90

TOS: A TEXT ORGANIZING SYSTEM

Kemal Koymen

Moore School of Electrical Engineering,* University of Pennsylvania, Philadelphia, Pennsylvania 19174.

SUMMARY

This paper reports research undertaken to conceptualize, design and implement a system for automatic indexing, classification and repositing of text items, which may be any aggregates of information in English language on a computer - readable media, in a standard format.

The ultimate goal of the research reported here is to devise all automatic processes which would read text items, and then index, classify and reposit them for subsequent search and retrieval. Only portions of the path to this goal have been made fully automatic. These portions consist of all automatic processes as follows:

1. Scanning the text items and assigning candidate index terms (words or phrases) to the items.
2. Discriminating and rejecting candidate index terms determined to be ineffective in forming a classification automatically.
3. Generating a classification system and repositing the text items in accordance with this system.

To complete the process, some degree of user involvement, on an interactive basis, is incorporated in the system, particularly for

*The author is currently an assistant professor at the Department of Mathematics and Statistics, and, Computer Science, American University, Washington, D.C. 20016.

The reported research was supported under contract N0014-67-A-0216-0007 from the Information Systems Program, Office of Naval Research.

discriminating the index terms which do not contribute to a satisfactory classification. Based on various reports derived automatically, the user can guide the system to systematically search for terms which are not helpful for and even hamper the subsequent classification and information retrieval, until the performance of the system is judged to be adequate.

The specific achievements of the reported research are stated below,

1. System interactiveness
2. Automatic index phrase recognition
3. Summary report, informing the user of the impact of user elected decisions to delete terms on a mass basis and advising him of percentages of reduction in index term vocabulary size or average number of index terms per item resulting from such mass term deletions.
4. Affinity dictionary, giving the user the ability to locate synonymous or near synonymous index terms.
5. Use of classification processes in discriminating unsuitable index terms.
6. An integrated automatic indexing and classification system.
7. Successful automatic indexing and classification of a textual data-base.

The system has been adequately documented (including a user guide) and tested for its reliability and dependability.

The research was conducted in the Moore School of Electrical Engineering, University of Pennsylvania and utilized the UNIVAC Spectra 70/46 computer, operating with the Univac VMOS and DMS. The system has been implemented in Univac version of FORTRAN IV.

INTRODUCTION

This paper reports research undertaken to conceptualize, design and implement a system for automatic indexing, classification and repositing of text items, which may be any aggregates of information in English Language on a computer - readable media, in a standard format.

The paper gives concise description of the processes making up the system in the following sections. Detail description of the system is given elsewhere (2). Two appendixes are provided with this paper. Appendix A contains a glossary of terms used in this paper. The reader is advised to refer to the glossary for those terms whose meanings are not clear enough for him. Appendix B contains a detail description of the classification algorithm used in the system.

The ultimate goal of the research on automatic indexing and classification is to devise all automatic processes which would read text items, and then index, classify and reposit them for subsequent search and retrieval. The research reported here realized a great portion of these automatic processes as shown below:

1. Scanning the text items and assigning candidate index terms (words or phrases)
2. Finding and rejecting candidate index terms determined to be ineffective in forming a classification automatically.
3. Generating a classification system and repositing the text items in accordance with this system.

To complete the process, some degree of user involvement, on

an interactive basis, is incorporated in the system particularly for discriminating the index terms which do not contribute to a satisfactory classification. Based on various reports derived automatically, the user can guide the system to systematically search for terms which are not helpful for and even hamper the subsequent classification and information retrieval, until the performance of the system is judged to be adequate.

The specific achievements of the reported research are stated below.

1. System Interactiveness

   All individual functions constituting the system can be executed on an on-line time sharing basis from a terminal. The operations of functions are controlled by statements in a specified language. The user-system interaction is accomplished by system prompts and answers provided by the user.

2. Automatic Index Phrase Recognition

   The system is capable of automatically recognizing standard and user specified phrases. The system is also able to automatically recognize candidate index phrases which are sequences of words separated by blanks and set off by "stop list" words or "special characters" on either side. If the user elects, low-frequency phrases may also be decomposed into sub-phrases which occur more frequently.

3. Summary report

   This report informs the user of the impact of user elected decisions to delete candidate index terms on a mass basis and advises him of percentages of reduction in index term vocabulary size

or average number of index terms per item resulting from such mass term deletion decisions.

4. Affinity Dictionary

   This dictionary gives the user the ability to locate synonymous or near synonymous index terms. It is believed that this is the first instant of automatic synonym and affinity finding.

5. Use of classification processes in discriminating unsuitable index terms.

6. An integrated automatic indexing and classification system.

7. Successful automatic indexing and classification of a textual data-base. A data-base of 425 text items in world affairs taken from issues of Times magazine published in 1963 has been processed and is used here to illustrate the functions of the system.

The system consists of two main components-the indexing system and the classification system, and operates in an on-line interactive manner. Figure 1 shows gross information flow in the system.

The input to the system is the so-called "Standard Formatted Text File". The original collection of text items must be placed on a computer-readable media in a format acceptable to the system. This format is referred to as Standard Format and the storage media of text items is referred to as Standard Formatted Text File.

Since all collections of text items are somewhat unique, it is the user's or the programmer's responsibility to write a computer program required to place his collection of text items into the Standard Formatted Text File.

The text items in the user's original collection may consist of titles, abstracts, full texts, index terms, or any combination of these. If the collection of text items has already been indexed, then the user needs only to place the index terms on the Standard Formatted Text File (for subsequent automatic classification of text items on the basis of index terms assigned to the text items); otherwise he may place the full texts, abstracts, titles or any combination of these on the Standard Formatted Text File.

The final products, or outputs, of the system are four directories and the data-base, rearranged in accordance with classification numbers assigned to the items automatically.

The rearrangement of text items is achieved through an automatic classification process. The main objective of this process is to group alike text items together into cells or near each other to facilitate searching, browsing and retrieval of text items at a later point in time. A cell is similar to a shelf in a library, where a set of similar objects are stored. In a collection of text items, which are indexed with index terms, the quantitative measure of the likeness of text items is relative,

and measured by the number of index terms common to two text items. These measures of likeness are compared to determine the most "alike" pair. The weights of all index terms are considered to be the same. Assigning of different weights is feasible, but has not been attempted here.

The "cells" are generated on the basis of index terms assigned to each text item (see Appendix B). The algorithm does not require any a-priori cells as a starting point, and forms a hierarchy by successively sub-dividing the collection of item surrogates into non-overlapping groups of text items until approximately signed cells are generated. Subsequently, within each cell is generated a complete set of index terms by forming the union of index terms used to index the respective text items. Then, a hierarchy of index terms is formed by intersecting these inclusive-overlapping sets of index terms and assigning the resulting index terms to the next level up the hierarchy, and in turn deleting these resulting index terms from the original sets. The resultant tree is referred to as the Hierarchical Classification Tree for the data-base. This tree respresents the rearranged data-base, or the so-called Classified Data-base. Figure 3 illustrates a sub-tree of the Hierarchical Classification Tree produced for the illustrative data-base of 425 text items.

Two properties of the Hierarchical Classification Tree in regard to searching and browsing through the Classified Data-base are worth to mention. First, the set of index terms assigned to a given item is contained in the set of index terms, which is made up of the union of index terms of the nodes in the direct path from the root node to the terminal node (cell) which contains the item. For instance, the index terms assigned to item 92 at node 1.1.3.3.2.1 are included in the set of index terms; actress, british press, London, Christine, Keeler, Britain, labor, Macmillan, Minister, Profumo, and so on. Second, each index term appears at most once in any path from the

root node to a terminal node, and the same index term may appear at more than one node. (the number of nodes at which the index term occurs is called the node frequency of the index term).

Node-to-key and key-to-node Directories are generated from the Hierarchical Classification Tree. These directories facilitate searching and browsing through the Classified Data-base. The Key-to-node Directory gives for each key (index term) a list of classification numbers assigned to the nodes that share the key. Vice-versa, the Node-to-key Directory gives the same information in an inverted manner, that is, it gives all the corresponding keys for each classification number (node). Tables 4 and 5 show portions of these directories for the illustrative data-base.

Finally, the Directory of Index Terms contains the set of index terms ordered alphabetically. In this directory, with each index term is associated the respective text item and sentence identification numbers. Table 9 shows a portion of this directory for the illustrative data-base. This table indicates that the index term "ABBE" has been assigned to text items 319 and 163. Furthermore, it also indicates that this term occurs in sentence 1 of the item 319, and in sentences 7 and 27 of the item 163.

The overall process is monitored by the user, who can receive reports and oversee the progress, through his use of time-sharing terminal. The major interactions of the user are indicated in braces in figure 1. Each box, in figure 1, represents a gross process. Table 1 lists for each gross process system functions which constitute the process. For those system functions requiring user interactions, the user actions are also indicated. For instance, the first of the system functions making up the gross process SCAN requires one mandatory and two optional user actions as indicated in table 1. Note that,

