znak '['), może on być zdefiniowany za pomocą instrukcji **<Owsk>< 1 >< sep1 >** W trakcie realizacji tej funkcji zostają zmienione *klucze* w **TBuf_n** wobec czego nie może być ona wykonywana wielokrotnie.
- < tr₁₇ p >** — Usuń z **DBbuf** wszystkie znaki do pierwszej cyfry (, gdy $p = c$) lub nie-cyfry (, gdy $p = i$).
- < tr₁₈ p >** — Wypisz z **DBbuf** wszystkie przed pierwszą cyfrą (, gdy $p = c$) lub nie-cyfrą (, gdy $p = i$).
- < tr₁₉ z >** — Wypisz z **DBbuf** wszystkie znaki do znaku z (wyłącznie) zamieniając wypisywane litery (z wyjątkiem pierwszego znaku) na małe (lowercase). Następnie usuń z **DBbuf** wypisany tekst wraz z kończącym lub kończącymi (jeżeli jest ich kilka pod rząd) znakami z . Jeżeli znakiem z jest spacja (wartość: 32), to funkcję tę można opisać: „wypisz z **DBbuf** pierwsze słowo zamieniając litery (poza pierwszą) na małe i usuwając je z **DBbuf** wraz z kończącymi spacjami”.
- < tr₂₀ 0 >** — Odwróć kolejność znaków w **DBbuf** (zawartość **DBbuf** zamień na zapis "wspak").
- < tr₂₁ r >** — Wypisz p początkowych znaków z bufora **DBbuf** uzupełniając spacjami (jeśli trzeba). p – zawartość rejestru r .
- < tr₂₂ r >** — Usuń z bufora **DBbuf** p początkowych znaków. p – zawartość rejestru r .

1.4 Kodowanie znaków specjalnych

Niedrukowalne znaki kodu ASCII mogą być wprowadzane do tekstów występujących w opisie drukowania na kilka sposobów:

1. Zapis **< liczba >** jest zapisem bajtu o wartości dziesiętnej *liczba*. Zapis ten może być bezpiecznie stosowany w tekstach przeznaczonych do skopiowania na wyjście (drukarkę, tekst dla edytora). Jeżeli *liczba* > 128, bajt będzie wewnętrznie zakodowany w postaci pary bajtów o wartościach 239, *liczba*–128. W przypadku, gdy kodowany bajt jest parametrem instrukcji sterującej powinno się stosować (dla wartości > 128) zapis **< #liczba >**.
2. Zapis **< #liczba >** jest zapisem bajtu o wartości dziesiętnej *liczba*, bez specjalnego kodowania liczb większych od 128.
3. Pewne kody ASCII mają własny zapis mnemotechniczny:
 - <ht>** – jest zapisem znaku **< HT >** (wartość 9)
 - <ff>** – jest zapisem znaku **< FF >** (wartość 12)
 - <e>** – jest zapisem znaku **< ESC >** (wartość 27)
 - <f>** – jest zapisem znaku **< FS >** (wartość 28)
 - <sp>** – jest zapisem znaku o wartości 177

<cr> – jest zapisem znaku o wartości 179

<crlf> – jest zapisem pary znaków o wartościach 179, 176

Kody <cr> i <crlf> zostały wprowadzone w związku ze specjalnym traktowaniem przez program MAKD znaków <CR> (wartość 13) i <LF> (wartość 10) występujących w opisie drukowania. Znaki te są pomijane przy wysyłaniu tekstu na drukarkę lub do zbioru danych dla edytora tekstu, natomiast kody <cr> i <crlf> są na wyjściu zamieniane na odpowiednio <CR> i <LF>.

Kod <sp> jest na wyjściu zamieniany na *spację* (wartość 32).

1.5 Warunkowa translacja

Zapisy postaci:

<EPSON> tekst <^>

<STARI> tekst <^>

<CITIZ> tekst <^>

<TEX> tekst <^>

występujące w opisie drukowania powodują, że *tekst* będzie włączony do formatu pod warunkiem, że program TFORM został wywołany odpowiednio z parametrem EP, X1, CI lub TX.

1.6 Budowa DEFINICJI

DEFINICJE jest to ciąg wierszy na początku opisu drukowania (formatu) w postaci źródłowej. Wiersz '\$\$\$' oddziela DEFINICJE od PROLOGU.

Wiersze DEFINICJI mają następującą postać:

def □ <symbol> □ <wartość> □ <komentarz>

symbol i *wartość* są dowolnymi (ale nie dłuższymi niż 25 znaków) tekstami nie zawierającymi spacji. Jeżeli zachodzi potrzeba użycia spacji w *wartości* to zaznacza się ją przy pomocy znaku '^' (wartość 94).

komentarz jest dowolnym tekstem (może zawierać spacje) kończącym się wraz z końcem wiersza.

□ - oznacza jedną lub więcej spacji.

Program TFORM, w pierwszej kolejności, zastępuje *symbole* przez ich *wartości* w całym opisie drukowania, a następnie przystępuje do przetwarzania opisu drukowania (z pominięciem DEFINICJI) na postać wynikową.

Wartość symbolu może być określona warunkowo w zależności od typu drukarki. Wtedy zapis <wartość> ma postać:

{TD₁} <wartość₁> {TD₂} <wartość₂>

i oznacza, że <wartość₁> będzie wybrana, gdy parametr „typ drukarki” przy wywołaniu TFORM będzie miał wartość TD₁, a <wartość₂>, gdy „typ drukarki” będzie TD₂. Takich członów może być więcej niż 2. TD_i może przyjmować wartości: X1, CI, EP, TX, EQ.

1.7 Wywołanie programu MAKD

Wywołanie programu MAKD ma postać następującą:

MAKD baza spwy zain druk dysk dek format oddok ldok kop p1 p2 tytuł

Opis parametrów wywołania:

baza — nazwa bazy, z której będą pobierane dokumenty. Nazwa może być poprzedzona ścieżką dostępu.

Jeżeli parametr *baza* zaczyna się od znaku '*' to, program, przed rozpoczęciem właściwej pracy, odwoła się do zbioru o nazwie otrzymanej przez odrzucenie początkowej gwiazdki. Zbiór ten, zwany *opibaz* zawiera *symboliczne nazwy* baz wraz z ich charakterystykami. Następnie zostanie wyświetlone menu, w którym pojawią

- się symboliczne nazwy baz i użytkownik będzie mógł wybrać którąkolwiek z baz za pomocą kursora;
- spwy* — sposób (kolejność) wybierania dokumentów z bazy. Może przyjmować następujące wartości:
spwy = 0 — wybieranie sekwencyjne, według kolejności *numerów systemowych* dokumentów bazy;
spwy = 1 — według indeksu; nazwę indeksu, według którego mają być wybierane dokumenty bazy podaje się w parametrze *zain*;
spwy = 2 — według *listy zaznaczonych dokumentów*; nazwę zbioru zawierającego powyższą listę podaje się w parametrze *zain*;
spwy = 3 — według *zewnątrznej listy zaznaczonych dokumentów*; nazwę zbioru zawierającego powyższą listę podaje się w parametrze *zain*. W przypadku listy zewnętrznej nie ma ograniczenia na wielkość listy oraz jest dopuszczalne aby ten sam numer powtarzał się na liście wielokrotnie;
- zain* — nazwa indeksu (gdy *spwy* = 1) lub nazwa zbioru zaznaczonych dokumentów (gdy *spwy* = 2 lub *spwy* = 3);
- druk* — typ drukarki. Wartości:
0 – drukarka igłowa STAR pracująca w reżymie IBM (będzie wybrany format o nazwie zaczynającej się od **OPX1**...)
1 – drukarka igłowa CITIZEN (będzie wybrany format o nazwie zaczynającej się od **OPCI**...)
2 – drukarka igłowa EPSON (będzie wybrany format o nazwie zaczynającej się od **OPEP**...)
3 – edytor T_EX (będzie wybrany format o nazwie zaczynającej się od **OPTX**...)
4 – drukarka EPSON 24-igłowa (będzie wybrany format o nazwie zaczynającej się od **OPEQ**...)
Uwagi w nawiasach świadczą, że funkcjonowanie parametru *druk* zależy od przestrzegania konwencji nazewnictwa formatów opisanej przy omawianiu programu **TFORM**;
- dysk* — przyjmuje wartość: **T**
niezależnie od wartości tego parametru tekst wyjściowy będzie wysyłany do zbioru na dysku. Nazwa zbioru będzie równa nazwie bazy, a jego rozszerzenie będzie zależało od wartości parametru *druk*;, gdy *druk* = 3, rozszerzeniem będzie **.TEX**; w przeciwnym przypadku rozszerzeniem będzie **.PRN**;
- dek* — dekodery; teksty pobierane z bazy podlegają dekodowaniu według jednego z 3 dekoderek:
dekoder 0 – zastępuje znaki akcentowane (również polskie litery) przez znaki bez akcentów (a więc: ą będzie zastąpione przez a, ü przez u itp.)
dekoder 1 – zachowuje polskie litery akcentowane zapisane w kodzie *Mazovii* oraz inne litery akcentowane o ile występują w *IBM Character Set #2* poza znakami *Mazovii*, pozostałe znaki akcentowane zastępuje przez znaki bez akcentów,
dekoder 2 – zapisuje pewne znaki specjalne w konwencji edytora T_EX,
dekoder 3 – nic nie dekoduje.
Szczegółowy opis dekoderek — patrz **1.7.1**
- format* — określa *format* sterujący formatowaniem wypisywanych (wyprowadzanych) z bazy informacji. W istocie, do pełnego wyznaczenia formatu wykorzystywany jest również parametr *druk*. Wyznacza on 4 pierwsze znaki nazwy zbioru zawierającego format, zaś wartość parametru *format* dostarcza resztę tej nazwy. Rozszerzeniem nazwy formatu jest **.TTT**. Zbiór zawierający format poszukiwany jest w katalogu bieżącym, następnie w katalogu określonym przez zmienną środowiskową **MAKDFORM** (jeżeli istnieje), a na końcu — w katalogu zawierającym bazę danych;
- oddok* — określa pierwszy wypisywany dokument; jeśli wartością tego parametru jest liczba naturalna, to oznacza ona numer kolejny dokumentu, od którego rozpoczyna

- się przetwarzanie (zgodnie z kolejnością wyznaczoną przez parametr *spwy*); jeśli wartością parametru jest liczba poprzedzona literą 's' (np.: s75), to liczba wskazuje numer systemowy dokumentu, od którego rozpocznie się przetwarzanie (wypisywanie). W ostatnim przypadku (tzn. numeru systemowego) kolejność pobierania dalszych dokumentów jest sekwencyjna, niezależnie od wartości parametru *spwy*;
- ldok* — liczba przetwarzanych (wypisywanych) dokumentów. Jeżeli wartością tego parametru jest '*' oznacza to, że należy przetwarzać dokumenty aż do wyczerpania bazy, indeksu lub listy zaznaczonych dokumentów;
- kop* — przyjmuje wartości:
T – przed przystąpieniem do przetwarzania każdego kolejnego dokumentu na ekranie pojawi się dyspozycja:
 Podaj liczbę kopii, naciśnij ENTER:
 Wprowadzona w odpowiedzi liczba wskaże, ile razy dany dokument ma być przetwarzany (wypisywany). Podana w ten sposób liczba kopii przebija (anuluje) ewentualną liczbę kopii określoną w formacie za pomocą instrukcji <Ofun><5><0>
N – brak możliwości określania liczby kopii przez operatora;
- p1, p2* — wartości tych 2 parametrów są tekstami, które zostaną umieszczone w rejestrach tekstowych *rt₀*, *rt₁* do ewentualnego wykorzystania przez format;
- tytul* — złożony z nie więcej niż 18 znaków tekst, który będzie wyświetlony u góry ekranu (dla identyfikacji procesu drukowania).
 Jeżeli pierwszym znakiem tekstu jest '+', to w formacie mogą występować zagnieźdzone instrukcje warunkowe.
 Jeżeli drugim znakiem tekstu jest '#', to program **MAKD** rozpocznie przetwarzanie bez umożliwienia modyfikacji parametrów oraz zakończy pracę bez zatrzymania umożliwiającego powtórne wykonanie (tzw. praca „bez konwersacji”).
 Jeżeli trzecim znakiem tekstu jest 'T' lub 'N' to, w odpowiedniej sytuacji, nie będzie pojawiał się komunikat:
 DMLch>950, czy dopisywać CRLF? (t/n/N)
 a działanie **MAKD** będzie takie jak gdyby odpowiedzią na powyższy komunikat było zawsze 't' (gdy 3-cim znakiem jest 'T') lub 'n' (gdy 3-cim znakiem jest 'N').
 Ten parametr nie może być modyfikowany w menu.

Po wywołaniu (jeżeli drugim znakiem parametru *tytul* nie jest '#') program **MAKD** wyświetli na monitorze wszystkie podane i niepodane parametry, umożliwiając ich zmianę (z wyjątkiem *tytul*) przed rozpoczęciem pracy. Sygnałem do rozpoczęcia pracy jest naciśnięcie klawisza 'ENTER'.

Po zakończeniu pracy program **MAKD** przechowuje wartości parametrów, z którymi został wywołany w utworzonym zbiorze **MAKD.PAR**. Przechowane wartości są wykorzystywane przy późniejszym wywołaniu **MAKD** bez parametrów jako wartości domyślne.

Jeżeli jakikolwiek parametr wywołania (, z wyjątkiem parametru *tytul*) zaczyna się od znaku '@', to wartość tego parametru pobierana jest ze zbioru **MAKD.PAR** (jeżeli taki zbiór istnieje). Jeżeli zbioru **MAKD.PAR** nie ma, to znak '@' w parametrze jest pomijany.

1.7.1 Szczegółowy opis dekodatorów

Wszystkie dekodery (1 - 3) jednakowo traktują znaki: | (wartość 124), ~ (wartość 126), « (wartość 174) i » (wartość 175), a mianowicie:

znak | – jest pomijany, bądź nie w zależności od wskaźnika W_0 kontrolowanego przez instrukcję <Owsk>;

znak ~ – powoduje, że zarówno ten znak jak i znak po nim następujący nie będą dekodowane;

znak « – powoduje wyłączenie dekodowania;

znak » – powoduje przywrócenie dekodowania (powrót do dekodera aktywnego przed wyłączeniem).

Dekoder 0:

(W nawiasach: przed przecinkiem – wartość dziesiętna znaku, po przecinku – znak po zdekodowaniu)
(128,C), (129,u), (130,e), (131,a), (132,a), (133,a), (134,a), (135,c), (136,e), (137,e), (138,e), (139,i),
(140,i), (141,c), (142,A), (145,e), (146,l), (152,S), (156,L), (158,s), (161,Z), (162,o), (164,n), (166,z),
(167,z), (237,"), (238,").

Dekoder 1:

(W nawiasach: przed przecinkiem – wartość dziesiętna znaku, po przecinku – znak po zdekodowaniu)
(237,"), (238,").

Dekoder 2:

(W nawiasach: przed przecinkiem – znak dekodowany, po przecinku – ciąg znaków po zdekodowaniu)
(#, {\#}), (\$, {\\$}), (% , {\%}), (& , {\&}), (< , \$<\$), (> , \$>\$), ([, {\[}),
(-, \$-\\$), ({ , \$\{ \$), (} , \$\} \$).

1.8 Przykład formatu 1

Poniższy format wybiera z bazy i formatuje informacje o podpolu "1" pola "100". W efekcie otrzymamy tekst złożony z wierszy odpowiadających poszczególnym dokumentom. Każdy wiersz będzie zaczynał się od tekstu "dok.n", gdzie n jest numerem systemowym dokumentu. Dalej będzie wypisana zawartość podpól "1" w polach "100". Nawiasy okrągłe wyznaczają pola, wewnątrz których przecinkami oddzielono wystąpienia podpól. Przy wielokrotnych polach poszczególne ich wystąpienia oddzielone będą średnikami. Przy dokumentach nie zawierających pola "100" wystąpi napis "brak".