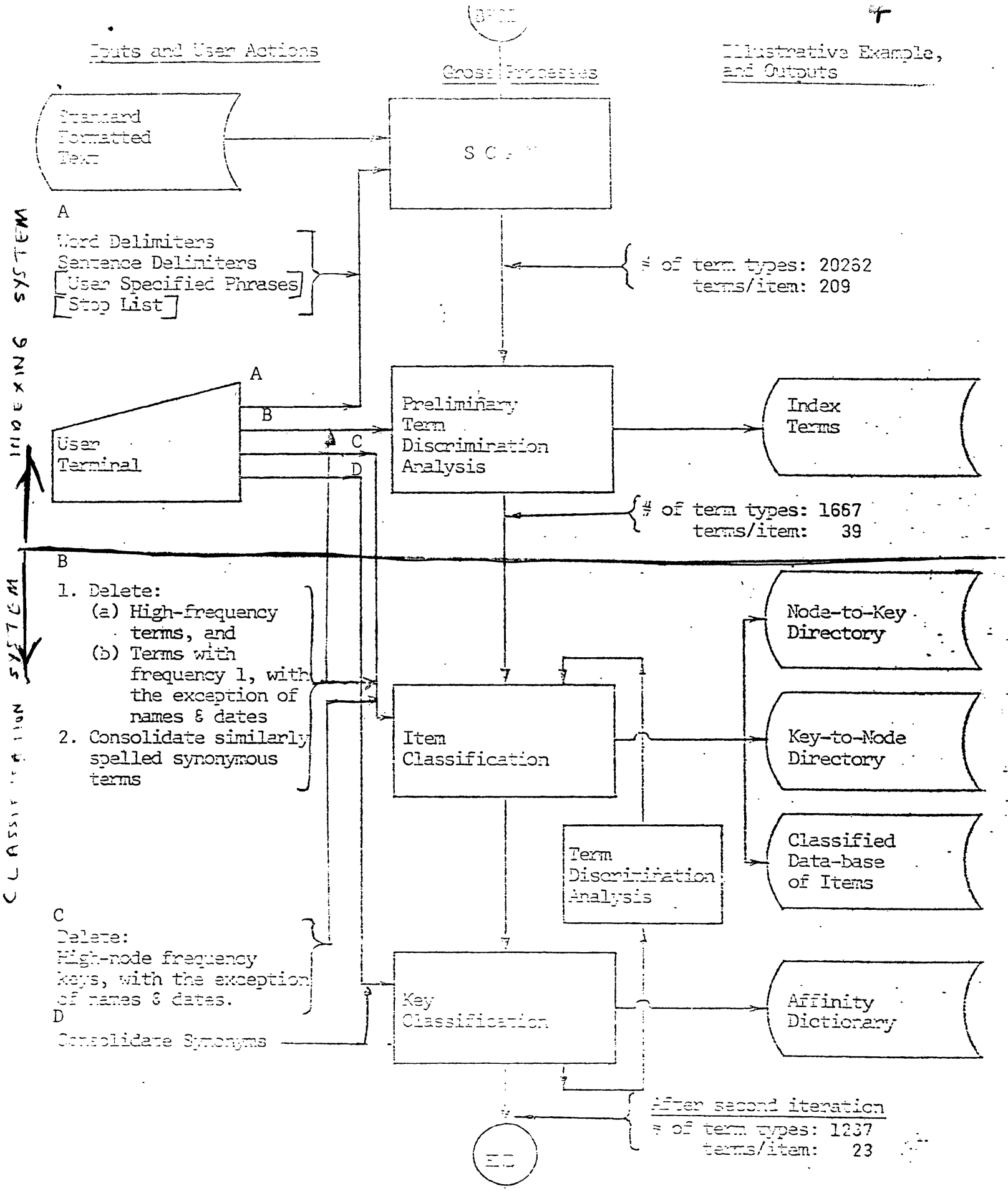


Figure 1

[illegible] Actions, Gross Processes, Inputs and Outputs in the Indexing and Classification [illegible] Text Items.

for each user interaction from the terminal illustrated in figure 1, there is a user action described in table 1. References to figure 1 and table 1 should be made as needed in reading the descriptions of these functions below.

It should be noted that the indexing and classification rely exclusively on the content of text items. Hence, the term a-posteriori is attributed to the system. The term a-posteriori is used adjectively, of knowledge or cognition originating entirely based on experience (examination of text items). In the a-posteriori indexing, none of the information needed for the indexing is available separately or independently of the text items being indexed. Similarly, in the a-posteriori classification, none of the information needed for the classification is available separately or independently of the objects being classified.

There exist several applications for the indexing and classification system reported here. An interesting and challenging application involves the use of the system in a Learning System. A-posteriori analysis, indexing and classification of text support subsequent "learning" of text. The "learning" process consists of the building up of a vocabulary and the categorization of textually expressed knowledge. The system can start learning from scratch, without any prerequisites of prior knowledge, and expand its knowledge by analysis, indexing and classification of subsequent text. Finally, as suggested by Sokal (7), in such a system, classifications can be exploited to explore basic principles of the facts and objects, which can then be used as the basis for prediction of future events.

A second major application for the indexing and classification system consists of the automation of legal search. The system can be exploited to

STEP NAME	GROSS PROCESS	SYSTEM FUNCTIONS	USER ACTIONS
INDEXING	SCAN	Extraction of words and phrases: Automatic phrase generation Refinement of phrases Merge of word and phrase tokens Production of in-item term types	1. Enter word and sentence delimiters. [2.] Enter user specified phrases [3.] Enter stop list.
	PRELIMINARY TERM DISCRIMINATION ANALYSIS	Finding and rejecting the terms which do not contribute to a satisfactory classification. Consolidation of similarly spelled synonymous index terms.	1. Examine the Summary Report, and Delete: (a) High-frequency terms, and (b) Terms with frequency = 1, with the exception of names and dates. 2. Examine the Report of similarly spelled terms and consolidate similarly spelled synonyms.
CLASSIFICATION	ITEM CLASSIFICATION	Generation of item surrogates Item classification Hierarchical Classification Tree Node-to-key and Key-to-node Directories Classified Data-base of items	
	KEY CLASSIFICATION	Generation of key surrogates Key classification Affinity Dictionary	
	TERM DISCRIMINATION ANALYSIS	Finding and rejecting the terms which do not contribute to a satisfactory classification. Finding and consolidating synonymous terms.	Examine the report containing the entire set of keys ordered by node frequencies, and delete high-node frequency ($\geq$ 10) keys, with the exception of names and dates. Examine the Affinity Dictionary, locate and consolidate synonyms.

Table 1

Steps In The Indexing and Classification of Text Items

alleviate the search problems arising out of the staggering growth of legal information and the inadequacy of current search techniques. The system enables the user to perform searches, on an online interactive basis, rapidly and efficiently independent of the size of the data-base. The hierarchical classification system gives the user the ability to narrow or broaden his area of interest by reformulating his search requests.

A third major application involves automating personal libraries (5). Many people such as doctors, authors, researchers, etc., have personal libraries of one sort or another. The individual can store his text items in accordance with an automatically derived classification system, and search for them at a subsequent time using index terms and classification numbers assigned to the text items automatically.

Other possible applications involve the automatic indexing and classification services for large organizations such as corporations and government agencies. Another interesting application is to provide a nation-wide search service for public libraries. Finally, the system can also be used in the production of back-of-the-book indexes. In such an index (i.e. Directory of Intex Terms in figure 1) with each index term is associated the page numbers and optionally the sentence numbers in respective pages.

As indicated in table 1, the entire process is divided into two parts: indexing and classification. Brief descriptions of these processes are given below.

INDEXING

Indexing is the assignment of one or more index terms (names concepts, descriptors, affiliations, words, phrases, etc.) to each text item. The processes of indexing extract and evaluate candidate index terms from the text items. Unsuitable candidate index terms are rejected from further candidacy through a term discrimination analysis, which consists of the process of finding and rejecting index terms which do not contribute to a satisfactory classification. Ambiguous or vague terms are systematically investigated, and as a result, these terms are consolidated, expanded or dropped altogether. The resulting index term vocabulary report gives to a human observer the feeling of being cleared of all errors normally produced in machine text processing. In this vocabulary, with each index term is associated the identification number of text items and sentences that contain the term.

As indicated in table 1, the process of indexing is divided into two parts: SCAN and PRELIMINARY TERM DISCRIMINATION ANALYSIS. These are described below.

SCAN

The process SCAN analyzes the text of each text item, extracts candidate index terms and generates phrases made up of the terms extracted.

The extraction of candidate index terms from the text is achieved in a signle pass guided by the user-terminal. The extracted terms consist of words, standard phrases and, if specified, user specified phrases. A

word is defined to be any string of characters set off by word delimiters on either side. A word delimiter may be any character which signals the beginning or ending of a word in text. Word delimiters are defined and entered through the terminal by the user. Blanks, left and right parentheses are typically defined as word delimiters. A standard phrase is defined as constituting a composition of words with certain syntactic and structural dependency. The class of standard phrases is made up of name phrases (U.S. Forces; Moore School of E.E.; etc.), date phrases (April 1, 1974; 15 May 1975; etc.), and time phrases (10:30 P.M.; etc.). A user specified phrase is defined as a sequence of specific words in a prescribed order. The component words and their relative orderings form a phrase dictionary, which is entered through the terminal by the user. As indicated in table 1, the specification of user specified phrases is optional.

In phrase analysis, sentence boundaries are recognized by sentence delimiters, which are defined and entered through the terminal by the user. A sentence delimiter may be any character which signals the beginning or ending of a sentence in text. For instance, period and question mark are typically defined as sentence delimiters.

During the extraction of candidate index terms, many high-usage words specified by the user in a "stop list" are rejected and the remaining words are automatically reduced to word stems. The stop list contains high-frequency, high-usage and multi-meaning words such as articles, conjunctions, propositions, auxilary verbs, and other high-usage verbs. Lists of such words are available in literature (3), and they number from 200 to 700 words and typically reduce the number of words to be extracted from the text from one to two thirds. The stop list may also contain words other than high-frequency ones, which are used in the automatic phrase generation

process where phrases are recognized by being delimited by stop list words or special characters. A modified version of the stop list, used by Borko (11) was used in processing the illustrative data-base.

Figure 4 shows the single-pass process of extraction of words and phrases from the text. The only portion of the flowchart that needs explanation is the component called "LEXICAL PROCESSOR." The goal of this processor is to take the input string of characters (i.e. the text of text item), which is presented to the processor in the English Language, and translate this into a string of grammatically correct P-sentences which make up the data-base (file 1) used in all subsequent processes including the phrase generation process.

A P-sentence is defined to be a set of quintuplets (L_j, T_j, IN, SN, A_j), where

L_j: The length of term T_j (in characters)

T_j: The extracted term, where a term is defined to denote a word, or a standard phrase or a user specified phrase.

IN: The identification number of the text item (assigned by the programmer who generates the Standard Formatted Text File) containing the term.

SN: The identification number of the sentence (assigned automatically) containing the term. This number denotes the relative position of the sentence in the item. The first sentence of the item is assigned the number '1'.

A_j: The relative term-address assigned to the term by the Lexical Processor. If the term T_j is a user specified phrase or a standard phrase consisting of adjacent words starting with capital letters, then its relative term-address is defined to be zero. If not, its relative term-address A_j is recursively defined as follows:

1. $A_{j+1} - A_j = 1$ If the terms T_{j+1} and T_j are adjacent in the text, that is, separated by one or more blank characters.

2. $A_{j+1} - A_j > 2$ If the terms T_{j+1} and T_j are not adjacent in the text, that is, separated by stop list word(s) and/or special character(s).

3. $A_1 > 1$

When the Lexical Processor completes the generation of a P-sentence (corresponding to a sentence of text in English Language), it instructs the system to save the quintuplets in the P-sentence on File 1. If the generation of a P-sentence is not completed, that is, the same sentence is still being processed, the system proceeds to extract the next term in text. Table 10 shows a portion of file 1 for the illustrative data-base.

Automatic phrase generation is achieved through the analysis of each P-sentence on file 1. Let P_{ij} denote the P-sentence generated from the jth sentence of the ith item, that is,

$$P_{ij} = (L_k, T_k, IN_i, SN_j, A_k) \mid k = 1, n$$

where n denotes the number of term tokens extracted from the sentence. Then, any string of terms $T_u\ T_{u+1} \ldots T_v$ $(v > u)$ is generated as a phrase if the following four conditions hold:

1. $A_j > 1, \quad j = u, v$
2. $A_{u+k} - A_{u+k-1} = 1, \quad k = 1, v-u$
3. $A_u - A_{u-1} > 1$
4. $A_{v+1} - A_v > 1$

where A_o and A_{n+1} are defined as:

$A_1 > 1 + A_o$ and $A_{n+1} > 1 + A_n$.

A phrase generated in this manner is actually a sequence of adjacent non-stop list terms in the same sentence of an item, which are separated by blanks and set off by stop list terms or special characters on both sides. Note that the first condition implies that only a word or a standard phrase, which is not a series of adjacent words starting with capital letters, can be a component term of such a phrase. Table 11 shows phrase tokens which have been generated from the term tokens in table 10.

For effective subsequent classifications of text items, index phrases must denote "alike" nature of text items that share these phrases as well as the "unlike" nature of text items that do not share these phrases. Therefore, it is necessary to simplify, reorganize and delete some phrases in order to convey this information. These processes are referred to as phrase refinement processes. Two refinement functions are provided: ERASE and DECOMPOSE. The first one enables the user to delete unsuitable index phrases, while the second one enables him to instruct the system to automatically decompose low-frequency phrases into sub-phrases or words which occur more frequently (assuming they convey more useful information). In processing the illustrative data-base, no deletion has been performed. However, the automatic decomposition process has been applied to all phrases of total frequency 1, which in turn caused the deletion of some phrases (i.e. those which could not be decomposed). For instance, the phrase "CENT OF U.S. NUCLEAR POWER" in table 11 has been transformed to "U.S. NUCLEAR POWER", since the sub-phrase "U.S. NUCLEAR POWER" already existed as a phrase. On the other hand, the phrase "DAMN NUISANCE" has been deleted through the automatic decomposition, because it had a total frequency of 1.

The term tokens of File 1 are next merged with the refined phrase tokens. Merging process requires both term and phrase tokens to be ordered by item and sentence identification numbers, as well as by term - and phrase-addresses respectively. The result of the merging process consists of all the refined phrase tokens and all the term tokens except those that constitute components of phrase tokens. Table 12 shows a portion of the file resulted from the merging process for the illustrative data-base. Note that the component terms of the refined phrase "U.S. NUCLEAR POWER", that is, "U.S NUCLEAR" and "POWER" (see Table 10), have been eliminated as a result of the merging process, and consequently they do not appear in table 12.

In the last phrase of SCAN (table 1), the term tokens (resulted from the merging process) in each item are ordered alphabetically, and in-item term types are produced by consolidating duplicate tokens within each item. During the process, in-item frequencies of term types are produced and associated with in-item term types. Table 13 shows a portion of the file resulted from this process for the illustrative data-base.

The resultant file makes up the source for the data-base for automatically classifying the text items on the basis of index terms assigned to the items. However, first the index terms which do not contribute to a satisfactory classification should be found and rejected from this file. This is partly achieved in the next phrase of the indexing process called preliminary term discrimination analysis.

PRELIMINARY TERM DISCRIMINATION ANALYSIS

The indexing system produces two reports to let the user find and reject the candidate index terms which do not contribute to a satisfactory classification. These two reports also enable the user to find and consolidate similarly synonymous terms so as to enhance the quality of subsequent

classification.

The first report, summary report, contains statistics on frequency distribution of candidate index terms, and advises the user of percentages of reduction in index term vocabulary size or average number of index terms per item resulting from mass term deletion processes. As indicated above, these reduction processes are performed in an attempt to enhance the quality of subsequent classification. The user thus can elect deletion of candidate index terms with total frequency 1, or with a high item or total frequency. The user can elect to except from the deletion names and dates which constitute candidate index phrases. The deleted terms are considered to be inefficient in the subsequent classification. Table 2 shows the beginning portion of a summary report produced for the illustrative data-base.

Examination of this report indicates that the initial number of candidate index term types ($RVS^{o}_{j_o}$.) is 20262 and the initial average number of in-item term types per item ($AT^{o}_{j_o}$) is 209. The subsequent numbers in the respective columns, that is, the fifth and last columns, show changes in these two totals if deletion of terms with the respective frequencies is performed. For instance, if all term types with item frequency 1 and total frequency less than or equal to 4 (i.e. 9074+715+118+45 term types) are deleted, then these two totals become 10310 and 187 respectively. It should be noted that the average number of in-item term types per item is an important factor which affects the quality of the subsequent classification (1,2). Additionally, the sixth column also shows the percentage of deleted term types.

Salton (6) reports three significant results on the term frequency distribution characteristics:

1. The best index terms are those with medium total frequency and an item frequency less than one half its total frequency.
2. The next best index terms are those with very low item frequency.
3. The least attractive index terms are those with a high item frequency and a total frequency exceeding the collection size.

The beginning and ending portions of the summary report become very valuable, when the user wants to delete unsuitable terms. The user may instruct the system to automatically perform deletion by specifying "frequency ranges" for term types. For instance, DELETE all term types with item frequencies 1-2 and total frequencies 1-4. Sometimes, some of the terms specified for deletion may be significant (e.g. names of people or places) and can be excluded from deletion process by indicating the respective terms to the system.

The second report contains lists of candidate index terms in which similarly spelled terms are brought together and arranged in groups. With each term is associated the respective total and item frequencies. The user can define the similarity of terms by specifying portions and number of characters which must be the same in terms. Dissimilarity can also be defined in a similar way. For instance, the system can be instructed to find those terms that have the same first three characters and have only two differences in the subsequent characters. This report is useful so as to locate and correct the errors (in spelling or typing), and find and consolidate the similarly spelled synonymous terms. In consolidation process, the frequencies associated with terms enable the user to choose one representative term to which the others are changed. These processes enhance the quality of subsequent classification process. Table 3 shows a portion

of such a report produced for the illustrative data-base. This table indicates that the term 'ACHIEV' can be changed to the term 'ACHIEVE', 'ADMINISTERE' to 'ADMINISTER', etc.

The file resulting from the preliminary term discrimination analysis is now used to generate item surrogates which make up the data-base for automatic classification of text items. An item surrogate is constituted by the respective item identification number and the codes for keys (index terms) assigned to the item. Note that using codes (integer numbers) instead of alphabetic terms in classification process tremendously increases the speed of classification process.

Finally, it should also be noted that a by-product of the indexing process consists of the Directory of Index Terms (see Figure 1). This directory is produced simply by ordering the file of in-item term types alphabetically and then consolidating multi-occurrences of in-item term types in the data-base. Table 9 shows a portion of the Directory of Index Term Types for the illustrative data-base.

CLASSIFICATION

Classification is essentially the result of a process to organize a set of objects in a systematic fashion so that "alike" objects are placed near each other. As indicated in table 1, the classification processes serve for three purposes:

1. Item classification
2. Key classification
3. Term discrimination analysis

ITEM CLASSIFICATION

The objective of item classification is to group "alike" items together into cells or near each other. Cells are similar to the shelves in a library where sets of "alike" objects are stored together. Likeness of items is determinable and measurable by the use of common index terms in the res-

K ITEM FRQ	M_k # OF TERMS	$TF^k_{j_k}$ TOT FRQ	$C^k_{j_k}$ # OF TERMS	$RVS_{j_k}^k$ REMAINING VOC SIZE	$PT_{j_k}^k$ % OF TERMS	$AT_{j_k}^k$ # OF TERMS/DOC
				20262		209
1	(10041)	1(	9074)	11188	44%	188
		2(	715)	10473	48%	187
		3(	118)	10355	48%	187
		4(	45)	10310	49%	187
		5(	27)	10283	49%	187
		6(	17)	10266	49%	187
		7(	10)	10256	49%	187
		8(	5)	10251	49%	187
		9(	9)	10242	49%	187
		10(	5)	10237	49%	187
		11(	4)	10233	49%	187
		12(	1)	10232	49%	187
		14(	1)	10231	49%	187
		15(	2)	10229	49%	187
		16(	4)	10225	49%	187
		18(	1)	10224	49%	187
		19(	1)	10223	49%	187
		20(	1)	10222	49%	187
		29(	1)	10221	49%	187
2	(3640)	2(	3170)	7051	65%	173
		3(	307)	6744	66%	172
		4(	82)	6662	67%	172

TABLE 2

SUMMARY REPORT

Beginning portion of example showing frequency distribution of terms and other statistics for the illustrative data-base.