Format został przygotowany dla drukarki igłowej pracującej w standardzie EPSON.

```
1      $$$
2      <e>x0<e>!<5><e>0Wystąpienia podpól "1"
3      pola "100"<crlf><crlf><K>
4      <PD>dok.<0drr><0> <K> Początek każdego dokumentu
5      <Np>100<K>
6      <PjI>[<K> Początek pola jednokrotnego
7      <PjK>]<crlf><K> Koniec pola jednokrotnego
8      <PwpI>[<K> Początek 1go wyst. pola wielokrotnego
9      <PwsI>; <K> Początek środk. wyst. pola wielokrotnego
10     <PwoI>; <K> Początek ostatniego. wyst. pola wielokrotnego
11     <PwoK>]<crlf><K> Koniec ostatniego. wyst. pola wielokrotnego
12     <Pbr>brak<crlf><K> Brak pola w dokumencie
13     <Npp>1<K>
14     <PPj>(<Dpp>)<K> Podpole jednokrotne
15     <PPwp>(<Dpp><K> 1sze wyst. podpola wielokrotnego
16     <PPws>, <Dpp><K> środk. wyst. podpola wielokrotnego
17     <PPwo>, <Dpp>)<K> ostatnie wyst. podpola wielokrotnego
18     <PPbr>(<K> brak podpola w dokumencie
19     <KD><K> Koniec każdego dokumentu
20     <EP><ff>
```

Komentarz (grube cyfry oznaczają numery wierszy formatu):

2–3: te dwa wiersze zawierają PROLOG – coś, co wykonuje się raz na początku przetwarzania.

2: wiersz ten zawiera rozkazy drukarki EPSON: <e>x0 – "drukuj w trybie draft"; <e>!<5> – "przejdź na druk skompresowany"; <e>0 – "odstęp między wierszami 1/8 cala".

4: instrukcja <0drr><0> – "wypisz zawartość rejestru 0" (który zawiera systemowy numer dokumentu).

19: <ff> – "przejdź do nowej strony".

Aby skorzystać z powyższego formatu należy:

1. Wprowadzić go (z pominięciem numerów wierszy) do zbioru tekstowego (o nazwie, powiedzmy OPXXPP1.SFO);
2. Przetworzyć go za pomocą programu TFORM (tzn. wykonać instrukcję: TFORM OPXXPP1 EP); w wyniku powstanie zbiór OPEPPP1.TTT będący postacią wyników formatu;

3. Wywołać program MAKD:

```
MAKD nazwa-bazy 0 0 2 T 1 PP1 1 100 N 0 0 PRÓBA
```

Program MAKD umieści tekst dotyczący pierwszych 100 dokumentów bazy na dysku w zbiorze o nazwie *nazwa-bazy*.PRN.

Zmodyfikujmy powyższy format do postaci następującej:

```
1   def $pole$ 100 nazwa pola
2   def $ppole$ 1 nazwa podpola
3   $$$
4   <e>x0<e>!<5><e>0Wystąpienia podpól "$ppole$"
5   pola "$pole$" <CrLf><CrLf><K>
6   <PD>dok.<Odr><O> <K>
7   <Np>$pole$<K>
8   <PjI>[<K>
9   <PjK>] <CrLf><K>
10  <PwpI>[<K>
11  <PwsI>; <K>
12  <PwoI>; <K>
13  <PwoK>] <CrLf><K>
14  <Pbr>brak<CrLf><K>
15  <Npp>$ppole$<K>
16  <PPj>(<Dpp>)<K>
17  <PPwp>(<Dpp>)<K>
18  <PPws>, <Dpp><K>
19  <PPwo>, <Dpp>)<K>
20  <PPbr>(<K>
21  <KD><K>
22  <EP><ff>
```

Otrzymany format działa tak samo jak poprzedni, jednak zmieniając w dwóch pierwszych wierszach nazwy pola i podpola otrzymamy formaty rejestrujące wystąpienia dowolnych pól i podpól w bazach.

1.9 Generowanie formatu wzorcowego

Zawarty w zestawie programów MAKD format GG99 (zbiory OPXXGG99.SFO i OPEPGG99.TTT) może być pomocny w momencie rozpoczynania prac nad nowym formatem. Program MAKD sterowany tym formatem wygeneruje tzw. *format wzorcowy*. Jest to format, który nic nie robi (nie produkuje żadnego wyjścia), ale jest prawidłowo zbudowanym formatem przygotowanym do obsługi wskazanych pól i podpól bazy. Należy teraz wypełnić go treścią.

Aby skorzystać z tego formatu należy postąpić jak następuje:

- Wprowadzić do bazy dokument, w którym trzeba wypełnić czymkolwiek (np. literką 'a') te podpola, które mają być brane pod uwagę.
- Zapamiętać numer systemowy wprowadzonego dokumentu.
- Wywołać MAKD w następujący sposób:

```
MAKD baza 0 0 2 T 1 GG99 snrsys 1 N 0 0
```

snrsys — jest numerem systemowym (np. s75).

W efekcie powstanie zbiór o nazwie *baza*.PRN zawierający prawidłowo zbudowany format, odwołujący się do wybranych podpól. Należy zmienić nazwę tego zbioru na nazwę postaci: OPxxln.SFO i wypełnić go właściwą treścią.

2 Edytor EDYTO

EDYTO jest prostym edytorem redagującym teksty dla drukarek igłowych STAR, EPSON, CITIZEN. Instrukcje sterujące tego edytora zapisuje się w konwencji analogicznej do użytej w *opisie drukowania* MAKD. Program TFORM przetwarza instrukcje EDYTO na postać wewnętrzną, łącznie z instrukcjami opisu drukowania MAKD.

2.1 Instrukcje EDYTO

Redagowaniem tekstu sterują następujące instrukcje:

<seal>< dl >< styl >< nl > ... tekst ... <crLf> — adjustuj do lewej;
<seap>< dl >< styl >< nl > ... tekst ... <crLf> — adjustuj do prawej;
<seac>< dl >< styl >< nl > ... tekst ... <crLf> — centruj;
<seao>< dl >< styl >< nl > ... tekst ... <crLf> — adjustuj obustronnie;
<sein>< lm > — ustaw lewy margines;
<sewc>< lz > — ustaw wcięcie dla wszystkich wierszy akapitu, poczynając od wiersza drugiego;
<senl>< nl > — odstęp pionowy;
<sede>< hw1 >< hw2 >< tsp > — zdefiniuj znaki wypełniacza i twardej spacji.

2.2 Wywołanie programu EDYTO

Program EDYTO wywołuje się następująco:

EDYTO dane uru konw ipar lpt

Opis parametrów wywołania:

dane — nazwa zbioru zawierającego dane (teksty, instrukcje), na których operuje edytor;
uru — przyjmuje wartości:
T – uruchamianie: zamiast na drukarkę, wysyłaj tekst wyjściowy do zbioru EDY.LOG
N – normalna praca: zredagowany test jest drukowany na drukarce;
konw — przyjmuje wartości:
1 – konwersacja: zostanie wyświetlone MENU umożliwiające zmianę dwóch pierwszych parametrów
0 – MENU nie będzie wyświetlone;
ipar — przyjmuje wartości:
1 – generowanie nazwy zbioru danych na podstawie zawartości zbioru MAKD.PAR. Dokładniej, nazwa zbioru danych będzie się składała z nazwy zbioru bazy używanej ostatnio przez program MAKD i przedłużenia .PRN. A więc dane będą pobierane ze zbioru utworzonego przez MAKD wywołanego z parametrem dysk = T. Wartość parametru dane, w tym przypadku, zostanie zignorowana.
0 – parametr dane określa nazwę zbioru, z którego będą pobierane dane;
lpt — przyjmuje wartości:
0 – drukuj na drukarce LPT1
1 – drukuj na drukarce LPT2.

Po wywołaniu programu EDYTO bez parametrów na monitorze zostanie wyświetlone MENU proponujące domyślne wartości pierwszych dwóch parametrów i umożliwiające ich zmianę. Domyślne wartości nie wyświetlanych parametrów są następujące: konw = 1, ipar = 0, lpt = 0.

2.3 Przykład formatu 2

Poniższy format służy do przygotowywania danych dla programu **EDYTO**. Podobnie jak format z paragrafu 1.8 wybiera on z bazy zawartości podpół `$ppole$` z pola `$pole$` stosując analogiczne znaki przestankowe i nawiasy. Teraz jednak każdy dokument nie tworzy oddzielnego wiersza, ale z kolejnych 21 dokumentów tworzy się akapit. Na jednej stronie tworzy się 9 takich akapitów, po czym następuje przejście do następnej strony.

```
1 def $pole$ 100 nazwa pola
2 def $ppole$ 1 nazwa podpola
3 def $dl$ 80 maksymalna długość wiersza akapitu (l.znaków)
4 def $styl$ 1 rodzaj druku (1-normalny, 25-gruby, 65-kursywa)
5 $$$
6 <e>x1<e>!<1><e>0<seac><$dl$><25><2>Wystąpienia podpół "$ppole$"
7 pola "$pole$" <crlf><K>
8 <PD><sewc><5><Orow><4><0><seal><$dl$><$styl$><2><sep><Onast><4>
9 <Orow><4><21><Ozer><4><sep><e>!<25>dok.<Odr><0>:<e>!<1> <K>
10 <Np>$pole$<K>
11 <PjI>[<K>
12 <PjK>] <K>
13 <PwpI>[<K>
14 <PwsI>; <K>
15 <PwoI>; <K>
16 <PwoK>] <K>
17 <Pbr>brak <K>
18 <Npp>$ppole$<K>
19 <PPj>(<Dpp>)<K>
20 <PPwp>(<Dpp>)<K>
21 <PPws>, <Dpp><K>
22 <PPwo>, <Dpp>)<K>
23 <PPbr>()<K>
24 <KD><Orow><4><0><crlf><Onast><5><sep>
25 <Orow><5><9><ff><Ozer><5><sep><K>
26 <EP><ff>
```

Komentarz (grube cyfry oznaczają numery wierszy formatu):

6-7: `<seac><dl><25><2>...<crlf>` – instrukcja EDYTO powodująca wycentrowanie tytułu.

8: `<seal><dl><$styl$><2>` – rozpoczyna akapit, którego zakończenie (`<crlf>`) powstaje w wierszu **24**; `<sewc><5>` powoduje, że wszystkie wiersze akapitu, począwszy od drugiego, są “wcięte” na 5 znaków.

Powyższy format przewiduje zastosowanie dwóch programów: **MAKD** — w celu wybrania informacji z bazy i przygotowania danych dla programu **EDYTO** i programu **EDYTO**, który sformuje dane w akapity i wyśle je na drukarkę igłową. W celu skoordynowania pracy tych dwóch programów wskazane jest ułożyć procedurę-bat, która może mieć następującą postać:

```
1 MAKD nazwa-bazy 0 0 2 T 1 PP2 1 * N 0 0 PRZYKŁAD_2
2 IF errorlevel 1 GOTO KONIEC
3 EDYTO * N 1 1 0 PRZYKŁAD_2
4 :KONIEC
```

Komentarz:

1: wywołanie programu **MAKD** z odpowiednimi parametrami.

2: zbadanie, czy program wywołany w poprzednim wierszu (**MAKD**) wykonał się bezbłędnie; jeżeli tak – przejście do następnego wiersza, jeżeli nie – przejście do wiersza zaczynającego się od “:KONIEC” (a więc zakończenie procedury).

3: wywołanie programu **EDYTO**, który przetworzy dane ze zbioru **nazwa-bazy**.PRN. Zbiór ten został utworzony i wypełniony przez program **MAKD**.

Powyżej założyliśmy, że opisany format został wprowadzony do zbioru **OPXXPP2.SFO** i przetworzony na postać wynikową za pomocą programu **TFORM** (**TFORM** **OPXXPP2** **EP**).

3 Edytor TABLI

Edytor **TABLI** służy do konstruowania tabel, które mogą liczyć wiele stron. Teksty redagowane są dla drukarek igłowych STAR, EPSON, CITIZEN. Instrukcje sterujące tego edytora zapisuje się w konwencji analogicznej do użytej w *opisie drukowania* MAKD. Program **TFORM** przetwarza instrukcje **TABLI** na postać wewnętrzną, łącznie z instrukcjami opisu drukowania MAKD.

Edytor **TABLI** prowadzi 9 rejestrów liczbowych str_1, \dots, str_9 zawierających liczby z przedziału $(-32768, +32767)$. Rejestry te są wstępnie inicjowane przez umieszczenie we wszystkich liczby 1.

Ponadto Edytor **TABLI** prowadzi 10 *długich* rejestrów liczbowych rd_0, \dots, rd_9 zawierających liczby z przedziału $(-2^{31}, +2^{31} - 1)$. Rejestry te są zerowane na początku programu.

3.1 Struktura danych dla TABLI

Dane edytora **TABLI** składają się z tekstów, separatorów i instrukcji sterujących.

Ogólna struktura danych **TABLI**:

```
TYTUŁ<stgp>GÓRNA PAGINA<stdp>DOLNA PAGINA
<stds>DOLNA STOPKA<stza>ZAKOŃCZENIE<stk>
... wiersze tabelki
```

TYTUŁ jest to tekst umieszczany na początku pierwszej strony tabelki;

GÓRNA PAGINA jest to tekst rozpoczynający tabelkę na każdej stronie;

DOLNA PAGINA jest to tekst kończący tabelkę na każdej stronie;

DOLNA STOPKA jest to tekst kończący każdą stronę tabelki. Jest on umiejscowiony, mniej więcej, w tym samym miejscu na każdej stronie. Miejsce to wyznacza parametr *wymiar strony*.

ZAKOŃCZENIE jest to tekst drukowany na ostatniej stronie tabelki zamiast DOLNEJ STOPKI.

ZAKOŃCZENIE może być pominięte (brak <stza>) i wtedy na ostatniej stronie drukuje się DOLNA STOPKA.

3.2 Instrukcje TABLI

Redagowaniem tekstu tabelki sterują następujące instrukcje:

<stk>< dl >< styl >< adju > ... tekst ... — tekst kolumny o szerokości *dl* znaków; *styl* i *adju* określają styl i adjustację;

<stkk>< nl > — znacznik końca wiersza tabelki; *nl* jest liczbą 'nowych linii' po wierszu tabelki. Gdy *nl* = 127 następuje przejście do nowej strony;

<stde>< hw1 >< hw2 >< tsp >< pion > — zdefiniuj znaki *wypełniacza*, *twardej spacji* i *pionowej kreski*;

<stst> — znacznik zastępowany przez bieżący numer strony;

<stbz> — blokuj/odblokuj (działa na zasadzie przełącznika) wpisywanie tekstu do tabelki. Nie blokuje się pobieranie liczb do rejestru (operacje <strd>). Instrukcja przydaje się, gdy trzeba pobrać liczbę do rejestru i dopiero stamtąd wysłać do tabelki;

<stre>< str >< p > — wykonaj operację *p* na rejestrze *str*.
p = 1: w miejscu instrukcji zostanie umieszczona zawartość rejestru *str* tekstowo w postaci dziesiętnej, a następnie zawartość rejestru *str* zostanie zwiększona o 1;

<strd>< rd >< p > — wykonaj operację *p* na rejestrze długim *rd*.
p = 0: wyzeruj rejestr *rd*;
p = 1: wpisz liczbę z bieżącej kolumny do rejestru *rd*. Liczba ta powstaje z cyfr występujących w tekście kolumny;
p = 2: w miejscu instrukcji zostanie umieszczona zawartość rejestru *rd* tekstowo w postaci dziesiętnej;

$p = 3$: w miejscu instrukcji zostanie umieszczona zawartość rejestru rd w postaci *minuty/sekundy* (liczbę w rejestrze traktuje się jako liczbę sekund);

$p = 4$: w miejscu instrukcji zostanie umieszczona zawartość rejestru rd w postaci dziesiętnej z przecinkiem przed dwoma ostatnimi cyframi (tylko wtedy gdy liczba jest co najmniej dwucyfrowa);

$p = 5$: do zawartości rejestru rd zostanie dodane 1, a następnie w miejscu instrukcji zostanie umieszczona zawartość rejestru rd tekstowo w postaci dziesiętnej;

<star>**<ciąg operacji>****K** — wykonaj *ciąg operacji* na rejestrach długich. W *ciągu operacji* mogą występować następujące:

D**<rd₁>****<rd₂>**—dodaj rd_2 do rd_1 , wynik w rd_1

M**<rd₁>****<rd₂>**—pomnóż rd_1 przez rd_2 , wynik w rd_1

R**<rd₁>****<rd₂>**—przenieś rd_2 do rd_1 ,

K—zakończ wykonywanie *ciągu operacji*;

<stty> — instrukcja powodująca wczytanie tekstu z monitora. Może wystąpić w TYTULE lub w GÓRNEJ PAGINIE. W przypadku TYTUŁU, na ekranie pojawi się tekst: 'podaaj tytuł główny ...' a w przypadku GÓRNEJ PAGINY — 'podaaj tytuł bieżący ...'. We wprowadzanym tekście należy zastępować znak *spacji* znakiem *podkreślenia* ('_'). Wprowadzony tekst zastąpi **<stty>** w TYTULE lub GÓRNEJ PAGINIE.

Uwagi:

<styl> — określa się za pomocą wartości stosowanych w rozkazie drukarki **<e>!** (patrz: tabela w 5.1).

<adju> — może przyjmować wartości: l – adjustuj do lewej, p – adjustuj do prawej, c – centruj.

<stst> — może wystąpić tylko w GÓRNEJ lub DOLNEJ PAGINIE oraz w DOLNEJ STOPCE.

Osobliwości:

Wartością parametru dl w instrukcji **<stk>** nie może być 10 ani 13. Aby zdefiniować kolumnę o szerokości 10 lub 13 należy użyć wartości dl 210 lub 213;

Aby użyć jako *pionowej kreski* znaku o wartości 179 należy zdefiniować pionową kreskę o wartości 178.

3.3 Wywołanie programu TABLI

Program **TABLI** wywołuje się następująco:

TABLI *dane wymys pion poz nrst post uru konw ipar lpt tytuł*

Opis parametrów wywołania:

dane — nazwa zbioru zawierającego dane (teksty, instrukcje), na których operuje edytor;

wymys — wymiar strony dla tabelki określony przez liczbę wierszy na stronie przeznaczonych na tabelkę;

pion — przyjmuje wartości:

T – kolumny tabelki oddzielać pionowymi kreskami

N – kolumn tabelki nie oddzielać pionowymi kreskami;

poz — przyjmuje wartości:

T – wiersze tabelki oddzielać poziomymi kreskami

N – wierszy tabelki nie oddzielać poziomymi kreskami;

nrst — numer pierwszej strony tabelki (liczba mniejsza od 10000);

post — liczba pierwszych stron tabelki, które będą pominięte podczas drukowania; (liczba mniejsza od 10000)

uru — przyjmuje wartości:

1 – uruchamianie: zamiast na drukarkę, wysyłaj tekst wyjściowy do zbioru TAB.LOG

0 – normalna praca: tabelka będzie drukowana na drukarce;

- konw* — przyjmuje wartości:
 1 – konwersacja: zostanie wyświetlone MENU umożliwiające zmianę pięciu pierwszych parametrów
 0 – MENU nie będzie wyświetlone;
- ipar* — przyjmuje wartości:
 1 – generowanie nazwy zbioru danych na podstawie zawartości zbioru MAKD.PAR. Wartość parametru *dane*, w tym przypadku, zostanie zignorowana.
 0 – parametr *dane* określa nazwę zbioru, z którego będą pobierane dane;
- lpt* — przyjmuje wartości:
 0 – drukuj na drukarce LPT1
 1 – drukuj na drukarce LPT2;
- tytul* — złożony z nie więcej niż 18 znaków tekst, który będzie wyświetlony u góry ekranu (dla identyfikacji procesu drukowania).

Po wywołaniu, programu **TABLI** bez parametrów na monitorze zostanie wyświetlone MENU proponujące domyślne wartości pierwszych pięciu parametrów i umożliwiające ich zmianę. Domyślne wartości nie wyświetlanych parametrów są następujące: *uru* = 0, *konw* = 1, *ipar* = 0, *lpt* = 0.

W przypadku gdy drukarka nie jest w pełni sprawna, może pojawiać się komunikat żądający jej włączenia, mimo że jest ona włączona. Wtedy, niekiedy, pomaga wciśnięcie klawisza: 'G'.

3.4 Przykład formatu 3

Poniższy format służy do przygotowywania danych dla programu **TABLI**. Wybiera on z bazy zawartości podpól \$ppole1\$ z pola \$pole1\$, \$ppole2\$ z pola \$pole2\$ i \$ppole3\$ z pola \$pole3\$ tworząc dane dla programu **TABLI** w celu utworzenia tabelki, w której kolumny będą odpowiadać polom, a wiersze — dokumentom z bazy.

```

1  def $m_lewy$ 7 lewy margines
2  def $pole1$ 100 nazwa pola 1
3  def $ppole1$ 1 nazwa podpola pola 1
4  def $pole2$ 200 nazwa pola 2
5  def $ppole2$ a nazwa podpola pola 2
6  def $pole3$ 210 nazwa pola 3
7  def $ppole1$ c nazwa podpola pola 3
8  def $slp$ 6 szerokość kolumny 1. porządkowej
9  def $spi$ 20 szerokość kolumny dla pola 1
10 def $sp2$ 25 szerokość kolumny dla pola 2
11 def $sp2$ 21 szerokość kolumny dla pola 3
12 def $st1$ 1 styl tekstu normalnego (1 - 12/cal, 0 - 10/cal)
13 $$$
14 <e>x1<e>0<e>!<$st1$>
15 <e>l<$m_lewy$><crlf>
16 <stde><208><209>~<124>
17 <e>!<1><e>x1                               Tablica zawartości 3 podpól w bazie<crlf>
18 <crlf>
19 <stgp><e>x1<e>!<$st1$>
20                                           <stst><crlf>
21 <e>!<$st1$><crlf>
22 -----<crlf>
23 lp. I   pole 1       I       pole 2       I       pole 3       I<crlf>
24 -----<crlf>
25 <stdp>-----<crlf>
26 <stds><e>x1<crlf>
27 <e>!<$st1$><K>

```

```

28 <PD><K>      Początek dokumentu
29 <Np>$pole1$<K>   pole 1
30 <PjI><stk><$slp$><$st1$>p<stre><1><1><stk><$sp1$><$st1$>c<K>
31 <PwpI><stk><$slp$><$st1$>p<stre><1><1><stk><$sp1$><$st1$>c<K>
32 <PwsI>, <K>
33 <PwoI>, <K>
34 <Pbr><stk><$slp$><$st1$>p<stre><1><1><stk><$sp1$><$st1$>c<K>
35 <Npp>$ppole1$<K>   podpole pola 1
36 <PPj><Dpp><K>
37 <PPwp>(<Dpp><K>
38 <PPws>, <Dpp><K>
39 <PPwo>, <Dpp><K>
40 <Np>$pole2$<K>   pole 2
41 <PjI><stk><$sp2$><$st1$>c<K>
42 <PwpI><stk><$sp2$><$st1$>c<K>
43 <PwsI>; <K>
44 <PwoI>; <K>
45 <Pbr><stk><$sp2$><$st1$>c<K>
46 <Npp>$ppole2$<K>   podpole pola 2
47 <PPj><Dpp><K>
48 <PPwp>(<Dpp><K>
49 <PPws>; <Dpp><K>
50 <PPwo>; <Dpp><K>
51 <Np>$pole3$<K>   pole 3
52 <PjI><stk><$sp3$><$st1$>c<K>
53 <PwpI><stk><$sp3$><$st1$>c<K>
54 <PwsI>; <K>
55 <PwoI>; <K>
56 <Pbr><stk><$sp3$><$st1$>c<K>
57 <Npp>$ppole3$<K>   podpole pola 3
58 <PPj><Dpp><K>
59 <PPwp>(<Dpp><K>
60 <PPws>; <Dpp><K>
61 <PPwo>; <Dpp><K>
62 <KD><stkk><1><K>   Koniec dokumentu
63 <EP><ff>

```

Komentarz (grube cyfry oznaczają numery wierszy formatu):

17–27: wiersze **17–18** zawierają TYTUŁ, wiersze **19–24** — GÓRNĄ PAGINĘ, wiersz **25** zawiera DOLNĄ PAGINĘ, a wiersze **26–27** — DOLNĄ STOPKĘ; w wierszu **20** <stst> wskazuje miejsce numeru strony.

Powyższy format przewiduje zastosowanie dwóch programów: **MAKD** — w celu wybrania informacji z bazy i przygotowania danych dla programu **TABLI** i programu **TABLI**, który sformuje dane w taki sposób, aby powstała tabelka i wyśle ją na drukarkę igłową. W celu skoordynowania pracy tych dwóch programów wskazane jest ułożyć procedurę-bat, która może mieć następującą postać:

```

1   MAKD nazwa-bazy 0 0 2 T 1 PP3 1 * N 0 0 PRZYKŁAD_3
2   IF errorlevel 1 GOTO KONIEC
3   TABLI * 60 T T 1 0 0 1 1 0 PRZYKŁAD_3
4   :KONIEC

```

Powyżej założyliśmy, że opisany format został wprowadzony do zbioru OPXXPP3.SFO i przetworzony na postać wynikową za pomocą programu **TFORM** (TFORM OPXXPP3 EP).

4 Edytor SEGKK

Edytor **SEGKK** służy do dzielenia dużych opisów bibliograficznych na kartki katalogowe o określonej wielkości.

Jeżeli opis pozycji bibliograficznej zostaje podzielony na kilka kartek, to każda z kartek zawiera *tekst kontynuacji*, *główkę* i kolejny fragment *tekstu* opisu bibliograficznego.

4.1 Struktura danych dla SEGKK

Pierwszy wiersz (przez wiersz rozumiemy tutaj ciąg znaków zakończony 'nową linią') danych, który zawiera znak **<sgpd>** jest *tekstem kontynuacji*.

Główka — to ciąg wierszy, z których pierwszy zawiera znak **<sgpg>**, a ostatni — **<sgkg>**. Główka jest częścią tekstu opisu bibliograficznego na pierwszej i na jedynej kartce katalogowej tworzącej opis bibliograficzny.

Tekst za główką, aż do wystąpienia kolejnego *tekstu kontynuacji*, jest w razie potrzeby dzielony na fragmenty wypisywane na kolejnych kartach katalogowych.

Tekst kontynuacji wypisywany jest na początku każdej kartki opisu, ale tylko wtedy, gdy opis nie mieści się na jednej kartce. W tekście kontynuacji znak **<sgpd>** jest wtedy zastępowany przez kolejny numer kartki. W tekście kontynuacji może wystąpić deklaracja **<sgde>< ls >** wpływająca na sposób formatowania główki bądź określająca inne szczegóły formatowania. Tylko pierwsze wystąpienie wiersza z **<sgpd>** jest traktowane jako *tekst kontynuacji*. Pozostałe takie wiersze znaczą jedynie początek nowego opisu i są pomijane.

4.2 Instrukcje, znaczniki SEGKK

- <sgpd>** — początek opisu; pierwszy wiersz zawierający ten znacznik jest traktowany jako *tekst kontynuacji*; podczas druku tekstu kontynuacji **<sgpd>** jest zastępowany przez kolejny numer kartki;
- <sgpg>** — znacznik pierwszego wiersza *główki*;
- <sgkg>** — znacznik ostatniego wiersza *główki*;
- <sgps>** — znacznik proponowanego początku nowej kartki. Wiersz zawierający ten znacznik będzie preferowany jako początkowy na nowej kartce;
- <sgnk>** — znacznik wymuszonego początku nowej kartki. Wiersz zawierający ten znacznik rozpocznie nową kartkę;
- <sgko>** — znacznik końca danych;
- <sgze>** — znak zamieniany na **< NUL >** podczas drukowania;
- <sgcr>** — znak zamieniany na **< CR >** podczas drukowania;
- <sgde>< ls >** — deklaracja:
 - gdy *ls* < 10 — ostatni wiersz *główki* skróć do *ls* słów, resztę zastępując 'ldots'.
 - Gdy *ls* = 11 — opisy nie mieszczące się na jednej kartce redukuj do jednej kartki, pomijając resztę tekstu.
 - Może wystąpić tylko w *tekście kontynuacji*;
- <sgde>< 12 >< nw >** — włącz wskaźnik *nw*. Znaczenie wskaźników:
 - 1 — wyłączony: zastępuj **<sgpg>** i **<sgkg>** przez spację; włączony: pomijaj **<sgpg>** i **<sgkg>** przy drukowaniu *główki*
 - 2 — wyłączony: zastępuj **<sgps>** przez spację; włączony: pomijaj **<sgps>** przy drukowaniu
 - 3 — wyłączony: nic nie rób; włączony: w ostatnim wierszu *główki* usuń tekst występujący po **<sgkg>**.
 - 4 — włączony: powoduje wydłużenie *główki* o jeden pusty wiersz (zapobiega to stykaniu się *główki* z kontynuacją opisu na karcie kontynuacyjnej).Może wystąpić tylko w *tekście kontynuacji*, przed ewentualnym **<sgde>< ls >**;
- <sgtd>< styl >tekst** — *tekst* winien być zbudowany jak następuje:
 - tekst oznaczenia dziurki***<sgcr>***standardowy tekst oznaczenia dziurki***<sgcr>**

instrukcja powoduje zastąpienie standardowego tekstu oznaczania dziurki przez tekst podany przed pierwszym wystąpieniem <sgcr>, a po wydrukowaniu kartki określa nowy tekst standardowy jako tekst zawarty między pierwszym i drugim wystąpieniem <sgcr>. < styl >, określa jaką czcionką ma być drukowany tekst (wg. konwencji opisanej w opisie instrukcji edytora YEDYT. Tekst oznaczania dziurki nie może być dłuższy niż 30 znaków.

4.3 Wywołanie programu SEGKK

Program SEGKK wywołuje się następująco:

SEGKK dane wymk lw lm dm korkr yotw xotw pozk uru konw ipar lpt vtab tytuł gm

Opis parametrów wywołania:

- dane* — nazwa zbioru zawierającego dane (teksty, instrukcje), na których operuje program;
- wymk* — Wymiar pionowy karty — podaje się w postaci liczby cali razy 216. (Np.: dla kartek 2,5 calowych – 540);
- lw* — Liczba wierszy na karcie — liczba ta musi być dzielnikiem liczby podanej jako wartość parametru *wymk*. Program dobierze odstęp między wierszami tekstu do wartości parametrów *wymk* i *lw*;
- lm* — Lewy margines — w postaci liczby znaków;
- dm* — Dolny margines — liczba wierszy pustych na dole kartki;
- korkr* — Korekta położenia kresek — powoduje przesunięcie do góry kresek rozdzielających kartki o *korkr* jednostek $\frac{1}{216}$ cala. Kreski rozdzielające kartki są drukowane tylko, gdy *vtab* > 1 lub *vtab* < -1;
- yotw* — Odległość pionowa znacznika otworu od kresek rozdzielających kartki liczona w jednostkach $\frac{1}{216}$ cala. Kreski rozdzielające kartki oraz znacznik otworu są drukowane tylko, gdy *vtab* > 1 lub *vtab* < -1;
- xotw* — Odległość pozioma znacznika otworu od lewego marginesu liczona w jednostkach $\frac{1}{60}$ cala. Kreski rozdzielające kartki oraz znacznik otworu są drukowane tylko, gdy *vtab* > 1 lub *vtab* < -1;
- poz* — Pozycja pierwszej kartki na stronie (wartości: 1, 2, 3, ...). Parametr ten pozwala wykorzystywać niepełne strony pozostałe poprzednim drukowaniu;
- uru* — przyjmuje wartości:
 1 – uruchamianie: zamiast na drukarkę, wysyłaj tekst wyjściowy do zbioru SEG.LOG
 0 – normalna praca: zredagowany test jest drukowany na drukarce;
- konw* — przyjmuje wartości:
 1 – konwersacja: zostanie wyświetlone MENU umożliwiające zmianę pięciu pierwszych parametrów
 0 – MENU nie będzie wyświetlone;
- ipar* — przyjmuje wartości:
 1 – generowanie nazwy zbioru danych na podstawie zawartości zbioru MAKD.PAR. Dokładniej, nazwą zbioru danych będzie nazwa bazy używanej ostatnio przez program MAKD z przedłużeniem .PRN. A więc dane będą pobierane ze zbioru utworzonego przez MAKD wywołanego z parametrem *dysk* = T. Wartość parametru *dane*, w tym przypadku, zostanie zignorowana.
 0 – parametr *dane* określa nazwę zbioru, z którego będą pobierane dane.
 2 – tak, jak wartość 0, a dodatkowo blokada “spychania do dołu” tekstu na ostatniej (niepełnej) kartce.
 3 – tak, jak wartość 1, a dodatkowo blokada “spychania do dołu” tekstu na ostatniej (niepełnej) kartce;
- lpt* — przyjmuje wartości:
 0 – drukuj na drukarce LPT1
 1 – drukuj na drukarce LPT2;

- vtab*k* — postać dwuczłonowa: dwie liczby (*vtab*, *k*) rozdzielone gwiazdką. Człon „*k”, jeżeli istnieje, oznacza liczbę kresek tworzących *znacznik podziału* (domyślne: 10). Człon *vtab* przyjmuje wartości:
- 0 – program definiuje stronę drukarki równą wymiarowi kartki; przejście do nowej kartki za pomocą znaku < FF >;
 - 1 – kartki powstają za pomocą tabulacji pionowej; przejście do nowej kartki za pomocą znaku < VT >;
 - 2, 3, ..., 9 – na fizycznej stronie będzie się drukować *vtab* kartek; między kartkami pojawiają się *znaczniki podziału*; po każdym *vtab* kartkach, wysyłany jest do drukarki rozkaz *nowa strona*;
 - 12, 13, 14 – na fizycznej stronie będzie się drukować 2, 3, 4 kartki; między kartkami pojawiają się *znaczniki podziału*, przy czym można definiować *tekst oznaczenia dziurki* za pomocą rozkazu < sgt d >; rozkaz *nowa strona* nie jest wysyłany do drukarki;
 - *vtab* jest liczbą poprzedzoną znakiem '-' (minus) – znaczenie jak wyżej, po usunięciu znaku '-' jednak dla ustalania wielkości strony fizycznej używane będą jednostki $\frac{1}{180}$ cala (tej opcji należy używać dla większości drukarek 24-igłowych);
- tytul* — złożony z nie więcej niż 18 znaków tekst, który będzie wyświetlony u góry ekranu (dla identyfikacji procesu drukowania);
- gm* — Górny margines — liczba pustych wierszy u góry kartki.