To be similar, adjacent terms must match on at least the first 3 characters. Maximum Length of Disagreement :2

The following terms have similar spellings and are candidates for consolidation:

CODE	ITEM FRQ	TOT FRQ	TERM TYPE
121	5	5	ACCUSAT
122	30	34	ACCUSE
123	6	6	ACCUSTOME
124	1	1	ACCUSTOM
130	37	42	ACHIEVE
131	1	1	ACHIEV
139	3	3	ACQUAINTANCE
140	3	4	ACQUAINTANC
151	1	1	ACTIVIST
152	14	15	ACTIVITY
162	1	1	ADAPTAT
163	3	3	ADAPT
169	1	1	ADDRESS
170	20	21	ADDRES
176	4	7	ADEN
177	2	2	ADEPT
180	2	3	ADHERENCE
181	4	4	ADHERENT
194	1	1	ADMINISTERE
195	4	4	ADMINISTER
197	9	11	ADMINISTRATOR
198	15	23	ADMINISTRAT
199	1	1	ADMIRABLE
200	1	1	ADMIRAB

Table 3

Example of search of candidate index terms to discover spelling errors and consolidate terms. Note: Only portion of output is shown.

pective items. The classification algorithm (see appendix B) used here is considered to be of a "divisive" type. It forms a classification tree by succcessively dividing the collection of item surrogates, on the basis of co-occurrences of index terms assigned to the items, until desired sized groups are generated. An item identification number and its respective codes (for index terms assigned to the item) are referred to as item surrogate.

Each node of the classification tree is associated with an exclusive sub-set of the collection of items. The node can also be considered to have associated with a non-exclusive sub-set of the key vocabulary, consisting of the union of keys used to index the respective items. Each node can be assigned a classification number based on its position in the tree. If a node occurs at the L th level of the tree as the K th node from left, then it is assigned the integer classification number

$$N_{LK} = K + \frac{N^{L-1} - 1}{N - 1}$$

or the canonical representation of this number in the form:

$$CAN\ (N_{LK}) = X_L . X_{L-1} \cdot \cdot \cdot X_2 . X_1, \quad \text{where } X_L = 1$$

$$X_U = MOD\ \left((N_{j_u} - C_u), N\right) + 1 \qquad u = 1, L-1$$

$$N_{j_k} = \left\lfloor \frac{N_{j_{k-1}} + N - 2}{N} \right\rfloor \qquad k = 2, L-1$$

$$N_{j_1} = N_{LK}$$

$$C_u = 1 + \frac{N^{L-U} - 1}{N-1}$$

N: Stratification number for the tree.

A terminal node of the tree constitutes a cell of items and its classification number is assigned to all items within the cell. Note that the classification tree generated is not necessarily a balanced (regular) tree, that is, one in which all terminal nodes are at the same level. This is so, since one node may meet the desired group size at one level while another node may need further partitioning. Figure 2 shows a sub-tree of the classification tree for the illustrative data-base.

Integer node numbers are not as useful as the corresponding canonical node numbers. When searching and browsing through the classification tree. This is so, since a canonical node number enables the user to locate the node in question in a tree in a simple manner. For instance, as indicated in figure 2, the canonical node number 1.1.3.3 indicates a node in the fourth level of the tree (4 digit positions) which emenates from the root node (1) and from the 1st node in the second level of the tree (the second digit from left), and finally from the 3rd node in the third level of the tree. However, the system stores the node numbers as integers and not in their canonical form for the purpose of space conservation. Therefore, the Retrieval System (12) always converts a canonical node number indicated by the user to the corresponding integer form by using the formula:

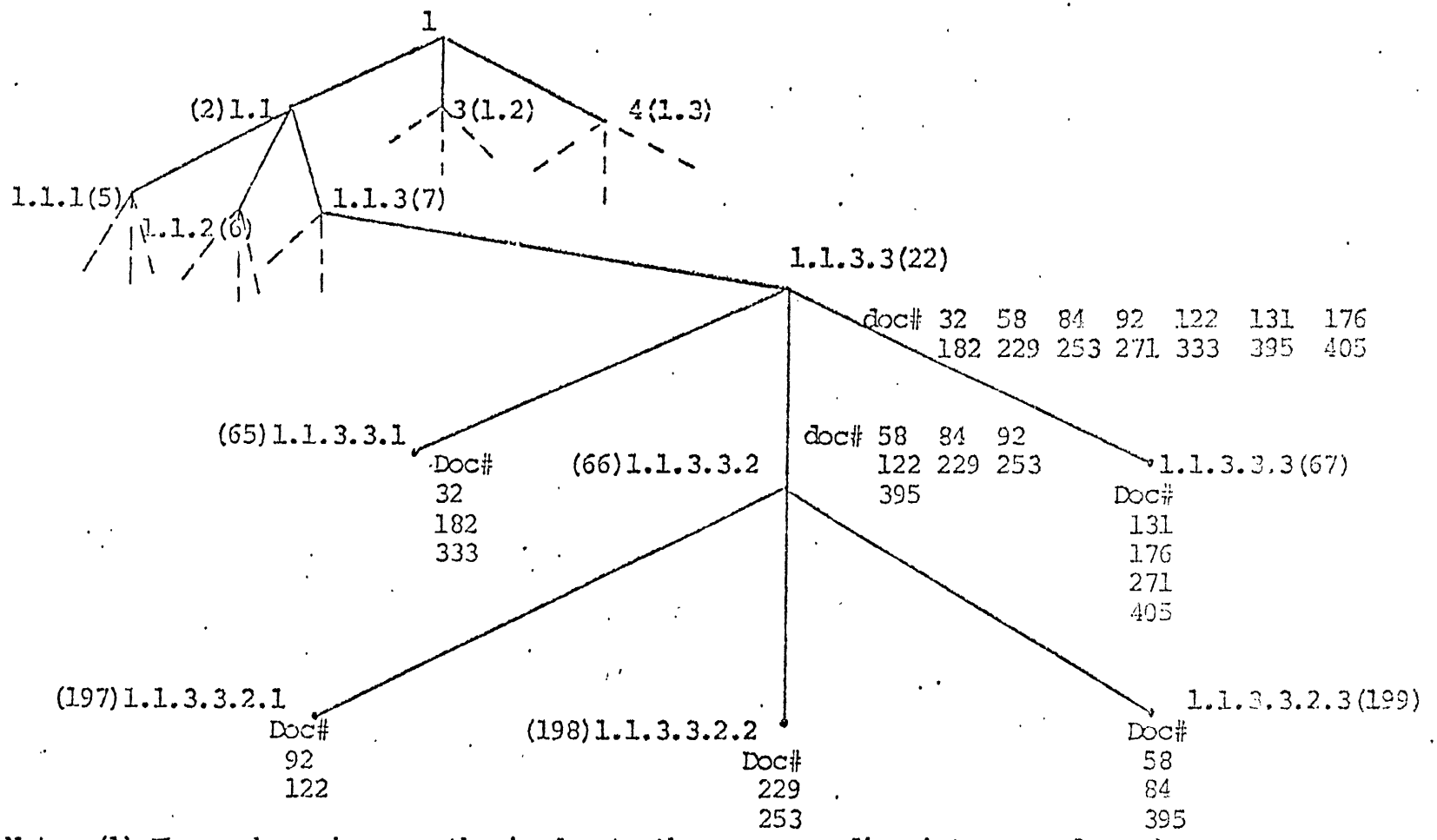
$$N_{LK} = X_1 N^0 + X_2 N^1 + \ldots + X_{L-1} N^{L-2} + X_L$$

where $CAN(N_{LK}) = X_L X_{L-1} \ldots . X_2 X_1$ and $X_L = 1$

The classification tree created is not suitable for browsing and searching. Therefore, a hierarchical tree is formed by intersecting key sets of the nodes and assigning the resulting keys to the parent node (next level up the hierarchy), and in turn deleting these resulting keys from the original nodes. Figure 3 illustrates a sub-tree of the hierarchical classification tree produced for the illustrative data-base. In fact, this sub-tree corresponds to that part of the classification tree illustrated in figure 2. This figure shows only

portions of key vocabularies associated with the nodes of the tree, because of the space limitation. As this illustrative sub-tree indicates, the key vocabulary assigned to a node gives the user certain insights so as to understand the subject matter of the text items in the cells encompassed by the node. For instance, node 1.1.3.3 indicates that the items encompassed are primarily concerned with affairs of Britain. Further division of the node 1.1.3.3 divides these affairs into economical problems of Britain, the affairs of Christine Keeler (Profumo Scandal), and the protests and marches of communist agitators from England, Scotland and Ireland against the House of Common. Further division of the node 1.1.3.3.2 can be interpreted in the same fashion.

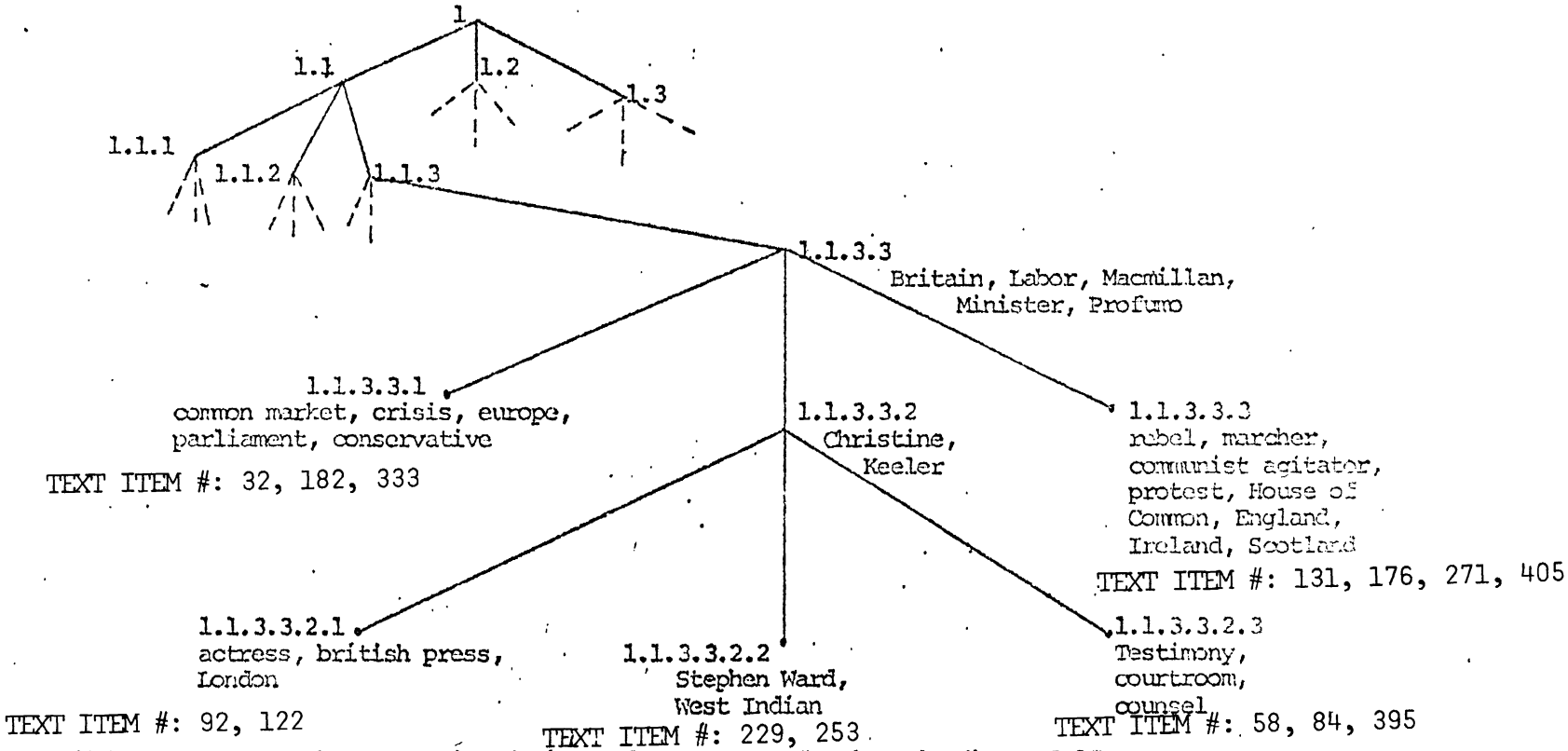
The hierarchical classification tree with keys at the nodes is described in two classification schedules: Key-to-node Directory and Node-to-key Directory. The Key-to-node Directory is ordered alphabetically by keys and contains for each key a list of canonical classification numbers of the nodes of the hierarchical classification tree, with which the key is associated. Each list of canonical classification numbers constitutes a "prefix language". Vice-versa, the Node-to-key Directory contains for each classification number a list of keys ordered alphabetically, which appear at the node (of the hierarchical classification tree) with the classification number in question. This directory is ordered by canonical classification numbers in a manner such that the ordered set of canonical classification numbers constitutes a "prefix language". Table 4 and 5 show the top and bottom portions of the Key-to-node and Node-to-key Directories respectively for the illustrative data-base.



Note: (1) The numbers in parenthesis denote the corresponding integer node numbers
(2) Stratification number = 3
(3) Group Size = 4 documents
(4) No. of nodes = 280
(5) No. of cells = 187

Figure 2

Example of a Sub-tree of the Classification Tree for the Illustrative Data-base



Note: Document numbers associated with nodes are already given in figure 3.17.

Figure 3

Example of a Sub-tree of the Hierarchical Classification Tree for the Illustrative Data-base

The classification schedules and the classified data-base can be used on an interactive basis through the Automatic Search and Retrieval System (12). However, if the user wishes, they can be placed on microfilm, where they can be searched by a microfilm reader, or in print-out form which can be searched manually.

KEY CLASSIFICATION

The objective of key classification is to bring "affinitive keys" near each other. Affinity of keys is determinable and measurable by the use of respective keys in common text items. The classification algorithm used in classifying text items is also used in classifying keys. The algorithm forms a classification tree by successively dividing the collection of key surrogates, on the basis of co-occurrences of classification process, until desired sized groups are generated. The code of a key and the classification numbers of cells associated with the key are referred to as a key surrogate.

The product of the key classification is an Affinity Dictionary, in which the keys are ordered by respective canonical classification numbers assigned to them by the key classification process. The ordered set of canonical classification numbers constitutes a "prefix language". This special ordering reflects the "affinity" of keys and consequently enables the user to find synonymous or near synonymous keys. Table 7 shows a portion of the Affinity Dictionary for the illustrative data-base. As can be examined, "ANKARA", "TURKEY" and "INONU" (the prime minister of

KEY	NODE
ABDEL	1.2.3.1.3.2; 1.3.1.1.1.1; 1.3.1.2.2.2; 1.3.1.2.3; 1.3.1.3.3.3; 1.3.2.3.1; 1.3.3.1.2; 1.3.3.3.3.1;
ABDEL NASSER	1.2.3.1.3.2; 1.3.2.2.3;
ABDUL	1.2.1.2.1.2; 1.2.1.2.1.3; 1.2.1.2.2; 1.2.1.2.3.1; 1.3.1.3.2.1; 1.3.1.3.3.1; 1.3.3.1.2;
ABDUL RAHMAN	1.2.1.1.1.2; 1.2.1.2.2; 1.2.1.2.3.1; 1.2.1.3.3.2; 1.2.1.3.3.2; 1.2.2.3.2;
ABDULLAH	1.1.2.1.1; 1.1.3.1.1; 1.3.1.1.2.2; 1.3.1.2.3; 1.3.3.1.2; 1.3.3.3.2.2;
ABDULLAH SALLAL	1.3.1.1.2.2; 1.3.1.2.1.1; 1.3.1.2.2.2; 1.3.2.2.3;
YUGOSLAVIA	1.1.1.3.2.1; 1.1.2.3.1.2; 1.1.2.3.1.3; 1.1.3.1.2.1; 1.1.3.2.3.3; 1.2.1.2.1.1; 1.2.2.3.3.3; 1.3.1.2.2.2; 1.3.1.3.1.3; 1.3.1.3.2.3; 1.3.1.3.3.2; 1.3.2.2.1; 1.3.3.2.2.1;
YUN	1.1.3.2.2.3; 1.1.3.2.3.2; 1.3.2.1.1.3;
ZANZIBAR	1.2.1.1.2; 1.2.1.2.1.3; 1.2.1.2.2; 1.2.2.3.3.1;
ZERMATT	1.2.1.1.2;
ZIONIST	1.1.3.1.1; 1.3.1.2.3; 1.3.3.3.2.3;
ZURICH	1.1.2.3.3; 1.2.1.2.1.2; 1.2.3.2.1.2;
13TH CENTURY	1.1.1.1.3; 1.1.1.3.3.3; 1.3.2.3.3.1; 1.3.3.1.2;
18TH CENTURY	1.1.2.1.3.3; 1.1.2.3.3; 1.1.3.2.1; 1.1.3.3.1;
19TH CENTURY	1.1.2.1.3.3; 1.2.3.2.1.3; 1.3.3.2.2.2; 1.3.3.2.3;

NUMBER OF KEY TYPES = 1667

Table 4

Top and Bottom Portions of the Key-to-node Table For The Illustrative Example

	NODE	KEYS
	1	
	1.1	
	1.2	
	1.3	
	1.1.1	
	1.1.2	
	1.1.3	
	1.2.1	
	1.2.2	
	1.2.3	
	1.3.1	
	1.3.2	
	1.3.3	
	1.1.1.1	COMMUNIST; MOSCOW; RED CHINA; RUSSIA; WASHINGTON;
	1.1.1.2	RUSSIA;
	1.1.1.3	
	1.1.2.1	
	1.1.2.2	RUSSIA;
	1.1.2.3	WASHINGTON;
	1.1.3.1	
	1.1.3.2	
	1.1.3.3	BRITAIN; LABOR; MACMILLAN; MINISTER; PROFUMO;
Terminal Cell →	*1.1.2.2.3.3.2	AGENT; BORDER; BUDAPEST; CARDINAL; CATHOLIC; DIPLOMAT; EMBASSY; FOREIGNER; HUNGARY; IRON; ISRAEL; JOSEF; MINDSZENTY; MINISTRY; MONGOLIA; MOSCOW; PEASANT; PHOTOGRAPH; PROTESTANT; ROMAN; SIBERIA; TESTAMENT; TRAIN; U.S. EMBASSY; U.S. OFFICIAL;
Terminal Cell →	*1.1.2.2.3.3.3	ARM; BRITAIN; CABINET MINISTER; CHAMBER; CONGRES; CONGRES PARTY; DEFENSE; DELHI; ENTERPRISE; FOOD; GENERAL ELECT; INDEPENDENCE; INDEPENDENT; INDIA; JAWAHARLAL; KRIPALANI; LEFTIST; MINISTER; MISSILE; NEHRU; PARLIAMENT; POLITICAL; PRO-WESTERN; PROPOSAL; VICTORY;

NO. OF NODES PRINTED = 280
NO. OF CELLS PRINTED = 187
AVERAGE NO. OF KEYS PER CELL = 78

Table 5

Top and Bottom Portions of The Node-to-key Table For The Illustrative Example
Note: In each node, keys are ordered alphabetically.

1.1.1.1.1 CANONICAL CLASSIFICATION (CELL) NUMBER
0001 ITEM IDENTIFICATION NUMBER

*TEXT 017 01/04/63 PAGE 020
THE ALLIES AFTER NASSAU IN DECEMBER 1960, THE U.S. FIRST PROPOSED TO HELP NATO DEVELOP ITS OWN NUCLEAR STRIKE FORCE. BUT EUROPE MADE NO ATTEMPT TO DEVISE A PLAN. LAST WEEK, AS THEY STUDIED THE NASSAU ACCORD BETWEEN PRESIDENT KENNEDY AND PRIME MINISTER MACMILLAN, EUROPEANS SAW EMERGING THE FIRST OUTLINES OF THE NUCLEAR NATO THAT THE U.S. WANTS AND WILL SUPPORT.
.....
KEYS ASSIGNED TO KEYS NUMBER 0001

ADENAUER	ALLIANCE	ALLY
ATLANTIC ALLIANCE	ATOMIC	BOMB
BRITAIN	BRITISH MEMBERSHIP	BRITON
CANNON	CHANCELLOR	CHARL
CITIZENSHIP	COMMON MARKET	CONFLICT
CRISI	DE GAULLE	DEFENSE
EUROPE	FIASCO	FRANCE
FRENCH	GERMAN	GERMANY
HAROLD MACMILLAN	INDEPENDENT	JACK
KENNEDY	KNIGHT	KONRAD ADENAUER
MACMILLAN	MANUFACTURER	MILITARY
MINISTER	MISSILE	NASSAU
NATIONAL	NATO	NUCLEAR
NUCLEAR WEAPON	POLARI	POLARI SUBMARINE FLEET
PRESIDENT KENNEDY	PRIME MINISTER MACMILLAN	PRIZE
RUSSIA	STRATEGIC	TORY
U.S.	U.S. NUCLEAR POWER	VAUNT
WEAPON	WEST GERMANY	

Table 6
Portion of Classified Data-base Report For The Illustrative Example

"TURKEY" in 1963 when the article was published in Times Magazine) are indeed synonyms.