Po wywołaniu programu **SEGKK** bez parametrów na monitorze zostanie wyświetlone MENU proponujące domyślne wartości pierwszych pięciu parametrów i umożliwiające ich zmianę. Domyślne wartości nie wyświetlanych parametrów są następujące: *uru* = 0, *konw* = 1, *ipar* = 0, *lpt* = 0, *vtab* = 0.

W przypadku gdy drukarka nie jest w pełni sprawna, może pojawiać się komunikat żądający jej włączenia, mimo że jest ona włączona. Wtedy, niekiedy, pomaga wciśnięcie klawisza: 'G'.

5 Edytor YEDYT

YEDYT jest edytorem, w którym można sterować rozmieszczeniem na stronie drukowanej akapitów za pomocą współrzędnych poziomych (*x*) i pionowych (*y*). Akapity są to ciągi wierszy sformatowane, za pomocą instrukcji `<seal>`, `<seap>`, `<seac>` lub `<seao>`.

Punkt odniesienia (początek układu) jest umieszczony w lewym górnym rogu kartki przesuniętym o, właściwe drukarce, marginesy lewy i górny. Współrzędne *y* rosną od góry do dołu kartki zaś współrzędne *x* – od lewej do prawej. Jednostką dla współrzędnej *x* jest $\frac{1}{60}$ cala, a dla współrzędnej *y* — $\frac{1}{216}$ cala.

Współrzędne *x*, *y* określają, w którym miejscu na stronie ma się rozpocząć drukowanie najbliższego akapitu. Kolejne wiersze akapitu będą się zaczynały w miejscach o nie zmienionej współrzędnej *x* i zwiększonej o wartość *skoky* współrzędnej *y*. Instrukcja `<sevs>` służy do określania wartości *skoky*.

Można drukować tekst na stronie w kilku *kolumnach* określonych przez instrukcję `<sekl>`. Bliższe informacje na ten temat podane są w opisie wspomnianej instrukcji. Gdy nie użyjemy instrukcji `<sekl>`, tekst będzie składany w jednej kolumnie.

Tekst do wydrukowania na stronie kompletuje się w pamięci maszyny, a wydruk następuje w następujących przypadkach:

1. gdy pojawi się instrukcja `<seff>`,
2. gdy pojawi się instrukcja `<sens>` (*nowa strona*),
3. gdy, podczas składania ostatniej kolumny, współrzędna *y* przekroczy wartość *maxy* określoną przez instrukcję `<semy>` (gdy kolumna nie jest ostatnia, powyższa sytuacja spowoduje przejście do następnej kolumny),
4. gdy, podczas składania ostatniej kolumny, pojawi się instrukcja `<senw>`, a współrzędna *y* jest większa od wartości *wmaxy* określonej przez instrukcję `<semy>` (gdy kolumna nie jest ostatnia, powyższa sytuacja spowoduje przejście do następnej kolumny).

Przypadki 2, 3 i 4 powodują przejście do nowej strony z jednoczesnym nadaniem wartości początkowej współrzędnej *y* (równej *margo* – margines górny) oraz wyzerowaniem zmiennej *yp*, określającej współrzędną *y* punktu odniesienia. Współrzędna *x* nie ulega zmianie przy przejściu do następnej strony.

Instrukcja `<seff>` nie powoduje przejścia do nowej strony i nie powoduje zmiany wartości współrzędnej *y* i *yp*.

Zmienna *yp* zmienia wartość w trakcie drukowania kolejnych wierszy określając zawsze współrzędną *y* ostatnio wydrukowanego wiersza. Jeżeli, zatem, po instrukcji `<seff>` pozostawimy zmienną *yp* bez zmiany, to możemy kontynuować drukowanie, na tej samej stronie, w starym układzie współrzędnych. Jeżeli jednak, po instrukcji `<seff>`, wyzerujemy tę zmienną (za pomocą instrukcji `<seyp>`), to dalszy druk będzie odbywał się w nowym układzie współrzędnych, którego punkt odniesienia będzie umiejscowiony na poziomie ostatniego wydrukowanego wiersza.

5.1 Instrukcje YEDYT

Redagowaniem tekstu sterują następujące instrukcje:

5.1.1 Instrukcje konstruowania akapitów

`<seal>< dl >< styl >< nl > ... tekst ... <crlf>` — utwórz akapit adjustowany do lewej;
`<seap>< dl >< styl >< nl > ... tekst ... <crlf>` — utwórz akapit adjustowany do prawej;
`<seac>< dl >< styl >< nl > ... tekst ... <crlf>` — utwórz akapit wycentrowany;
`<seao>< dl >< styl >< nl > ... tekst ... <crlf>` — utwórz akapit adjustowany obustronnie;
`<sepu>< og > ... tekst ... <og >` — pusty akapit: pomiń 'tekst' wraz z ogranicznikami *og* (*og* – dowolny znak);
`<sede>< hw1 >< hw2 >< tsp >` — zdefiniuj znaki *wypełniacza* i *twardej spacji*;

- <sein>< lm > — ustaw lewy margines dla kolejnych akapitów; *lm* — liczba znaków. Lewy margines można określić bardziej precyzyjnie za pomocą instrukcji <sexx>;
- <sewc>< lz > — ustaw wcięcie dla wszystkich wierszy akapitu poczynając od wiersza drugiego; *lz* — liczba znaków;
- <senl>< nl > — odstęp pionowy;
- <sere>< sgr >< p > — wykonaj operację *p* na rejestrze *sgr*.
p = 1: w miejscu instrukcji zostanie umieszczona zawartość rejestru *sgr* (tekstowo, w postaci dziesiętnej), a następnie zawartość rejestru *sgr* zostanie zwiększona o 1;
p = 2: jak przy *p* = 1, ale bez zmieniania zawartości rejestru *sgr*;
p = 3: w miejscu instrukcji zostanie umieszczony znak, którego wartość ASCII jest zapisana w rejestrze *sgr*;
- <serd>< rd >< p > — wykonaj operację *p* na rejestrze długim *rd*.
p = 1: w miejscu instrukcji zostanie umieszczona zawartość rejestru *rd* (tekstowo, w postaci dziesiętnej);
- <sera>< sgr >< n > — w miejscu instrukcji zostanie umieszczona zawartość rejestru *sgr* tekstowo, w postaci dziesiętnej i adjustowana do prawej w *n*-znakowym polu (uzupełniona spacjami od lewej). Jeżeli liczba ma więcej niż *n* cyfr – ucięte zostaną cyfry najstarsze. Następnie zawartość rejestru *sgr* zostanie zwiększona o 1;
- <seda>< rd >< n > — w miejscu instrukcji zostanie umieszczona zawartość rejestru długiego *rd* tekstowo, w postaci dziesiętnej i adjustowana do prawej w *n*-znakowym polu (uzupełniona spacjami od lewej). Jeżeli liczba ma więcej niż *n* cyfr – ucięte zostaną cyfry najstarsze;
- <sedb>< rd >< n > — działa jak <seda> z tym, że 2 ostatnie cyfry liczby będą oddzielone od pozostałych przecinkiem. (Przecinek będzie dodany tylko dla liczb co najmniej 2-cyfrowych.);
- <sepo>< n > — początek sekwencji, która będzie przetworzona za pomocą funkcji FUN*n*;
- <seko> — koniec sekwencji rozpoczętej <sepo>< n >.

W powyższych instrukcjach:

dl — oznacza maksymalną długość wiersza akapitu liczoną w znakach. Przejście do nowego wiersza następuje na spacji poprzedzającej słowo, które nie zmieści się w wierszu. Gdy takiej spacji nie ma, słowo zostanie złamane. Adjustacja dokonuje się po podziale akapitu na wiersze.

styl — jest liczbą określającą rodzaj czcionki, która będzie użyta przy drukowaniu akapitu. Liczba ta powstaje ze zsumowania wartości przydzielonych cechom druku według następującej tabelki:

10 zn. na cal	0
12 zn. na cal	1
druk proporcjonalny	2
druk skompresowany	4
druk wytłuszczony	8
druk pogrubiony	16
druk podwójnej szerokości	32
kursywa	64
podkreślanie	128

wewnątrz akapitu można zmieniać czcionkę za pomocą instrukcji <e>!< styl >.

nl — określa odstęp pionowy po ostatnim wierszu akapitu. Jego wartość wyniesie *nl* × *skoky* (*skoky* określa instrukcja <sevs>).

tsp — znak *twardej spacji* w druku będzie zastąpiony przez spację, ale przy podziale akapitu na wiersze jest traktowany jako część słowa (zapobiega łamaniu wiersza).

hw1, *hw2* — znaki *wypełniacza* funkcjonują jedynie w akapitach adjustowanych obustronnie (instrukcja <seao>). Powodują one równomierne „rozepchnięcie” tekstu w miejscach swego występowania tak, aby wiersz stał się dokładnie długości *dl*. Znak *hw2* jest „mocniejszy” od znaku *hw1* w tym sensie, że wystąpienie znaku *hw2* w jakimkolwiek wierszu akapitu powoduje, że działanie znaków *hw1* w tym wierszu zostanie zredukowane do zastąpienia ich przez pojedynczą spację.

sgr — YEDYT prowadzi 9 rejestrów sgr_1, \dots, sgr_9 , które mogą zawierać liczby z przedziału $(-32768, +32767)$. Rejestry sgr_1 i sgr_2 otrzymują wartość początkową przy wywołaniu programu YEDYT (patrz 5.2).

rd — YEDYT prowadzi 9 rejestrów długich rd_1, \dots, rd_9 , które mogą zawierać liczby z przedziału $(-2^{31}, +2^{31} - 1)$.

funkcje FUN*n*:

FUN1 — przetwarza tekst postaci **hh.mm.ss** na postać: **hh.mm'ss**"

FUN2 — przetwarza tekst postaci **hhmmss** na postać: **hh.mm'ss**"

5.1.2 Instrukcje ustawiania współrzędnych

<sexx>< @wx > — nadaj współrzędnej *x* wartość < wx >;

<seyy>< @wy > — nadaj współrzędnej *y* wartość < wy >;

<seyp>< @yp > — określ współrzędną *y* punktu odniesienia, nadaj jej wartość *yp*;

<sevs>< @skoky > — nadaj wartość zmiennej *skoky* (odstęp między wierszami).

5.1.3 Instrukcje konstrukcji strony

<semy>< @maxy >< @wmaxy > — ustaw zmienne *maxy* i *wmaxy* określające sytuację, w których następuje automatyczne przejście do nowej strony;

<semg>< @margo > — ustaw współrzędną *y* marginesu górnego *margo*; *margo* jest wartością początkową współrzędnej *y* na początku strony;

<sens> — przejdź do nowej strony bezwarunkowo;

<senk> — przejdź do nowej kolumny na stronie – bezwarunkowo;

<senw> — przejdź do nowej kolumny na stronie, gdy współrzędna *y* jest większa od *wmaxy*;

<senr>< nrstr > — kolejna strona otrzyma numer *nrstr*; zmienna *nrstr* jest automatycznie zwiększana o 1 po każdym przejściu do nowej strony;

<seff> — wydrukuj dotychczas skompletowany tekst bez przechodzenia do nowej strony.

5.1.4 Instrukcja definiowania paginy

<sepa>< @pwspx >< @pwspy >< dl >< styl > ... tekst ... <crlf> — określ tekst paginy. Parametry *pwspx* i *pwspy* określają współrzędne na stronie tekstu paginy; pozostałe parametry określają wiersz paginy, który będzie sformatowany według zasad instrukcji <seao>. Ponadto, jeżeli w tekście wystąpi znak '%' to zostanie on zastąpiony przez bieżący numer strony (wartość zmiennej *nrstr*).

5.1.5 Instrukcje działań na rejestrach

<sear>< sgr >liczba<ciąg operacji>K — Instrukcja ta powoduje (a) wpisanie wartości *liczba* do rejestru *sgr*, a następnie (b) wykonanie *ciągu operacji* na rejestrach.

liczba jest ciągiem cyfr zamienianym na wartość liczbową.

W *ciągu operacji* mogą występować następujące:

D< n >liczba — dodaj do zawartości rejestru sgr_n liczbę *liczba*

O< n >liczba — odejmij od zawartości rejestru sgr_n liczbę *liczba*

M< n >liczba — pomnóż zawartość rejestru sgr_n przez liczbę *liczba*

Q< n >liczba — podziel zawartość rejestru sgr_n przez liczbę *liczba*, iloraz zostaw w sgr_n , a resztę w sgr_{n+1}

R< n1 >< n2 > — przenieś zawartość rejestru sgr_{n2} do sgr_{n1}

X< n > — nadaj współrzędnej *x* wartość równą zawartości rejestru sgr_n

x< n > — wpisz bieżącą wartość współrzędnej *x* do rejestru sgr_n

Y< n > — nadaj współrzędnej *y* wartość równą zawartości rejestru sgr_n

y< n > — wpisz bieżącą wartość współrzędnej *y* do rejestru sgr_n

K — zakończ wykonywanie *ciągu operacji*. Jest to również oznaczenie końca instrukcji <sear>

<sead>< rd >liczba<ciąg operacji>K — Instrukcja ta powoduje (a) wpisanie wartości *liczba* do rejestru długiego *rd*, a następnie (b) wykonanie *ciągu operacji* na rejestrach.

liczba jest ciągiem cyfr zamienianym na wartość liczbową.

W *ciągu operacji* mogą występować następujące:

A< rd₁ >< rd₂ > — dodaj do zawartości rejestru *rd₁* liczbę z rejestru *rd₂*. Wynik – w *rd₂*.

K — zakończ wykonywanie *ciągu operacji*. Jest to również oznaczenie końca instrukcji <sead>

5.1.6 Instrukcja definiowania kolumn na stronie

<sekl>< lk >< @kl₁ >...< @kl_{lk} > — instrukcja ta określa liczbę kolumn (*lk*) oraz współrzędne *x* początków kolumn (@kl₁, ... @kl_{lk}). Przejście do nowej kolumny, gdy zaistnieją odpowiednie warunki, polega na zwiększeniu współrzędnej *x* o szerokość kolumny poprzedniej oraz nadaniu współrzędnej *y* wartości *margo* (margines górny ustalony instrukcją <semg>).