Let it be noted that the Affinity Dictionary can be used on an interactive basis through the Automatic Search and Retrieval System (12), or manually when it is produced on microfilm or in print-out form.

TERM DISCRIMINATION ANALYSIS

Classification can be repetitively performed to produce reports, aiding the user in the final term discrimination analysis. Two reports are generated for this purpose.

The first report contains the entire set of keys ordered by respective node frequencies. The node frequency of a key is defined to be the number of nodes at the hierarchical classification tree, with which the key is associated. If the number of nodes associated with a key is relatively large, this would indicate that the key is not effective in classifying text items and consequently can be deleted from the vocabulary (with the exception of names and dates constituting keys). Table 8 shows the beginning and ending portions of such a report produced for the illustrative data-base.

The second report consists of an Affinity Dictionary, which has already been discussed in previous section. This report gives the user the ability to find and consolidate synonymous or near synonymous keys. Ambiguous and vague keys can be expanded or consolidated using this report. These processes enhance the quality of subsequent classification and information retrieval.

CELL#	KEY	NODE#
1.2.4.3.4.2.1	AIR FORCE	1.2.1.2.2; 1.2.1.2.3.1; 1.2.3.2.1.2; 1.2.3.3.2.1;
		1.3.1.2.1.2; 1.3.1.3.3.1; 1.3.2.1.1.3; 1.3.2.1.2.1; 1.3.3.1.2; 1.3.3.3.1.1;
1.2.4.3.4.2.2	ANKARA	1.2.3.2.1.2; 1.2.3.2.2.1; 1.3.1.3.1.2 1.3.1.3.3.2;
1.2.4.3.4.2.3	TURKEY	1.1.1.1.2.1; 1.1.3.1.3; 1.2.1.2.2; 1.2.1.2.3.3; 1.2.3.2.1.2; 1.2.3.2.2.1; 1.3.1.2.1.1; 1.3.1.2.1.3; 1.3.1.3.1.2; 1.3.1.3.2.2; 1.3.2.3.1; 1.3.3.1.2;
1.2.4.3.4.2.4	INONU	1.2.1.2.3.3; 1.2.3.2.1.2; 1.3.1.2.1.3; 1.3.1.3.1.2;
1.2.4.3.4.3.1	MUNICH	1.1.1.1.3; 1.1.2.2.3.1; 1.2.1.2.1.1; 1.2.1.3.1; 1.2.1.3.2; 1.2.2.2.2; 1.2.2.2.3.1; 1.2.2.2.3.2; 1.2.2.3.3.1; 1.2.3.2.1.2; 1.2.3.2.3.1; 1.2.3.3.3; 1.3.1.1.1.3;

Table 7
A Portion of the Affinity Dictionary for the Illustrative Data-base.

NODE FREQ	CODE	KEY
89	1257	POLITICAL
87	1553	U.S.
80	1061	MINISTER
80	229	BRITAIN
76	479	DEATH
74	545	ECONOMY
71	397	COMMUNIST
69	650	FRENCH
69	105	ARMY
68	1382	RUSSIA
65	1254	POLICE
64	634	FRANCE
57	1052	MILITARY
57	578	EUROPE
56	1105	NATIONAL
56	555	ELECT
55	1358	REGIME
55	1192	PARLIAMENT
55	300	CAPITAL
54	960	LONDON
53	1283	PRES
53	1164	OFFICER
52	486	DEFENSE
51	615	FIRE
51	102	ARM
49	64	AMERICAN
47	1623	VOTE
47	349	CHINA
46	1084	MOSCOW
46	290	CAMPAIGN
45	1628	WASHINGTON
45	1262	POLITICIAN
45	1190	PARI
45	817	JAIL
1	33	AGUNG
1	10	ABUBAKAR TAFAWA BALEWA
1	8	ABU DHABI

Table 3
Example of Search of Keys for Further Deletion
Note: Only top and bottom portions of output for the illustrative example is shown.

It should be noted that item and key classifications should be repetitively performed until the quality of the item classification is judged to be satisfactory. In this respect, a ceiling value for the number of nodes per key can be subjectively determined by examining the first report. For instance, for the illustrative data-base, a ceiling of 10 nodes per key was considered to be satisfactory.

CONCLUSIONS

The research reported here accomplished to integrate several components and create an integrated indexing and classification system, which can be used in indexing and classification of text items.

The system was first tested by processing of 1669 text items in world affairs supplied by Foreign Broadcast Information Service. However, another data-base of 425 text items in world affairs taken from issues of Times magazine published in 1963 have been processed interactively, with much greater care resulting in a "well" indexed data-base, in the sense that ambiguous and vague index terms have been systematically investigated, and as a result, these terms have been consolidated, expanded or dropped altogether. The index term vocabulary report gives to a human observer the feeling of being cleared of all errors normally produced in machine text processing. Finally, a third experiment involved generating an index to a dissertation (2).

Qualitatively, and necessarily subjectively, the indexing and classification make sense when examined by humans, and are judged as useful for retrieval. Naturally, in one way or another, a procedure must be used to evaluate retrieval effectiveness. The experiments, to

evaluate retrieval effectiveness, are in progress, and will hopefully result in valuable retrieval statistics.

Other important achievements of the reported research are summarized below.

1. System interactiveness
2. Automatic Index Phrase Recognition
3. Summary Report, informing the user of the impact of user elected decisions to delete candidate index terms on a mass basis and advising him of percentages of reduction in index term vocabulary size or average number of index terms per item resulting from such mass term deletions.
4. Affinity Dictionary, giving the user the ability to locate synonymous or near synonymous index terms.
5. Use of classification processes in discriminating index terms which do not contribute to a satisfactory classification.