5.2 Wywołanie programu YEDYT

Program YEDYT wywołuje się następująco:

YEDYT dane nrstr licz1 licz2 post uru konw ipar lpt tytuł

Opis parametrów wywołania:

- dane* — nazwa zbioru zawierającego dane (teksty, instrukcje), na których operuje edytor;
- nrstr* — numer strony od którego rozpocznie się numerowanie stron w przypadku stosowania automatycznej paginy (instrukcja <sepa>);
- licz1* — wartość początkowa dla rejestru *sgr₁*;
- licz2* — wartość początkowa dla rejestru *sgr₂*;
- post* — liczba pierwszych stron, które będą pominięte podczas drukowania;
- uru* — przyjmuje wartości:
 - 0 – normalna praca: zredagowany tekst jest drukowany na drukarce
 - 1 – uruchamianie: zamiast na drukarkę, wysyłaj tekst wyjściowy do zbioru EDY.LOG
 - 2 – wyświetlaj na ekranie teksty pojawiające się między akapitami. W zasadzie, nie powinno ich być;
- konw* — przyjmuje wartości:
 - 1 – konwersacja: zostanie wyświetlone MENU umożliwiające zmianę dwóch pierwszych parametrów
 - 0 – MENU nie będzie wyświetlone;
- ipar* — przyjmuje wartości:
 - 1 – generowanie nazwy zbioru danych na podstawie zawartości zbioru MAKD.PAR. Dokładniej, nazwą zbioru danych będzie nazwa zbioru bazy używanej ostatnio przez program MAKD z przedłużeniem .PRN. A więc dane będą pobierane ze zbioru utworzonego przez MAKD wywołanego z parametrem *dysk* = T. Wartość parametru *dane*, w tym przypadku, zostanie zignorowana.
 - 0 – parametr *dane* określa nazwę zbioru, z którego będą pobierane dane;
- lpt* — przyjmuje wartości:
 - 0 – drukuj na drukarce LPT1
 - 1 – drukuj na drukarce LPT2;
- tytuł* — złożony z nie więcej niż 18 znaków tekst, który będzie wyświetlony u góry ekranu (dla identyfikacji procesu drukowania).

Po wywołaniu, programu YEDYT bez parametrów na monitorze zostanie wyświetlone MENU proponujące domyślne wartości pierwszych czterech parametrów i umożliwiające ich zmianę. Domyślne wartości nie wyświetlanych parametrów są następujące: *uru* = 0, *konw* = 1, *ipar* = 0, *lpt* = 0.

W przypadku gdy drukarka nie jest w pełni sprawna, może pojawiać się komunikat żądający jej włączenia, mimo że jest ona włączona. Wtedy, niekiedy, pomaga wciśnięcie klawisza: 'G'.

5.3 Przykład formatu 4

Poniższy format służy do przygotowywania danych dla programu YEDYT. Wybiera on z bazy zawartości podpół \$ppole\$ z pola \$pole\$ tworząc dane dla programu YEDYT w celu wydrukowania w dwóch kolumnach. W prezentowanej wersji, wybierane są zawartości podpół a pola 200 (w formacie MARC-BN jest to tytuł książki).

```

1 def $gm$ {EP}100{EQ}83 górny margines
2 def $lm$ 5 lewy margines w znakach
3 def $kol2$ 180 współrzędna x 2 kolumny w 1/60 cala
4 def $pag$ {EP}2200{EQ}1833 pozycja paginy na stronie
5 def $zks$ {EP}2000{EQ}1666 zbliżający kon. str. od góry
6 def $kst$ {EP}2100{EQ}1750 koniec strony od góry
7 def $ods$ {EP}30{EQ}25 odstęp między wiersz.
8 def $dl$ 30 maksymalna długość wiersza (w kolumnie)
9 def $s2$ 75 szerokość tekstu paginy
10 def $styl$ 1 rodzaj druku (1-normalny, 25-gruby, 65-kursywa)
11 def $pole$ 200 nazwa pola
12 def $ppole$ a nazwa podpola
13 def $wc1$ 3 wcięcie dla tekstów dłuższych od szer.kolumny
14 $$$
15 <e>1<$lm$><e>x1
16 <sekl><2><0><$kol2$>
17 <sepa><0><$pag$><$s2$><25><224>- % -<224><crlf>
18 <semy><$kst$><$zks$><sevs><$ods$><semg><$gm$><sey><50>
19 <seac><$s2$><25><2><0cekr><6>tytułu<crlf>
20 <sede><224><225>~<senw><sewc><$wc1$><K>
21 <PD><K> Początek dokumentu
22 <Np>$pole$<K> nazwa pola
23 <Npp>$ppole$<K> nazwa podpola
24 <PPj><senw><seal><$dl$><$styl$><1><Dpp><crlf><K>
25 <PPwp><senw><seal><$dl$><$styl$><1><Dpp><crlf><K>
26 <PPws><senw><seal><$dl$><$styl$><1><Dpp><crlf><K>
27 <PPwo><senw><seal><$dl$><$styl$><1><Dpp><crlf><K>
28 <KD><K> Koniec dokumentu
29 <EP><sens><crlf><e>M

```

Komentarz (grube cyfry oznaczają numery wierszy formatu):

1, 4–7: wiersze te zawierają tzw. warunkowe definicje symboli. Np., w wierszu 4 symbol \$pag\$ otrzyma wartość 2200 dla drukarki EP (Epson 9 igieł), a — 1833 dla drukarki EQ (Epson 24 igły). Parametr ten określa położenie paginy (wiersza zawierającego numer strony) na stronie, określając jego odległość od górnego marginesu w jednostkach $\frac{1}{216}$ cala, dla drukarki 9-igłowej, i — $\frac{1}{180}$ cala, dla drukarki 24-igłowej. Drukarki te bowiem operują różnymi jednostkami przy określaniu odstępów pionowych.

16: instrukcja <sekl><2><0><\$kol2\$> realizowana przez program YEDYT mówi, że tekst ma być składany w dwóch kolumnach, których lewe krawędzie będą oddalone od lewego marginesu strony o 0 i 180 jednostek, odpowiednio. Jednostką, tym razem (jest to właściwość drukarki Epson), jest $\frac{1}{60}$ cala.

17: instrukcja <sepa> określa budowę paginy: % — będzie zastąpiony przez bieżący numer strony, <224> — znak "wypełniacza", spowoduje, że tekst zawarty między tymi znakami będzie wycentrowany.

18: instrukcja <semy> określa, w jakich sytuacjach nastąpi automatyczne przejście do nowej strony; instrukcja <sevs> określa wielkość odstepu między wierszami w jednostkach zależnych od typu drukarki.

Powyższy format przewiduje zastosowanie dwóch programów: **MAKD** — w celu wybrania informacji z bazy i przygotowania danych dla programu YEDYT i programu YEDYT, który sformuje dane w taki sposób, aby powstał tekst dwukolumnowy i wysła go na drukarkę igłową. W celu skoordynowania pracy tych dwóch programów wskazane jest ułożyć procedurę-bat, która może mieć następującą postać:

```
1   MAKD nazwa-bazy 0 0 2 T 1 PP4 1 * N 0 0 PRZYKŁAD_4
2   IF errorlevel 1 GOTO KONIEC
3   YEDYT * 1 1 1 0 0 1 1 0 PRZYKŁAD_4
4   :KONIEC
```

Powyżej założyliśmy, że opisany format został wprowadzony do zbioru OPXXPP4.SFO i przetworzony na postać wynikową za pomocą programu **TFORM** (TFORM OPXXPP4 EP).

6 Program SKSORT

Program SKSORT sortuje zbiory o określonej niżej strukturze.

Zbiór wejściowy powinien posiadać następującą strukturę:

PROLOG **sep1** ciąg rekordów wej. **sep1** EPILOG

sep1 jest dowolnym wybranym znakiem kodu 8 bitowego;

PROLOG jest dowolnym tekstem nie zawierającym znaku **sep1**;

EPILOG jest dowolnym tekstem.

ciąg rekordów wej. składa się z rekordów, z których każdy ma następującą strukturę:

KLUCZ **sep2** TEKST **sep3**

sep2 i **sep3** – dowolnie wybrane znaki kodu 8 bitowego;

KLUCZ jest dowolnym tekstem nie zawierającym znaków **sep2** i **sep3**;

TEKST jest dowolnym tekstem nie zawierającym znaków **sep3**;

rekordy nie mogą zawierać znaku **sep1**.

Wynikiem sortowania jest zbiór wyjściowy w postaci:

PROLOG ciąg rekordów wyj. EPILOG

PROLOG i EPILOG są skopiowane ze zbioru wejściowego, zaś ciąg rekordów wyj. jest ciągiem wybranych i posortowanych według kluczy TEKST'ów z rekordów wejściowych.

W PROLOGu można umieścić do 5 tekstów (do 24 znaków każdy), które zostaną usunięte z PROLOGu ale zachowane do wykorzystania w *przetwarzaniu końcowym*. Tekstami tymi są teksty zawarte między dwoma znakami **sep4**. Kolejne takie teksty (o ile wystąpią) oznaczać będziemy *BLOT₀, . . . , BLOT₄*.

Opcjonalnie, TEKST rekordu może mieć strukturę:

sep6 TW **sep6** TEKSTW

TW – tekst wiodący (do 49 znaków),

TEKSTW – tekst właściwy rekordu.

W trakcie przetwarzania końcowego (patrz niżej) teksty TW (wraz z separatorami **sep6**) są usuwane, ale zaznaczane są momenty zmiany tekstu TW. Mianowicie, przed rekordem zawierającym zmieniony (w stosunku do poprzedniego rekordu) tekst TW zostanie dopisany tekst *BLOT₀* uprzednio zdefiniowany w PROLOGU.

Przykład zastosowania: jeżeli *tekstem wiodącym* będzie pierwsza litera hasła zawartego w TEKSTW to powyższy mechanizm pozwoli na jakieś zaznaczenie początku nowej litery w indeksie haseł.

Proces sortowania odbywa się w 3 etapach:

- przygotowanie do sortowania, na które składają się:
 - weryfikacja rekordów,
 - kodowanie kluczy.
- sortowanie właściwe
- wyprowadzanie, które może zawierać tzw. *przetwarzanie końcowe*.

Weryfikacja rekordów dokonywana jest przez wybraną (parametr programu) *funkcję weryfikującą*. Aktualnie dostępne są następujące funkcje weryfikujące:

WER0 – weryfikacja „pusta”, wszystkie rekordy są akceptowane bez zmian

WER1 – zakłada się, że klucze sortowania są postaci: *l.liczba* lub *liczba*, gdzie *l* oznacza pojedynczą literę. Podczas weryfikacji pierwszego rekordu, operator będzie zmuszony do określenia litery *l* lub jej braku w odpowiedzi na komunikat *Podaj prefiks . . .* lub *** (*** - oznacza brak prefiksu). Następnie będą pomijane rekordy, których prefiks (lub jego brak) nie zgadzają się z określonym.

Kodowanie kluczy wyznacza porządek sortowania. Można wybrać (parametr programu) jedną z następujących *funkcji kodujących*:

KOD0 – kodowanie „puste”, bez zmian;

KOD1 – wprowadza następujący porządek sortowania:

min spacja aąbcćdeęfghijklmnoópqrsstuvwxýzz0123456789 *max*

Ponadto, utożsamione zostają duże i małe litery oraz, przed zakodowaniem:

- znaki ! ' [] { } zostają pominięte,
- znak \ zostaje pominięty wraz ze znakiem występującym po nim,
- pozostałe znaki zostają najpierw utożsamione ze *spacją*, a następnie ciągi *spacji* zredukowane do jednej *spacji*;
- *min*, *max* – są to znaki o wartości (w kodzie ASCII) odpowiednio 1 i 2. W formatach przetwarzanych programem **TFORM** można stosować zapis *< min >*, *< max >*.

Zakłada się, że na wejściu, polskie litery są zapisane w kodzie MAZOVII (dla LATIN II - patrz *KOD33*).

KOD2 – dokonuje takiego samego kodowania jak funkcja *KOD1* z tym, że dodatkowo, przed zakodowaniem zostają usunięte z klucza znaki zawarte między *^* i **sep5** włącznie. Gdy usuwaną sekwencję kończy podwójny znak **sep5** – zostaną usunięte oba. Domyślną wartością znaku **sep5** jest '|';

KOD3 – wprowadza następujący porządek sortowania:

min spacja, znaki przestankowe (w kolejności kodu ASCII)

AaĄąBbCcĆćDdEeĘęFfGgHhIiJjKkLlŁłMmNn

ŃńOoÓóPpQqRrSsŚśTtUuVvWwXxYyZzŻżŹź

0123456789 *max*

Spacje ani znaki przestankowe nie są redukowane. Znaki *min*, *max* — patrz opis *KOD1*.

KOD33 – dokonuje takiego samego kodowania jak funkcja *KOD1* ale dla danych z polskimi literami w kodzie LATIN II.

KOD34 – dokonuje takiego samego kodowania jak funkcja *KOD2* ale dla danych z polskimi literami w kodzie LATIN II.

KOD35 – dokonuje takiego samego kodowania jak funkcja *KOD3* ale dla danych z polskimi literami w kodzie LATIN II.

Funkcje kodujące działają w oparciu o *tablice kodujące* (patrz Dodatek D.), które można modyfikować określając *zbiór zmian* w tablicy (patrz 6.1 - opis parametru *kod*).

Podczas wyprowadzania posortowanych TEKST'ów do zbioru wyjściowego może być wykonywane tzw. *przetwarzanie końcowe* za pomocą jednej z niżej opisanych funkcji. W trakcie przetwarzania końcowego uwzględniany jest *tryb przetwarzania końcowego*, który wpływa na sposób porównywania tekstów. (Wybór funkcji i trybu przetwarzania końcowego – patrz 6.1.)

Funkcje przetwarzania końcowego:

PRZET0 - przetwarzanie „puste”, bez zmian;

PRZET1 – zakłada się, że przetwarzane rekordy zawierają separator **sep4**, a przed **sep3** występuje para CR,LF. Jeżeli w kolejnych rekordach sekwencje poprzedzające **sep4** są identyczne, to część końcową (po **sep4**) drugiego rekordu dołącza się do pierwszego rekordu usuwając z niego CR,LF i poprzedzając ' ; '. Ten proces powtarza się z następnymi rekordami aż wystąpi zmiana części pierwszej (przed **sep4**) rekordu. Wtedy zaczyna się kompletowanie następnego rekordu. Liczba połączonych w ten sposób rekordów zostaje zapamiętana i zostanie umieszczona w wyprowadzonym tekście w miejsce znacznika **zlwyst** (jeżeli taki znacznik wystąpi).

Możliwe jest ograniczenie liczby zbieranych w powyższy sposób rekordów do

liczby, po przekroczeniu której dalsze rekordy będą pomijane aż do wystąpienia zmiany części pierwszej rekordu (patrz 6.1 – opis parametru *zlwy*);

PRZET2 – proces przetwarzania jest analogiczny jak w funkcji *PRZET1* z tym, że nie usuwa się CR,LF, a dopisuje *spację* (zamiast ‘,’);

PRZET3 – jak *PRZET1*, — bez poprzedzania ‘,’).

PRZET4 – zakłada się, że przetwarzane rekordy są zbudowane jak następuje:

tekst1 **sep4** tekst2 **sep4** tekst3 **sep4** tekst

Podczas wyprowadzania kolejnych rekordów, pomija się powtarzające się ‘tekst1’, ‘tekst2’ lub ‘tekst3’. Przykład zastosowania tej funkcji można wyobrazić gdy przyjmujemy, że tekst1,2,3 zawierają odpowiednio tytuły działów, poddziałów i podpoddziałów jakiejś bibliografii.

PRZET5 – zakłada się, że przetwarzane rekordy są zbudowane jak następuje:

tekstcz **sep4** tekstw **sep4** tekstkon

Podczas wyprowadzania, w serii rekordów z powtarzającym się ‘tekstcz’, zostawia się tylko pierwszy ‘tekstcz’ a pozostałe pomija oraz pomija się wszystkie ‘tekstkon’ z wyjątkiem ostatniego.

UWAGA: ‘tekstkon’ nie zostanie dopisany po ostatnim rekordzie (na końcu danych). Powinien dopisać go EPILOG.

PRZET6 – zakłada się, że w PROLOGU został zdefiniowany tekst *BLOT₀*. Z posortowanych rekordów usuwa się wszystkie, z wyjątkiem pierwszego i ostatniego, pomiędzy które zostanie wstawiony tekst *BLOT₀*.