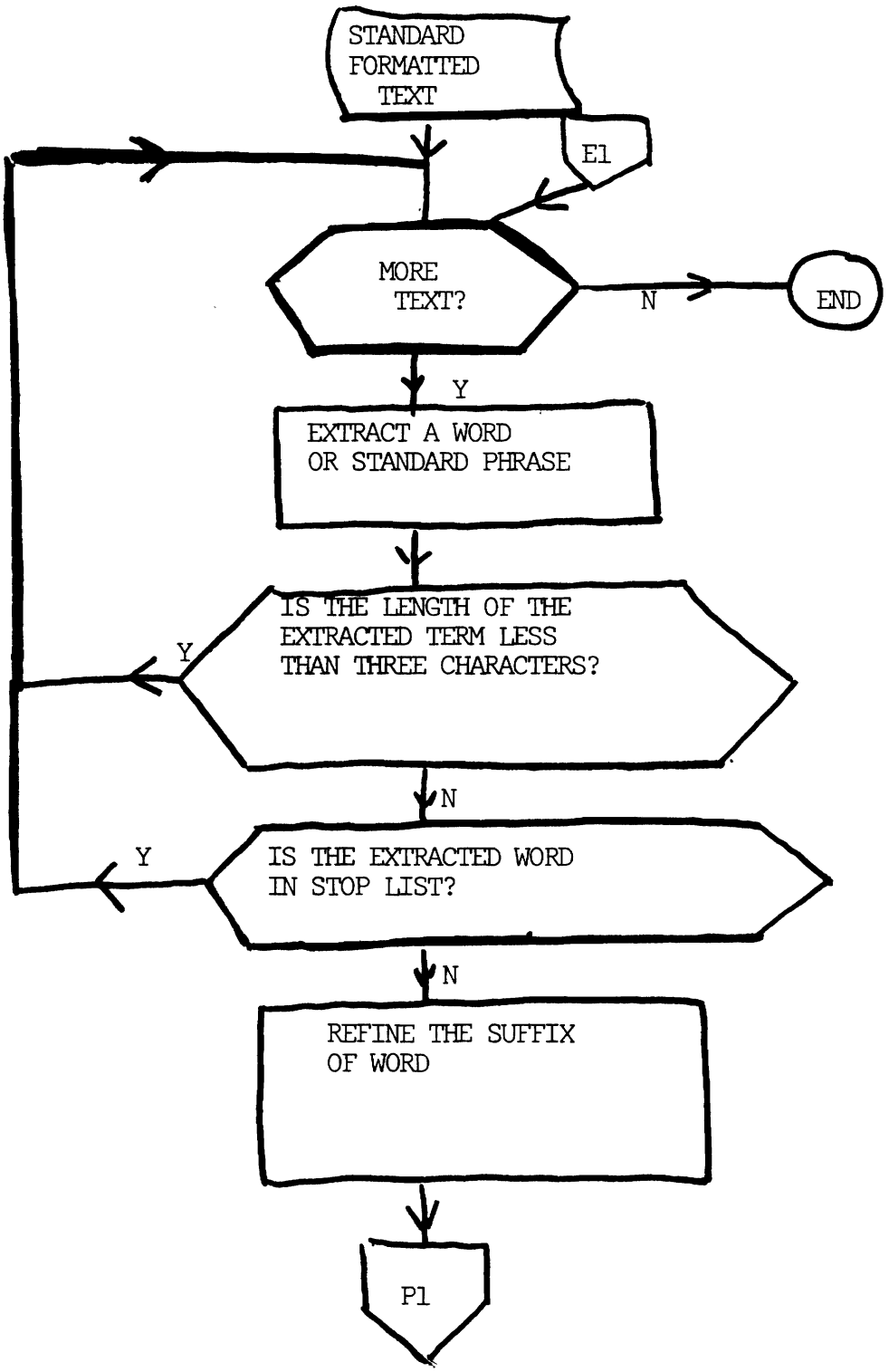


FIGURE 4
PROCESSES OF THE EXTRACTION OF WORDS AND PHRASES FROM THE TEXT

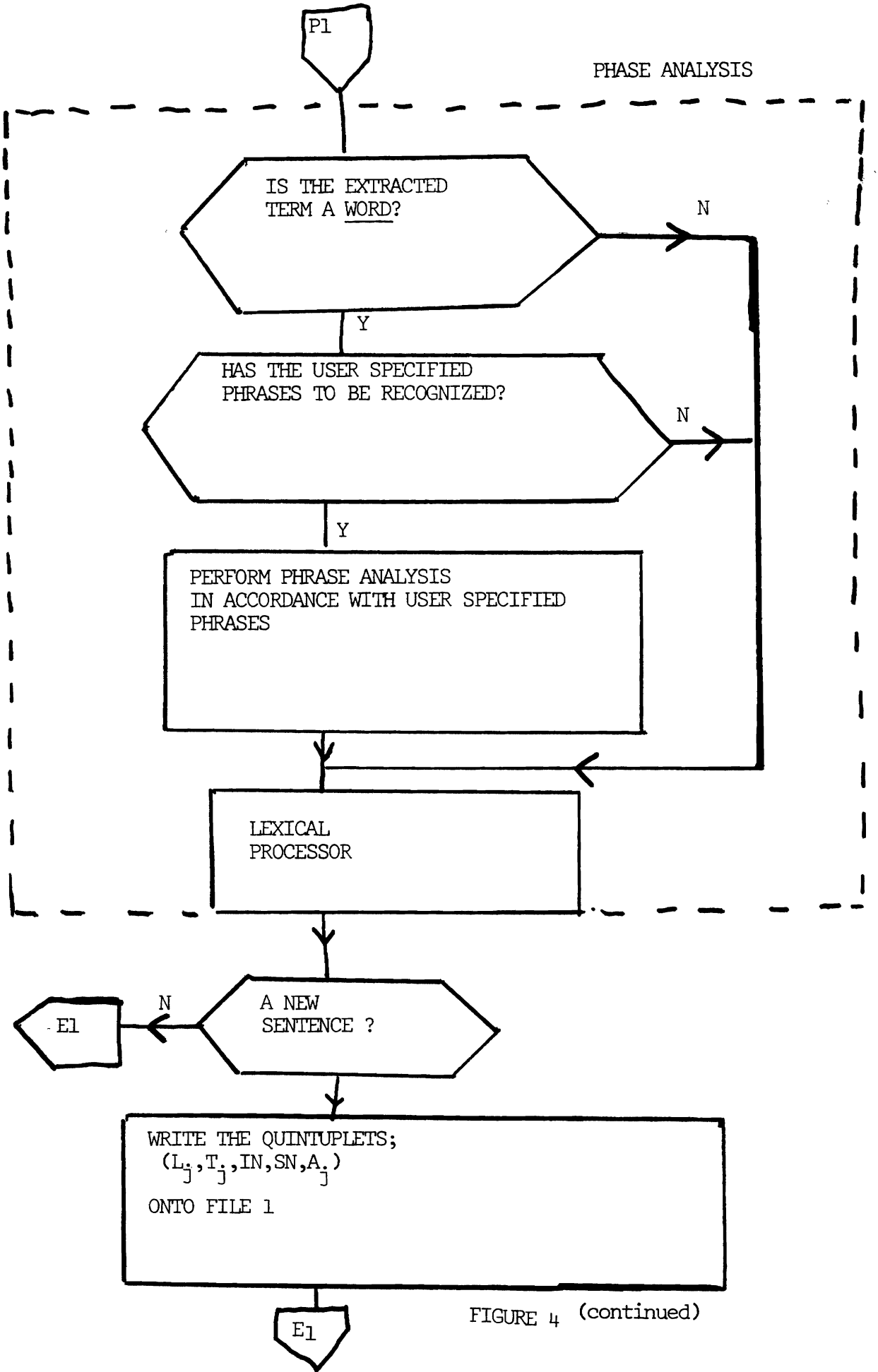


FIGURE 4 (continued)

ITEM FREQ	TOTAL FREQ.	TERM TYPE
2	3	ABBE

ITEM #	IN-ITEM FREQ.	SEN.#
319	1	1
163	2	7;27

ITEM FREQ	TOTAL FREQ.	TERM TYPE
3	3	ABDEL NASSER

ITEM #	IN-ITEM FREQ.	SEN. #
115	1	10
227	1	14
230	1	6

TABLE 9
PORTION OF THE DIRECTORY OF INDEX TERMS FOR THE ILLUSTRATIVE DATA-BASE.

L_j TERM LENGTH	T_j TERM TOKEN	IN ITEM #	SN SEN.#	A_j TERM-ADDRESS
6	SOVIET	1	35	29
8	RETALIAT	1	35	30
6	EFFORT	1	36	5
7	BRITISH	1	36	8
6	FRENCH	1	36	10
6	MANAGE	1	36	16
6	CREATE	1	36	18
7	NUCLEAR	1	36	20
8	CAPACITY	1	36	21
9	REPRESENT	1	36	23
4	CENT	1	36	29
2	OF	1	36	30
13	U.S. NUCLEAR	1	36	31
5	POWER	1	36	32
4	DAMN	1	37	7
8	NUISANCE	1	37	8
5	STATE	1	37	17
10	DEPARTMENT	1	37	18
8	OFFICIAL	1	37	19
4	MEAN	1	38	4

TABLE 10

PORTION OF FILE 1 FOR THE ILLUSTRATIVE DATA-BASE

Notes: 1. File 1 is the result of candidate index term (token) extraction process.
2. Word Delimiters used : < > ( ) % @ ' " blank
3. Sentence Delimiters used: . ? !
4. Portion of text item 1 from which the term tokens shown in the table have been extracted is given below:

"----------
Fact is, the British and French Nuclear weapons could never be used independently of the U.S. against Russia without inviting devastating soviet retaliation. After all their efforts, the british and french will have managed to create a nuclear capacity that represents only 4 per cent of U.S. Nuclear power./ It is just a damned nuisance, / said a state department official last week./ It means nothing military except that we will be expected to bail out the first country that throws the first --------
----------"

Table 10
- continued -

Note that the term tokens in the table have been extracted from the underlined portions of the above text.

PHRASE LENGTH	ITEM #	SEN.#	NUMBER OF MEMBERS	PHRASE-ADDRESS	PHRASE TOKEN
15	1	35	2	29	SOVIET RETALIAT
16	1	36	2	20	NUCLEAR CAPACITY
27	1	36	4	29	CENT OF U.S. NUCLEAR POWER
13	1	37	2	7	DAMN NUISANCE
25	1	37	3	17	STATE DEPARTMENT OFFICIAL

TABLE 11

List of phrase tokens which have been generated from the term tokens in Table 10.

Note: The phrase-address of a phrase token is defined to be the term-address of its first component term, and is later used in eliminating the respective component terms from further consideration through the merging process.

ITEM #	SEN. #	TERM TOKEN
1	35	SOVIET RETALIAT
1	36	EFFORT
1	36	BRITISH
1	36	FRENCH
1	36	MANAGE
1	36	CREATE
1	36	NUCLEAR
1	36	CAPACITY
1	36	REPRESENT
1	36	CENT
1	36	U.S. NUCLEAR POWER
1	37	DAMN
1	37	NUISANCE
1	37	STATE
1	37	DEPARTMENT
1	37	OFFICIAL
1	38	MEAN

TABLE 12

List of term tokens resulted from the merging of term tokens in table 10 with phrase tokens in table 11.

ITEM #	IN-ITEM TERM TYPE	IN-ITEM FREQ.	SEN. #
1	ARSENAL	2	34;41
1	ATLANTIC ALLIANCE	1	26
.	.	.	.
.	.	.	.
.	.	.	.
1	BRITAIN	13	6;8;9;13;15;16;
.	.	.	18;21;25;26;27;
.	.	.	28;40
.	.	.	
1	COMMON MARKET	1	27
.	.	.	.
.	.	.	.
.	.	.	.

TABLE 13

Portion of File of in-item term types for the Illustrative Data-base.

BIBLIOGRAPHY

1. Jones, K.S.: Collection Properties Influencing Automatic Term Classification Performance, Information Storage and Retrieval, Vol. 9, pp. 499-513, Pergamon Press, 1973.

2. Koymen, K.: TOS: A Text Organizing System, Doctoral Dissertation, University of Pennsylvania, 1974.

3. Kucera, H., and W. N. Francis: Computational Analysis of Present Day American English, Brown University, Providence, Rhode Island, 1967.

4. Litofsky, B.: Utility of Automatic Classification Systems For Information Storage and Retrieval, Doctoral Dissertation, University of Pennsylvania, 1969.

5. McEowen, James R.: A Design For A Personal Automated Library, Doctoral Dissertation, University of Pennsylvania, 1969.

6. Salton, G.: A Theory of Indexing, Department of Computer Science, Cornell University, 1974.

7. Sokal, Robert R.: Numerical Taxanomy, Scientific American, Vol. 215, No. 6, December 1966, pp. 106-116.

8. Prywes, N.S., A.L. Lang and S. Zagorsky: A-posteriori Indexing, Classification and Retrieval of Textual Data. Information Storage and Retrieval, Vol. 10, pp. 15-27, Pergamon Press, 1974.

9. Lang, L.A., and Zagorsky, S.: Implementation of an Automatic, A-posteriori, Hierarchical Classification System, Master thesis, The University of Pennsylvania, The Moore School of Electrical Engineering, December, 1972.