PRZET7 – jak *PRZET4* tylko o 1 większa liczba tekstów oddzielonych **sep4**. Zakłada się, że rekordy są zbudowane jak następuje:

tekst1 **sep4** tekst2 **sep4** tekst3 **sep4** tekst4 **sep4** tekst

Podczas wyprowadzania kolejnych rekordów, pomija się powtarzające się ‘tekst1’, ‘tekst2’, ‘tekst3’ lub ‘tekst4’.

Tryby przetwarzania końcowego:

tryb 0 - przy porównywaniu tekstów rozróżnia się wszystkie znaki kodu ASCII (porównywanie dokładne);

tryb 1 - przy porównywaniu tekstów utożsamia się duże i małe litery przyjmując kod MAZOVIA dla liter polskich;

tryb 2 - przy porównywaniu tekstów utożsamia się duże i małe litery przyjmując kod LATIN 2 dla liter polskich;

tryb 16 - jak *tryb 0*, ale przed porównywaniem, teksty zostają znormalizowane tzn. ciągi spacji zostają zamienione na jedną spację a spacje na końcu zostają usunięte;

tryb 17 - jak *tryb 1*, ale przed porównywaniem, teksty zostają znormalizowane tzn. ciągi spacji zostają zamienione na jedną spację a spacje na końcu zostają usunięte;

tryb 18 - jak *tryb 2*, ale przed porównywaniem, teksty zostają znormalizowane tzn. ciągi spacji zostają zamienione na jedną spację a spacje na końcu zostają usunięte.

6.1 Wywołanie programu SKSORT

Program **SKSORT** wywołuje się następująco:

SKSORT wej rob wyj sep1 sep2 sep3 wer kod przet.t sep4 sep5 zlwyst.g sep6

Opis parametrów wywołania:

- wej* — nazwa zbioru wejściowego; gdy nazwę zastąpimy gwiazdką, program utworzy nazwę na podstawie parametrów MAKD zapisanych w zbiorze MAKD.PAR;
- rob* — nazwa zbioru roboczego, w którym SKSORT zapisuje sortowane rekordy z zakodowanymi kluczami;
- wyj* — nazwa zbioru wyjściowego;
- sep1* — wartość dziesiętna znaku sep1 (oddzielającego PROLOG i EPILOG od ciągu rekordów). Wartość domyślna: 232;
- sep2* — wartość dziesiętna znaku sep2 (oddzielającego KLUCZ od TEKSTU w rekordzie). Wartość domyślna: 186;
- sep3* — wartość dziesiętna znaku sep3 (znaczącego koniec rekordu). Wartość domyślna: 187;
- wer* — numer funkcji weryfikującej *WERn*. Wartość domyślna: 0.
- kod* — jeżeli wartością tego parametru jest liczba, to wskazuje ona numer funkcji kodującej klucze *KODn*. W przeciwnym przypadku tekst tego parametru, po dopisaniu rozszerzenia **.KNF**, zostanie potraktowany jako nazwa zbioru zmian w tablicy kodującej (patrz 6.1.1). Zbiór zmian, oprócz samych zmian, określa również funkcję kodującą, która będzie zastosowana. Wartość domyślna: 1;
- przet.t* — numer funkcji (przed kropką) i trybu (po kropce) stosowanych w przetwarzaniu końcowym *PRZETn*. Wartość domyślna: 0.0;
- sep4* — wartość dziesiętna znaku sep4 (separator dodatkowy używany w niektórych funkcjach *PRZETn*). Wartość domyślna: 188;
- sep5* — wartość dziesiętna znaku sep5 (separator dodatkowy używany w niektórych funkcjach *KODn*). Wartość domyślna: 124;
- zlwyst.g* — *zlwyst* – wartość dziesiętna znaku zlwyst (znacznik liczby połączonych rekordów w *PRZET1,2,3*). Wartość domyślna: 185. *g* – ograniczenie liczby łączonych rekordów. Wartość domyślna: 1 000 000;
- sep6* — wartość dziesiętna znaku sep6 (separator tekstu wiodącego). Wartość domyślna: 184.

6.1.1 Zbiór zmian w tablicy kodującej

Program **SKSORT**, w trakcie realizacji kodowania KLUCZY, wykorzystuje tablice kodujące: **tab** (dla funkcji 1, 2, 33 i 34) i **tab1** (dla funkcji 3 i 35). Tablice te, opisane szczegółowo w Dodatku D, można modyfikować określając *zbiór zmian*.

Zbiór zmian jest zbiorem tekstowym złożonym z ciągu linii, z których każda określa pojedynczą *zmianę* w tablicy.

Budowa *zmiany* jest następująca:

k < kod > < ascii > < nwart >

gdzie:

- < kod > – numer funkcji kodującej,
- < ascii > – wartość ASCII znaku (liczba w zapisie dziesiętnym),
- < nwart > – nowa wartość (liczba w zapisie dziesiętnym).

< kod > występujący w ostatniej *zmianie* określa funkcję kodującą, która będzie zastosowana (po zrealizowaniu zmian w tablicy kodującej).

6.2 Przykład formatu 5

Poniższy format służy do przygotowywania danych dla programów **SKSORT** (w pierwszej kolejności) i (w drugiej kolejności) **YEDYT**. Wybiera on z bazy zawartości podpół \$ppole\$ z pola \$pole\$ tworząc dane dla programu **SKSORT**, który je posortuje i przekaże programowi **YEDYT** w celu wydrukowania w dwóch kolumnach. W prezentowanej wersji, wybierane są zawartości podpół 1 pola 100 (w formacie MARC-BN jest to nazwisko autora). Dzięki wyborowi odpowiedniej (*PRZET3*) funkcji przetwarzania końcowego, **SKSORT** połączy jednakowe nazwiska podając w nawiasach liczbę ich wystąpień.

```
1 def $gm$ {EP}100{EQ}83 górny margines
2 def $lm$ 5 lewy margines w znakach
3 def $kol2$ 180 współrzędna x 2 kolumny w 1/60 cala
4 def $pag$ {EP}2200{EQ}1833 pozycja paginy na stronie
5 def $zks$ {EP}2000{EQ}1666 zbliżający kon. str. od góry
6 def $kst$ {EP}2100{EQ}1750 koniec strony od góry
7 def $ods$ {EP}30{EQ}25 odstęp między wiersz.
8 def $dl$ 30 maksymalna długość wiersza (w kolumnie)
9 def $s2$ 75 szerokość tekstu paginy
10 def $styl$ 1 rodzaj druku (1-normalny, 25-gruby, 65-kursywa)
11 def $pole$ 100 nazwa pola
12 def $ppole$ 1 nazwa podpola
13 def $wc1$ 3 wcięcie dla tekstów dłuższych od szer.kolumny
14 $$$
15 <e>1<$lm$><e>x1
16 <sekl><2><0><$kol2$>
17 <sepa><0><$pag$><$s2$><25><224>- % -<224><crlf>
18 <semy><$kst$><$zks$><sevs><$ods$><semg><$gm$><seyy><50>
19 <seac><$s2$><25><2><0cekr><6>tytułu<crlf>
20 <sede><224><225>~<senw><sewc><$wc1$><232><K>
21 <PD><K> Początek dokumentu
22 <Np>$pole$<K> nazwa pola
23 <Npp>$ppole$<K> nazwa podpola
24 <PPj><Dpp> AA<#186><senw><seal><$dl$><$styl$><1>
25 <Dpp> (<#185>)<#188><crlf><#187><K>
26 <PPwp><Dpp> AA<#186><senw><seal><$dl$><$styl$><1>
27 <Dpp> (<#185>)<#188><crlf><#187><K>
28 <PPws><Dpp> AA<#186><senw><seal><$dl$><$styl$><1>
29 <Dpp> (<#185>)<#188><crlf><#187><K>
30 <PPwo><Dpp> AA<#186><senw><seal><$dl$><$styl$><1>
31 <Dpp> (<#185>)<#188><crlf><#187><K>
32 <KD><K> Koniec dokumentu
33 <EP><232><sens><crlf><e>M
```

Komentarz (*grube cyfry oznaczają numery wierszy formatu*):

20: <232> – przyjęta wartość **sep1**. Zaznacza koniec PROLOGu. Drugi raz <232> pojawia się w wierszu

33: wskazując początek EPILOGu.

24, 26, 28, 30: <#186> – oddziela KLUCZ i TEKST w rekordach; 186 jest przyjętą wartością **sep2**.

25, 27, 29, 31: <#187> – zaznacza koniec rekordu; 187 jest przyjętą wartością **sep3**. <#188> – jest to separator **sep4** oddzielający część *hasłową* od części *opisowej* sortowanych rekordów (patrz opis funkcji *PRZET1*, 2 i 3). <#185> – znacznik **zlwyst**, który w trakcie przetwarzania końcowego będzie zastąpiony przez liczbę powtórzeń porównywanego tekstu.

Powyższy format przewiduje zastosowanie trzech programów: **MAKD** — w celu wybrania informacji z bazy i przygotowania danych dla programów **SKSORT** i **YEDYT**, programu **SKSORT** — w celu posortowania danych i programu **YEDYT**, który sformuje dane w taki sposób, aby powstał tekst dwukolumnowy i wysłał go na drukarkę igłową. W celu skoordynowania pracy tych trzech programów wskazane jest ułożyć procedurę-bat, która może mieć następującą postać:

```
1   MAKD nazwa-bazy 0 0 2 T 1 PP5 1 * N 0 0 PRZYKŁAD_5
2   IF errorlevel 1 GOTO KONIEC
3   SKSORT * RRR.PSO RRR.SOR 232 186 187 0 1 3
4   IF errorlevel 1 GOTO KONIEC
5   DEL RRR.PSO
6   YEDYT RRR.SOR 1 1 1 0 0 1 0 0 PRZYKŁAD_5
7   :KONIEC
```

Powyżej założyliśmy, że opisany format został wprowadzony do zbioru OPXXPP5.SFO i przetworzony na postać wynikową za pomocą programu **TFORM** (TFORM OPXXPP5 EP).

7 Programy pomocnicze

7.1 Program ZAMIANY

Program ZAMIANY wykonuje zamiany tekstów w dowolnym zbiorze tekstowym. Oddzielny zbiór zawiera *listę zamian*, według której wykonywane są zamiany.

Lista zmian składa się z: *wiersza separatorów* i ciągu wierszy opisujących poszczególne *zamiany*.

Wiersz separatorów jest ciągiem 13 znaków:

xypqnyyytsruk

i służy do określenia:

- separatorów **x** i **y** używanych w opisie tzw. *zamian strukturalnych*;
- separatora **p** używanego dla *zamian z podziałem*;
- separatora **q** używanego dla *zamian wierszowych*;
- separatora **n** używanego dla *zamian kontekstowych*;
- separatorów **t** i **s** używanych w opisie tzw. *zamian ograniczonych*;
- znacznika **r** wskazującego nowe podziały linii tekstu. Jeżeli wartością znacznika **r** jest litera 'N', to linie tekstu nie będą dzielone; w przeciwnym przypadku w czasie wyprowadzania każdej linii będzie ona dzielona w miejscach występowania znacznika **r** (znacznik **r** będzie zastępowany przez *nową linię*);
- znacznika **u** mówiącego czy usuwać puste wiersze (**u** = 'U' – usuwać) domyślnie 'N' (nie usuwać);
- separatora **k** używanego dla *zamian eliminujących*.

Najczęściej przyjmuje się wartości **x**='|', **y**='#', **p**='`', **q**='%', **n**='+', **t**='\$', **s** = <186> i **k**=';'.
Domyślną wartością znaczników **r** i **u** jest 'N', a pozostałych separatorów – '#'.
Opisy zamian występujące w kolejnych wierszach mogą być następujących typów: *zwykłe, strukturalne, z podziałem, wierszowe, kontekstowe, ograniczone i eliminujące*.

- Zamiany *zwykłe* mają budowę następującą:

ztekstsztekstnz

gdzie: *teksts* – tekst stary (zamieniany)

tekstn – tekst nowy

z – dowolny znak nie występujący ani w *teksts*, ani w *tekstn* oraz różny od **x**, **p**, **q**, **n**, **t**,
i **k** określonych w wierszu separatorów

W wyniku realizacji zamiany zwykłej wszystkie wystąpienia *teksts* w całym zbiorze zostaną zastąpione przez *tekstn*.

- Zamiany *strukturalne* mają budowę następującą:

x tekstac xtektb(y)x

gdzie: *teksta* – tekst zamieniany

c – znak występujący bezpośrednio po *teksta*

tektb(y) – struktura tekstu nowego

x – dowolny znak nie występujący ani w *teksta* ani w *tektb* oraz różny od **y**,

y – znak, którego wystąpienia w *tektb* będą zastąpione przez znak **c**.

W wyniku realizacji zamiany strukturalnej wszystkie wystąpienia *tekstac* w całym zbiorze zostaną zastąpione przez *tektb*, w którym uprzednio wszystkie wystąpienia znaku **y** zostaną zastąpione przez znak **c**.

- Zamiany z *podziałem* mają budowę następującą:

$p\text{teksts}p\text{tekstn}p$

gdzie: *teksts* – tekst stary (zamieniany)

tekstn – tekst nowy

p – dowolny znak nie występujący ani w *teksts*, ani w *tekstn* oraz różny od *x*, *q*, *n*, *t*, i *k* określonych w wierszu separatorów

Zamiany z podziałem, oprócz zamiany tekstu starego na nowy, dodają znaki nowej linii (*cr,lf*) przed każdym tekstem nowym. W ten sposób, z jednego wiersza powstaje tyle wierszy nowych ile było zamian, przyczem nowe wiersze będą się zaczynały od tekstu *tekstn*.

- Zamiany *wierszowe* mają budowę następującą:

$q\text{teksts}q\text{tekstn}q$

gdzie: *teksts* – tekst stary (zamieniany)

tekstn – tekst nowy

q – dowolny znak nie występujący ani w *teksts*, ani w *tekstn* oraz różny od *x*, *p*, *n*, *t*, i *k* określonych w wierszu separatorów

Zamiany dokonuje się tylko wtedy gdy *teksts* równa się całemu wierszowi. Jeżeli ponadto, *tekstn* jest pusty – następuje usunięcie (pominięcie) zamienianego wiersza.

- Zamiany *kontekstowe* mają budowę następującą:

$n\text{znakn}\text{tekst}q$

gdzie: *znak* – dowolny znak (zamieniany)

tekst – dowolny tekst

n – dowolny znak nie występujący ani w *teksts*, ani w *tekstn* oraz różny od *x*, *p*, *q*, *t*, i *k* określonych w wierszu separatorów

Jeżeli w wierszu występuje znak *znak* i jest on umieszczony pomiędzy dwoma literami to ten znak zostanie zastąpiony przez tekst *tekst*. Przez literę rozumie się tutaj dużą lub małą literę alfabetu łacińskiego (littery polskie i akcentowane nie traktuje się jako littery).

- Zamiany *ograniczone* mają budowę następującą:

$t\text{teksts}t\text{tekstn}t$

gdzie: *teksts* – tekst stary (zamieniany)

tekstn – tekst nowy

t – znak określony w wierszu separatorów i nie występujący ani w *teksts* ani w *tekstn* oraz różny od *x*, *p*, *q*, *n*, i *k* określonych w wierszu separatorów

W wyniku realizacji zamiany ograniczonej wszystkie wystąpienia *teksts*, w każdym wierszu zbioru, w części poprzedzającej znak *s*, będą zastąpione przez *tekstn*.

- Zamiany *eliminujące* mają budowę następującą:

$k\text{tekst}k$

gdzie: *tekst* – dowolny tekst

k – dowolny znak nie występujący w *tekst*, oraz różny od *x*, *p*, *q*, *n*, i *t* określonych w wierszu separatorów

Jeżeli wiersz zawiera tekst *tekst* to zostaje usunięty (pominięty).

Osobliwości:

W trakcie realizacji zamian, znak o wartości 250 będzie zamieniony na znak o wartości 0.

7.1.1 Wywołanie programu ZAMIANY

Program **ZAMIANY** wywołuje się następująco:

ZAMIANY *wej wyj zam*

Opis parametrów wywołania:

- wej* — nazwa zbioru wejściowego, podlegającego zamianom. Jeżeli nazwę zbioru zastąpimy gwiazdką, program utworzy nazwę na podstawie parametrów MAKD zapisanych w zbiorze MAKD.PAR;
- wyj* — nazwa zbioru wyjściowego, w którym zostanie umieszczony tekst po zamianach. Jeżeli nazwę zbioru zastąpimy gwiazdką, zbiór wyjściowy zostanie umieszczony w zbiorze wejściowym, a więc zbiór wejściowy zostanie zmodyfikowany;
- zam* — nazwa zbioru zawierającego listę zamian. Domyślnym rozszerzeniem tego zbioru jest .CEX.

7.2 Program MAZAM

Program **MAZAM** wykonuje zamiany tekstów w dowolnym zbiorze tekstowym według spisu zamian podanego w specjalnym zbiorze zwanym *listą zamian MAZAM*. W odróżnieniu od opisanego wyżej programu **ZAMIANY**, który dokonuje zamian „wiersz po wierszu” (lub, jak kto woli, „linia po linii”), program **MAZAM** dokonuje zamian „partia po partii”.

Zakłada się, że zbiór wejściowy zawiera powtarzający się tekst zwany *separator partii*.

Partia jest to fragment zbioru wejściowego

- od początku zbioru do pierwszego wystąpienia *separatora partii* wraz z tym separatorem,
- zawarty pomiędzy kolejnymi *separatorami partii* plus separator kończący, lub
- zawarty między ostatnim *separator partii* a końcem zbioru.

Lista zamian MAZAM ma budowę następującą:

wiersz (linia) 1 — zawiera określenie *separatora partii*,
dalsze wiersze (linie) — opisują kolejne *zamiany*.

Zamiany w *liście zamian MAZAM* mają strukturę taką jak zamiany *zwykle* programu **ZAMIANY**, tzn pierwszy znak wiersza (linii) staje się separatorem oddzielającym „tekst stary” od „tekstu nowego” i ograniczającym je z obu stron.

Zarówno *separator partii* jak i tekst (*stary* i *nowy*) w zamianach, mogą zawierać oprócz zwykłych znaków tekstowych również znaki: CR (wartość 13) i LF (wartość 10). Znaki te w liście zamian należy zapisywać, odpowiednio, '\r' i '\n'. Natomiast znak '\' należy zapisywać jako '\\'.

7.2.1 Wywołanie programu MAZAM

Program **MAZAM** wywołuje się następująco:

MAZAM *wej wyj zam*

Opis parametrów wywołania:

- wej* — nazwa zbioru wejściowego, podlegającego zamianom. Jeżeli nazwę zbioru zastąpimy gwiazdką, program utworzy nazwę na podstawie parametrów MAKD zapisanych w zbiorze MAKD.PAR;
- wyj* — nazwa zbioru wyjściowego, w którym zostanie umieszczony tekst po zamianach. Jeżeli nazwę zbioru zastąpimy gwiazdką, zbiór wyjściowy zostanie umieszczony w zbiorze wejściowym, a więc zbiór wejściowy zostanie zmodyfikowany;
- zam* — nazwa zbioru zawierającego *listę zamian MAZAM*. Domyślnym rozszerzeniem tego zbioru jest .CEZ.

7.3 Program TSKR

Program **TSKR** kopiuje tekstowy zbiór wejściowy na zbiór wyjściowy „łamiąc” długie wiersze. Jako długie, uznaje się wiersze, których długość przekracza określoną (przez trzeci parametr programu) liczbę znaków *dlug*. Podczas kopiowania wiersza, po przekroczeniu *dlug* znaków, program „łamie” wiersz na najbliższej spacji tzn. reszta kopiowanego wiersza staje się nowym wierszem, który z kolei może być również „złamany” w sposób analogiczny. Dodatkowo, na początku każdego nowopowstałego, w wyniku złamania, wiersza zostanie dopisany tekst *tłam* podany w czwartym parametrze programu. Domyślnie, tekst *tłam* jest tekstem pustym.

7.3.1 Wywołanie programu TSKR

Program **TSKR** wywołuje się następująco:

TSKR *wej wyj dlug tłam*

Opis parametrów wywołania:

- wej* — nazwa zbioru wejściowego, podlegającego skracaniu wierszy;
- wyj* — nazwa zbioru wyjściowego, w którym zostanie umieszczony tekst ze skróconymi wierszami;
- dlug* — graniczna liczba znaków wiersza, po przekroczeniu której nastąpi łamanie wierszy na spacjach;
- tłam* — tekst umieszczany na początku każdego wiersza będącego kontynuacją wiersza złamanego. Domyślnie, jest to tekst pusty.

7.4 Program REDINDD

Program **REDINDD** służy do grupowania kolejnych numerów w indeksach książek. Dokładniej, wzrastający ciąg numerów, oddzielonych przecinkami lub innymi znakami przestankowymi, zostaje zredukowany, przez wydzielenie podciągów numerów kolejnych, które zastępuje się zapisem n_1-n_k (gdzie: n_1 – pierwszy numer podciągu, n_k – ostatni numer podciągu). Ponadto, jeżeli n_1 i n_k mają tę samą liczbę cyfr wtedy w n_k pomija się pierwsze cyfry, które są identyczne z odpowiednimi cyframi n_1 .

Bardziej precyzyjnie: Program **REDINDD** kopiuje zbiór wejściowy na zbiór wyjściowy aż do napotkania tekstu wskazanego przez trzeci parametr programu. Wtedy, nadal kopując, poszukuje (w tym samym wierszu) pierwszej liczby. Jeżeli natrafi na ciąg liczb oddzielonych przecinkami lub innymi znakami przestankowymi zakończony końcem wiersza (CR,LF) lub końcem wiersza poprzedzonym znakiem ‘;’ (;,CR,LF) to dokonuje opisanej wyżej redukcji numerów. Zredukowany ciąg numerów zostaje wysłany do zbioru wyjściowego. Powyższa procedura powtarzana jest aż do końca zbioru wejściowego.

Redukcja numerów działa prawidłowo pod warunkiem, że zredukowany ciąg numerów znajdują się na końcu pojedynczego wiersza.

7.4.1 Wywołanie programu REDINDD

Program **REDINDD** wywołuje się następująco:

REDINDD *wej wyj tcn*

Opis parametrów wywołania:

- wej* — nazwa zbioru wejściowego;
- wyj* — nazwa zbioru wyjściowego;
- tcn* — tekst poprzedzający ciąg zredukowanych numerów.

7.5 Program STYTABB

Program STYTABB służy do eliminowania powtarzających się tytułów w trzystopniowej hierarchii numerycznej. Jego działanie polega na kopiowaniu zbioru wejściowego na zbiór wyjściowy z modyfikowaniem zlokalizowanych w zbiorze wejściowym *tytułów numerycznych*.

Załóżmy, że *tytuły numeryczne* mają postać:

$$\backslash\text{tyta}\{xyz\}$$

gdzie każda z liter x, y, z reprezentuje 3-cyfrową liczbę powstałą z uzupełnienia zerami od lewej liczb tworzących oznaczenie działu w bibliografii. Np.: dla działu 3.11.2 otrzymamy qyz = 003011002 (x = 003, y = 011, z = 002).

Program STYTABB, najpierw każdy *tytuł numeryczny* $\backslash\text{tyta}\{xyz\}$ zamieni na 3 elementy:

$$\begin{aligned} &\backslash\text{tta}\{x\} \\ &\backslash\text{ttb}\{xy\} \\ &\backslash\text{ttc}\{xyz\} \end{aligned}$$

reprezentujące tytuły różnych poziomów (a, b, c). Następnie, pominięte zostaną: tytuł $\backslash\text{tta}\{x\}$ i/lub tytuł $\backslash\text{ttb}\{xy\}$ jeżeli takie tytuły już wystąpiły wcześniej. Pomijane są również $\backslash\text{ttb}\{xy\}$, gdy y = 000 i $\backslash\text{ttc}\{xyz\}$, gdy z = 000.

7.5.1 Wywołanie programu STYTABB

Program STYTABB wywołuje się następująco:

STYTABB *wej wyj tyn*

Opis parametrów wywołania:

- wej* — nazwa zbioru wejściowego;
- wyj* — nazwa zbioru wyjściowego;
- tyn* — tekst rozpoczynający *tytuł numeryczny* w zbiorze wejściowym (w powyższym opisie programu założyliśmy, że *tyn* = " $\backslash\text{tyta}\{\}$ ").

7.6 Program STATY

Program STATY kopiuje tekstowy zbiór wejściowy na zbiór wyjściowy aż do napotkania w zbiorze wejściowym tzw. *ciągu operacyjnego*. Następnie wykonuje napotkany *ciąg operacyjny* i kontynuuje kopiowanie aż do napotkania kolejnego *ciągu operacyjnego*, który wykonuje, itd... aż do osiągnięcia końca zbioru wejściowego co kończy działanie programu.

Ciąg operacyjny jest tekstem o następującej budowie:

$\langle\text{ico}\rangle$ *ciąg operacji K*

gdzie: $\langle\text{ico}\rangle$ — *identyfikator ciągu operacyjnego*. Jest to dowolny tekst określony przez trzeci (pozycyjny) parametr programu. Wartością domyślną $\langle\text{ico}\rangle$ jest tekst "!!".

Operacje wykonywane w ramach ciągu operacyjnego mogą działać na *rejestrach*. Program STATY udostępnia 99 rejestrów:

$RD_1 \dots RD_{99}$

są to rejestry tzw. *długie*, tzn. zawierające liczby z zakresu $(-2^{31}, +2^{31} - 1)$.

Ciąg operacji może zawierać następujące operacje:

- Z** $\langle n \rangle$ — zeruj rejestr RD_n (wstaw do RD_n liczbę 0)
- A** $\langle n \rangle$ *liczba* — dodaj do zawartości rejestru RD_n liczbę *liczba*
- C** $\langle n \rangle$ *liczba* — odejmij od zawartości rejestru RD_n liczbę *liczba*
- D** $\langle n_1 \rangle \langle n_2 \rangle$ — dodaj zawartość rejestru RD_{n_2} do RD_{n_1} i wynik wstaw do RD_{n_1}
- F** $\langle n_1 \rangle \langle n_2 \rangle$ — od liczby zawartej w rejestrze RD_{n_1} odejmij liczbę z RD_{n_2} i wynik wstaw do RD_{n_1}

M $\langle n1 \rangle \langle n2 \rangle$	— pomnóż zawartości rejestrów RD_{n1} i RD_{n2} i wynik wstaw do RD_{n1}
R $\langle n1 \rangle \langle n2 \rangle$	— przenieś zawartość rejestru RD_{n2} do RD_{n1}
B $\langle n \rangle$	— w miejscu operacji wpisz liczbę będącą zawartością rejestru RD_n , w zapisie dziesiętnym
X $\langle n \rangle$	— w miejscu operacji wpisz liczbę będącą zawartością rejestru RD_n (w zapisie dziesiętnym) oddzielając dwie ostatnie cyfry znakiem przecinka
E $\langle n \rangle$	— w miejscu operacji wpisz liczbę będącą zawartością rejestru RD_n (w zapisie dziesiętnym) oddzielając 6 ostatnich cyfr znakiem przecinka a następnie usuwając 4 ostatnie cyfry (denominacja złotych)
T $\langle d \rangle \langle tekst \rangle$	— w miejscu operacji wpisz tekst <i>tekst</i> o długości d znaków
W $\langle w \rangle \langle r \rangle$	— zbadaj warunek w dotyczący rejestru RD_n – gdy warunek nie jest spełniony, pomiń wszystkie operacje do znaku #. Warunki: $w = 1$ – zawatość $RD_n \leq 0$ $w = 2$ – zawatość $RD_n > 0$ $w = 3$ – zawatość $RD_n < 0$
#	— koniec pomijania dla operacji warunkowych
P $\langle rd \rangle$	— wykonuj ciąg operacji zawarty między P $\langle rd \rangle$ i Y , $\langle rd \rangle$ razy ($\langle rd \rangle$ – zawatość rejestru RD). (Pętla wykonywana $\langle rd \rangle$ razy)
Y	— koniec powtarzanego ciągu dla operacji P $\langle rd \rangle$ (koniec pętli)
K	— zakończ wykonywanie <i>ciągu operacyjnego</i> . Jest to również oznaczenie końca ciągu operacyjnego

Wykonanie *ciągu operacyjnego* polega na wykonaniu zawartych w nim operacji przyczem, sam ciąg operacyjny nie jest kopiowany do zbioru wyjściowego a jedynie w jego miejsce (na wyjściu) zostają wpisane wyniki operacji **B**, **X**, **E** i **T**.

7.6.1 Wywołanie programu STATY

Program **STATY** wywołuje się następująco:

STATY wej wyj ico

Opis parametrów wywołania:

<i>wej</i>	— nazwa zbioru wejściowego;
<i>wyj</i>	— nazwa zbioru wyjściowego;
<i>ico</i>	— identyfikator <i>ciągu operacyjnego</i> – tekst rozpoczynający <i>ciąg operacyjny</i> . Domyślnie jest to tekst "!!-".

7.7 Program ADAPT

Program **ADAPT** przekształca tekst zawierający *sekwencje-ESC* drukarki igłowej EPSON (tryb ESC/P) na inny zapis według wskazanego *dekodera*. Sekwencje-ESC są to specyficzne dla ESC/P ciągi znaków zaczynające się od znaku ESC (wartość 27). Sposób konstruowania 'dekodera' opisany jest poniżej. Dekoder, który steruje pracą programu **ADAPT** określa, jaki znak będzie traktowany jako znak ESC, jak mają być dekodowane inne znaki i na co mają być zamieniane wskazane sekwencje-ESC.

Konstrukcja dekodera składa się z następujących etapów:

- utworzenie zbioru tekstowego zawierającego postać źródłową dekodera. Zbiór taki powinien mieć rozszerzenie .DSO
- przetworzenie postaci źródłowej dekodera na jego postać wynikową (maszynową) za pomocą programu **DFORM**. Zbiór taki będzie miał rozszerzenie .DEK

Budowę postaci źródłowej dekodera wyjaśnimy na przykładzie dekodera przekształcającego tekst przygotowany do wydrukowania na drukarce igłowej EPSON na tekst dla drukarki

pracującej w trybie IBM-Proprinter. Tekst tego dekodera znajduje się na najbliższej stronie. A oto objaśnienia:

Wiersze 1–64 opisują sposób dekodowania pojedynczych znaków. Wiersze te tworzą tablicę zawierającą 256 zapisów postaci <a,b>, które odnoszą się do 256 znaków kodu ASCII. Cyfry umieszczone z lewej i ponad takimi zapisami określają, odpowiednio, pierwszą i drugą cyfrę szesnastkową znaku dekodowanego.

Zapis <0,0> — oznacza: "nie zmieniaj tego znaku"

Zapis <1,a> — oznacza: "zamień znak na znak o wartości a"

Zapis <2,0> — oznacza: "traktuj ten znak jako ESC"

Wiersze 65–80 opisują dekodowanie sekwencji-ESC. Poczynając od wiersza 66 mają one jednakową strukturę: pierwszy element określa długość sekwencji wejściowej, następnie – między spacjami – podana jest sekwencja wejściowa, zapis <fi> określa sposób dekodowania, po nim podana jest długość sekwencji wyjściowej, sekwencja wyjściowa i oznaczenie końca pojedynczej zamiany (<K>). Sposób dekodowania:

Zapis <f1> — zwykła zamiana sekwencji wejściowej na wyjściową;

Zapis <f2> — zamiana sekwencji wejściowej i znaku występującego bezpośrednio za nią na sekwencję wyjściową tak, aby znak występujący po sekwencji wejściowej znalazł się w miejscu wskazanym przez znak '#' w sekwencji wyjściowej. Znak podstawiony za '#' zostaje, za każdym razem, przechowany i jeżeli w kolejnym zapisie <f2>, w sekwencji wyjściowej wystąpi znak '^', to na jego miejsce będzie wstawiony znak ostatnio przechowany (patrz. wiersz 79). Znaki '#' i '^' zostały zdefiniowane w wierszu 65. Można więc je zastąpić przez dowolne inne znaki.

Zapis <f3> — likwidacja sekwencji wejściowej (zamiana na nic).

W celu przetworzenia dekodera zapisanego w zbiorze DEKOD.DSO na postać wynikową należy wykonać:

DFORM DEKOD

W wyniku powstanie zbiór DEKOD.DEK, który może być wykorzystany przez program ADAPT.

7.7.1 Wywołanie programu ADAPT

Program ADAPT wywołuje się następująco:

ADAPT *wej dek uru*

Opis parametrów wywołania:

- wej* — nazwa zbioru wejściowego, podlegającego zamianom. Jeżeli nazwę zbioru zastąpimy gwiazdką, program utworzy nazwę na podstawie parametrów MAKD zapisanych w zbiorze MAKD.PAR;
- dek* — nazwa zbioru zawierającego dekodery (musi posiadać rozszerzenie .DEK);
- uru* — przyjmuje wartości:
 - 1 – uruchamianie: zamiast na drukarkę, wysyłaj zdekodowany tekst do zbioru ADA.LOG
 - 0 – normalna praca: zdekodowany tekst jest drukowany na drukarce.


```

65 <DEK>#~
66 <2> <e>0 <f1><5> <e>A<9><e>2 <K>
67 <2> <e>M <f1><2> <e>: <K>
68 <3> <e>x<1> <f1><3> <e>I<3> <K>
69 <3> <e>x<0> <f1><3> <e>I<1> <K>
70 <3> <e>!<1> <f1><7> <18><e>F<e>H<e>: <K>
71 <3> <e>!<65> <f1><7> <18><e>F<e>H<e>: <K>
72 <3> <e>!<0> <f1><5> <18><e>F<e>H <K>
73 <3> <e>!<5> <f1><8> <e>F<e>H<e>:<18><15> <K>
74 <3> <e>!<4> <f1><6> <e>F<e>H<18><15> <K>
75 <3> <e>!<24> <f1><5> <18><e>E<e>G <K>
76 <3> <e>!<25> <f1><7> <18><e>:<e>E<e>G <K>
77 <3> <e>!<29> <f1><8> <e>:<e>E<e>G<18><15> <K>
78 <2> <e>l# <f2><4> <e>X#<136> <K>
79 <2> <e>Q# <f2><4>X#~ <K>
80 <2> <e>$ <f3><0> <K>
81 <KON>

```

A Wykorzystanie znaków ASCII > 127

Tablica wykorzystania znaków > 127

	8	9	A	B	C	D	E	F
0	(PPwp)	Ę	Ź	mlf			(Ó)	el
1	Np	ę	ż	msp				Pbr
2	PjI	ł	ó (ó)	fi				PPbr
3	PwPI	PjK	Ó	mcr			(Ń)	
4	PwSI	PwpK	ń (Ą)				(ń)	
5	PwoI	Ć	Ń (ą)					Pob
6	ą (ć)	PwoK	ź					<i>sgnk</i>
7	PPj	Npp (Ś)	ż					<i>sgps</i>
8	PPwp (ł)	Ś (ś)	PD (Ę)					<i>sgde</i>
9	PPws	bf K	PwsK (ę)					<i>sgko</i>
A	PPwo	bf KD	Bpp					<i>sgze</i>
B	Po	bf Dpp	(ź)					<i>sgcr</i>
C	PPo	Ł	(Npp)					<i>sgkg</i>
D	ć (Ź)	O... (Ł)	(O...)	(Ż)			<i>sees</i>	<i>sgpg</i>
E	EP	ś	(PD)	(ż)			<i>sgtd</i>	<i>sgpd</i>
F	Ą (Ć)	sep	(PwsK)				esc	

W nawiasach okrągłych — wersja LATIN 2

Kody instrukcji **EDYTO**, **TABLI** i **YEDYT**

	2	3	4	5
0	<i>seal</i>	<i>senr</i>		<i>stk</i>
1	<i>seac</i>	<i>semg</i>		<i>stkk</i>
2	<i>seap</i>	<i>senw</i>		<i>stgp</i>
3	<i>seao</i>	<i>sere</i>		<i>stdp</i>
4	<i>sein</i>	<i>sepo</i>		<i>stst</i>
5	<i>sede</i>	<i>seko</i>		<i>stds</i>
6	<i>senl</i>	<i>sear</i>		<i>stde</i>
7	<i>sewc</i>	<i>sekl</i>		<i>stre</i>
8	<i>seos</i>	<i>senk</i>		<i>stty</i>
9	<i>seyy</i>	<i>sera</i>		<i>strd</i>
A	<i>sexx</i>	<i>sead</i>		<i>star</i>
B	<i>seff</i>	<i>serd</i>		<i>stza</i>
C	<i>seyp</i>	<i>seda</i>		<i>stbz</i>
D	<i>sepa</i>	<i>sepu</i>		
E	<i>semy</i>	<i>sedb</i>		
F	<i>sens</i>			

B Procedury drukowania (bat'y)

Poniższe procedury drukowania (bat'y) wraz z formatami można traktować jako przykłady zastosowania programów EDYTO, TABLI, SEGKK, YEDYT, SKSORT i ZMIANY. Procedury te, jak i związane z nimi formaty drukowania zawarte są w standardowym zestawie pakietu MAK.

KKA(KKB) – Druk kart katalogowych (2 st. szczeg.)

Typ bazy	– PB(MARCBN)
Pola/podpola	– większość (2 stopień szczegółowości)
bat	– KKA lub KKB (różnią się sposobem drukowania sygnatur)
Programy	– MAKD, EDYTO, SEGKK
Format	– WWK0 (WWK1)
Drukarka	– Epson

ODO(ODOS) – Odsyłacze osobowe bez(z) sygnatur(ami)

Typ bazy	– PB(MARCBN)
Pola/podpola	– większość (odsyłacze od pól 700, 710, 720, 2-gi i 3-ci autor)
bat	– ODO (bez sygnatur) lub ODOS (z sygnaturami)
Programy	– MAKD, EDYTO, SEGKK
Format	– WWO2(WWO3)
Drukarka	– Epson

KTA – Karty tytułowe

Typ bazy	– PB(MARCBN)
Pola/podpola	– większość
bat	– KTA
Programy	– MAKD, EDYTO, SEGKK
Format	– WWT1
Drukarka	– Epson

BIBLEP – Bibliografia (bez sortowania)

Typ bazy	– PB(MARCBN) lub pochodne
Pola/podpola	– 100/1, 2, 3, 4, 5, 6; 110/1, 2, 6, 7; 120/1, 2, 3, 4, 5, 6, 8; 130/1, 3; 200/a, e; 210/a, c, d; 205/a; 215/a;
bat	– BIBLEP
Programy	– MAKD, YEDYT
Format	– RB0
Drukarka	– Epson

BIBLEPS – Bibliografia (z sortowaniem)

Typ bazy	– PB(MARCBN) lub pochodne
Pola/podpola	– 100/1, 2, 3, 4, 5, 6; 110/1, 2, 6, 7; 120/1, 2, 3, 4, 5, 6, 8; 130/1, 3; 200/a, e; 210/a, c, d; 205/a; 215/a;
bat	– BIBLEPS
Programy	– MAKD, SKSORT, YEDYT
Format	– BI1
Drukarka	– Epson, Epson LQ

WYKAZ – Bibliografia w postaci tabelki)

Typ bazy	– PB(MARCBN) lub pochodne
Pola/podpola	– 100/1, 2; 200/a, e; 210/a, c, d; 205/a; 001/n, r;
bat	– WYKAZ
Programy	– MAKD, SKSORT, TABLI
Format	– BI3
Drukarka	– Epson

ZAWBAZ – Wydruk zawartości dokumentów bazy

Typ bazy	– dowolny
Pola/podpola	– wszystkie
bat	– ZAWBAZ
Programy	– MAKD, YEDYT
Format	– GG0
Drukarka	– Epson, Epson LQ

C Formaty dla indeksów

Poniżej podana jest charakterystyka ogólnodostępnych formatów sterujących przygotowaniem różnego rodzaju indeksów. Aby wydrukować odpowiedni indeks należy wywołać wskazany 'bat' i po uzyskaniu menu programu MAKD wpisać w okienko 'format' nazwę wybranego formatu. Dalej, postępujemy wg zasad ogólnych.

IN70 – Indeks haseł przedmiotowych PB

Struktura wydruku – hasło przedmiotowe – lista numerów systemowych; 1 kolumna
Typ bazy – PB
Pola/podpola – 600/a, b, c, d, s, w, t, r, e, f, g, h, i, k
bat – IND60
Drukarka – Epson
Parametry SKSORT – 232, 186, 187, 0, 1, 1

IN71 – Indeks wydawców

Struktura wydruku – nazwa wydawcy – lista numerów systemowych; 1 kolumna
Typ bazy – PB, FIDES itp.
Pola/podpola – 210/c
bat – IND60
Drukarka – Epson
Parametry SKSORT – 232, 186, 187, 0, 1, 1

IN72 – Indeks serii/podserii/tytułów wydawnictw wielotomowych

Struktura wydruku – tytuł serii – lista numerów systemowych; tytuł wyd. wielotomowego – lista nr-ów syst.; tytuł podserii – lista nr-ów syst.; 1 kolumna
Typ bazy – PB, FIDES
Pola/podpola – 225/a, c; 226/a; 228/a, c
bat – IND60
Drukarka – Epson
Parametry SKSORT – 232, 186, 187, 0, 1, 1

IN73 – Indeks tytułowy

Struktura wydruku – tytuł – lista numerów systemowych; 1 kolumna
Typ bazy – PB, FIDES itp.
Pola/podpola – 130/1; 200/a; 224/a
bat – IND60
Drukarka – Epson
Parametry SKSORT – 232, 186, 187, 0, 1, 1

IN74 – Indeks autorów korporatywnych

Struktura wydruku – autor korp. – lista numerów systemowych; 1 kolumna
Typ bazy – PB, FIDES itp.
Pola/podpola – 110/1; 120/1, 3
bat – IND60
Drukarka – Epson
Parametry SKSORT – 232, 186, 187, 0, 1, 1

IN75 – Indeks autorów

Struktura wydruku – autor – lista numerów systemowych; 1 kolumna
Typ bazy – PB, FIDES itp.
Pola/podpola – 100/1, 2, 3, 4, 5, 6
bat – IND60
Drukarka – Epson
Parametry SKSORT – 232, 186, 187, 0, 1, 1

IN76 – Indeks słów kluczowych (z liczbą wystąpień)

Struktura wydruku – słowo kluczowe – lista numerów systemowych; 1 kolumna
Typ bazy – FIDES
Pola/podpola – 610/x
bat – IND60
Drukarka – Epson
Parametry SKSORT – 232, 186, 187, 0, 1, 1

IN77 – Indeks słów kluczowych (z liczbą wystąpień)

Struktura wydruku	– słowa kluczowe w 3 kolumnach
Typ bazy	– FIDES
Pola/podpola	– 610/x
bat	– IND80
Drukarka	– Epson
Parametry SKSORT	– 232, 186, 187, 0, 1, 3

IN50 – Indeks haseł przedmiotowych z odniesieniem do tytułów i

bbn

Struktura wydruku	– hasło przedmiotowe – lista książek (tytuł, bbn); 1 kolumna
Typ bazy	– PB
Pola/podpola	– 600/a, b, c, d, s, w, t, r, e, f, g, h, i, k; 200/a; 001/n;
bat	– IND70
Drukarka	– Epson
Parametry SKSORT	– 232, 186, 187, 0, 2, 2, 188, 124

IN51 – Indeks haseł przedmiotowych z odniesieniem do autorów, tytułów i bbn

Struktura wydruku	– hasło przedmiotowe – lista książek (autor, tytuł, bbn); 1 kolumna
Typ bazy	– PB
Pola/podpola	– 600/a, b, c, d, s, w, t, r, e, f, g, h, i, k; 100/1, 2; 200/a; 001/n;
bat	– IND70
Drukarka	– Epson
Parametry SKSORT	– 232, 186, 187, 0, 2, 2, 188, 124

IN52 – Indeks słów kluczowych z odniesieniem do autorów, tytułów i sygnatury

Struktura wydruku	– słowo kluczowe – lista książek (autor, tytuł, sygnatura); 1 kolumna
Typ bazy	– FIDES
Pola/podpola	– 610/x; 100/1, 2; 200/a; 993/s;
bat	– IND70
Drukarka	– Epson
Parametry SKSORT	– 232, 186, 187, 0, 2, 2, 188, 124

IN53 – Indeks słów kluczowych z odniesieniem do autorów, tytułów i sygnatury (tytuł skrócony)

- Struktura wydruku** – słowo kluczowe – lista książek (autor, tytuł, sygnatura); 1 kolumna; tytuł skrócony
- Typ bazy** – FIDES
- Pola/podpola** – 610/x; 100/1, 2; 200/a; 993/s;
- bat** – IND70
- Drukarka** – Epson
- Parametry SKSORT** – 232, 186, 187, 0, 2, 2, 188, 124

D Tablice kodujące programu SKSORT

Tablica kodująca *tab*
używana w funkcjach kodujących 1, 2, 33 i 34
w nawiasach podano modyfikację dla funkcji 33 i 34

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	64	0	32	0	32	6	11	53	0	0	0	0(31)	0
1	82	0	0	65	2	34	2	34	42	11	54	0	0	0	0	0
2	79	0	81	66	4	36	4	36	9	25	31(31)	0	0	0	0	0
3	0	0	0	67	6	38	6	38	2	30	31	0	0	0	0(29)	0
4	0	0	0	68	8	40	8	40	2	30	29(3)	0	0	0	0(29)	0
5	0	0	0	69	9	42	9	42	2	7	29(3)	0	0	0	0	0
6	0	0	0	70	12	44	12	44	3(7)	42	53	0	0	0	0	0
7	0	0	81	71	14	46	14	46	6	42(39)	54	0	0	0	0	0
8	0	0	81	72	16	48	16	48	9(25)	39(39)	0(11)	0	0	0	0	0
9	0	0	81	73	18	50	18	50	9	30	0(11)	0	0	0	0	0
A	0	0	0	0	20	52	20	52	9	42	0	0	0	0	0	0
B	0	0	0	0	22	81	22	81	18	0	0(53)	0	0	0	0	0
C	0	0	0	0	24	80	24	0	18	25	0	0	0	0	0	0
D	0	0	0	0	26	81	26	81	7(53)	0(25)	0	0(54)	0	0	0	0
E	0	0	0	0	28	0	28	80	2	39	0	0(54)	0	0	0	0
F	0	0	0	0	30	0	30	0	3(7)	0	0	0	0	0	0	0

Wartości wyróżnione:

0 — spacja,

80 — znak pomijany wraz z następującym po nim,

81 — znak pomijany, 82 — mniejszy od spacji.

Tablica kodująca *tab1*
używana w funkcjach kodujących 3 i 35
w nawiasach podano modyfikację dla funkcji 35

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	1	102	17	74	23	75	38	46	98	0	0	0	0(72)	0
1	122	0	2	103	32	76	33	77	87	47	100	0	0	0	0	0
2	119	0	3	104	36	78	37	79	45	63	73(73)	0	0	0	0	0
3	0	0	4	105	38	80	39	81	33	71	72	0	0	0	0(68)	0
4	0	0	5	106	42	84	43	85	33	71	69(34)	0	0	0	0(69)	0
5	0	0	6	107	44	86	45	87	33	40	68(35)	0	0	0	0	0
6	0	0	7	108	48	88	49	89	35(41)	87	99	0	0	0	0	0
7	0	0	8	109	50	90	51	91	39	87(82)	101	0	0	0	0	0
8	0	0	9	110	52	92	53	93	45(63)	82(83)	0(46)	0	0	0	0	0
9	0	0	10	111	54	94	55	95	45	70	0(47)	0	0	0	0	0
A	0	0	11	0	56	96	57	97	45	86	0	0	0	0	0	0
B	0	0	12	0	58	18	59	24	55	0	0(99)	0	0	0	0	0
C	0	0	13	0	60	19	61	25	55	62	0	0	0	0	0	0
D	0	0	14	0	64	20	65	26	41(98)	0(62)	0	0(100)	0	0	0	0
E	0	0	15	0	66	21	67	27	32	83	0	0(101)	0	0	0	0
F	0	0	16	0	70	22	71	28	34(40)	0	0	0	0	0	0	0

Wartości wyróżnione:

0 — spacja,

120 — znak pomijany wraz z następującym po nim,

121 — znak pomijany, 122 — mniejszy od spacji.

Uwaga: w obu tablicach, modyfikacja przy przejściu na LATIN II (wartości w nawiasach) dokonywana jest po wykonaniu ewentualnych modyfikacji użytkownika. W przypadku kolizji, wartością końcową będzie zatem wartość podana w nawiasach.