10. Lefkovitz, D.: File Structure for On-line Systems, Spartan Books, March 1969, See Appendix B.

11. Borko, H.: Experiment In Book Indexing by Computer, Information Storage and Retrieval, Vol. 6, pp. 5-16, Pergamon Press, 1970.

12. Horton, M.: Automatic Retrieval System. Ph.D. Dissertation in Preparation, University of Pennsylvania.

ACKNOWLEDGEMENTS

The author would like to express his gratitude to Dr. Noah S. Prywes (8), for his help and advice throughout the research performed.

The results of past work performed by Mrs. S. Zagorsky and Mr. A. Lang (9), and Dr. D. Lefkovitz (10) have been gratefully used in pursuing the research reported here. I am grateful to Mr. Mike Horton*, with whom I had valuable discussions in regard to the research performed. Thanks are also due to Mrs. Fahimeh Jalili, and Mr. Yang Chang who helped in processing the data-base.

Finally, I am also grateful to Miss Barbara Weber for the typing up of this paper, and to the Information System Branch of the Office of Naval Research for supporting this research.

*Mr. Mike Horton is a graduate student in the Moore School of Electrical Engineering, University of Pennsylvania, and is currently writing his dissertation in the field of information storage and retrieval.

APPENDIX A

GLOSSARY

Affinity Dictionary: A dictionary in which the keys are ordered by respective classification numbers assigned to them by the key classification process, in a manner such that the ordered set of classification numbers constitutes a prefix language.

A-posteriori: used adjectively, of knowledge or cognition originating entirely based on experience (examination of text items).

A-posteriori classification: Classification in which none of the information needed for the classification is available separately or independently of the objects being classified.

A-posteriori indexing: Indexing in which none of the information needed for the indexing is available separately or independently of the text items being indexed.

Candidate Index Terms: Index terms to which the term discrimination analysis should be applied to find and retain the suitable ones as well as to find and reject the unsuitable ones. i.e. those which do not contribute to a satisfactory classification.

Cell: A group of limited number of "alike" items, or keys grouped together by the automatic classification process. The cell constitutes also a terminal node of the classification tree.

Classification: An arrangement of objects in classes or groups. This is required to produce a classification system.

Classification number: A prefix language canonical number assigned to each node of a (hierarchical) classification tree, based on the position of the node in the tree. Keys can be assigned to nodes of a hierarchical classification tree but text items can be assigned only to terminal nodes.

Classification Schedule: A set of hierarchically structured classification rules, such as progressively more specialized subject areas, used in the classification of text items or keys.

Classification System: A scheme of organization of objects defined in a classification schedule, by which objects can be classified, namely progressively assigned to more specialized groups of objects. The classified objects may be text items or keys.

Classification Tree: A tree in which each node is associated with an exclusive sub-set of the collection of items and a non-exclusive sub-set of the key vocabulary, consisting of the union of keys used to index the respective items.

Classified Data-base: The rearranged data-base in which the text items are ordered by their respective canonical classification numbers.

Code of a Key: A sequential number assigned to the key, to facilitate references to key.

Hierarchical Classification Schedule: see classification schedule.

Hierarchical Classification Tree: A representation of a hierarchical classification schedule in tree form, where the more generic classification rules are at a parent node and the more specialized classification rules are at the sibling nodes.

Identification numbers: Sequentially generated numbers assigned to text items and sentences within each text item.

Index Term: Name, concept, descriptor, affiliation, word, phrase, etc., that may be assigned to a text item for use ultimately for classification or retrieval of the items.

Indexing: The assignment of one or more several index terms to a text item.

In-item Frequency of Term Type: Number of occurrences of the term type within an item.

In-item Term Type: The unique occurrence of term type within an item.

Item: see Text item.

Item classification: The generation of classification system for classifying text items and the assignment of classification numbers to text items. The classified text items, when ordered by the respective classification numbers, are also referred to as the classified data-base of text items.

Item Frequency of Term Type: Number of items to which the term type has been assigned as an index term.

Item Surrogate: An item identification number and its respective codes for keys (index terms) assigned to the item.

Key: A candidate index term that has been evaluated in the term discrimination analysis and determined to be effective in classifying items and in retrieval. It is to be retained as a basis for classification and subsequent retrieval. Other candidate index terms are rejected in the term discrimination analysis.

Key Classification: The generation of classification system for classifying keys and the assignment of classification numbers to keys. The keys, when ordered by the respective classification numbers, are referred to as the Affinity Dictionary of keys.

Key Surrogate: The code of a key and the classification numbers of cells associated with the key.

Key-to-node Directory: A classification schedule, in which keys are ordered alphabetically and with each key is associated the classification numbers of the nodes of the hierarchical classification tree which contain the key. Each list of classification numbers constitutes a prefix language.

Node-to-key Directory: A classification schedule which contains for each classification number a list of keys ordered alphabetically, which appear at the node (of the hierarchical classification tree) with the classification number in question. The directory is ordered by classification numbers so that the ordered set of classifcation numbers constitutes a prefix language.

Prefix Language: A linear sequential language (function) for the designation of trees. The order from left to right (top to bottom) in which the nodes appear in the prefix language is the order in which a rat seeking the last end would first meet the nodes of the tree as a maze, if the rat could remember that it had already been at a node and preferred to keep to its right and to continue down.

Retrieval: The process of searching and identifying of specific items which contain the desired data being sought.

Sentence Delimiter: Any character which signals the beginning or ending of a sentence in text.

Standard Phrase: A composition of words with certain syntactic and structural dependency.

Term Discrimination Analysis: The process of discriminating and rejecting candidate index terms which do not contribute to a satisfactory classification.

Term Token: An occurrence of term type.

Term Type: A unique value or representation that defines a term.

Terminal Node: A node of the (hierarchical) classification tree, at which no further partitioning takes place.

Text Item: Any aggregate of information in English Language

Total Frequency of Term Type: The total number of occurrences of the term type in the data-base.

User Specified Phrase: A sequence of specific words in a prescribed order.

Word Delimiter: Any character which signals the beginning or ending of a word in text.

APPENDIX B

CLASSIFICATION ALGORITHM

The classification algorithm has been originally devised by Dr. Lefkovitz, and slightly modified by the author. The algorithm applied on a given node to form any N (stratification number for the tree) sibling nodes consists of a three-pass process. Each of these passes is described below.

Pass 1

This pass partitions the key vocabulary of a node into N non-exclusive groups by adding the keys of each item, one at a time, to one of the N groups as follows:

1. Number the resulting groups as: 1, 2, . . . , N. Initially, the collection of item surrogates is positioned at the root node of the tree.
2. The next item description, that is, the set of keys assigned to the item, is read from the collection. The group which contains the most keys of the item is found and denoted as "i". If there are two or more such groups, denote the one with the fewest distinct keys as group i. If there are still two or more groups, arbitrarily select the one with the lowest group number as group i.
3. Let the number of keys in group i be denoted by n_i and the number of keys assigned to the item but not in group i, be denoted by a_i. The following criterion is tested

   $n_i + a_i \leq n_j + a_j + E \qquad j = 1, N$

   where E denotes a sensitivity factor. If the criterion holds, then the keys of the item are added to group i. If not, the keys are

added to group k, where

$n_k + a_k \leq n_j + a_j + E \qquad j = 1, N$

Now a test is made to determine if at least two groups contain some keys. Because if only one group contains all the keys, the classification process is endless since each partition will result in one group containing all items. This situation could occur when the items in the node being partitioned have few keys per item and there is a large number of common keys between items. If the situation does arise, where only one group contains keys, the node is artificially partitioned by first dividing the keys in the one group into N equal groups and then restarting PASS 1, with each of the N groups containing some keys, whereas PASS 1 in a normal partition would start with all groups being empty.

Pass 2

At the start of this pass, no items have been assigned to any groups; only the keys of the items have been assigned. At the end of the pass, those items whose keys appear in only one group, are assigned to the group, while those whose keys appear in more than one group are counted as redundancies and are written to an intermediate file for processing in PASS 3.

Items whose keys appear in only one group are assigned to the group by:

1. Flagging the keys of the item in that group.
2. The item is written to one of the N scratch files numbered coordinately with the key groups so that an item whose keys appear in group i, is written to temporary scratch file i.

Pass 3

If there are no redundancies, the partitioning is complete. However, if

there are redundancies, PASS 3 is performed to minimize them as follows:

1. Each item on the intermediate file is read and assigned to the group which has all the keys of the item flagged. The item is assigned to the first group with all of the item's keys flagged and no other groups are checked.
2. If no group contains all flagged keys of the item, then only those groups which contain all keys of the item, are considered. Of these groups, the group that has the most keys of the item, is flagged. If there is more than one, the group with the lowest group number is selected. The item is assigned to the chosen group by writing the item on the appropriate scratch file, and flagging the keys of the item not yet flagged.

Now a test is made to determine if at least two groups contain some items. Because if only one group contains all the items, the classification process is endless since each partition will result in one group containing all items. This situation could occur, if the key partitioning has been done on an artificial basis. If the situation does arise, where only one group contains items, the node is artificially partitioned by dividing the items in the one group into N equal subgroups and then assigning each subgroup of items to the respective sibling group. At the end of PASS 3, all items will have been assigned to groups, indicating that the partition is complete. Now each subgroup represents a new independent group that may be further repartitioned. However, each subgroup is tested to determine if its size, that is, the number of items assigned to it, is under or equal to the "desired group size" specified by the user. If it is, the subgroup is not further partitioned and constitutes a terminal node, whose classification number is in turn

assigned to all items in the node. If not, the algorithm is applied on the subgroup in question.

University of Pennsylvania
THE MOORE SCHOOL OF ELECTRICAL ENGINEERING

Technical Report

TOS: A TEXT ORGANIZING SYSTEM

VOL. II

APPENDIX A - USER GUIDE
APPENDIX B - PROGRAMMER
APPENDIX C - PROGRAM DESCRIPTIONS AND FLOW CHARTS
APPENDIX D - GLOSSARY

by

Kemal Koymen

Project Supervisor
Noah S. Prywes

August 1974

Prepared for the
Office of Naval Research
Information Systems
Arlington, Va. 22217

under

Contract N00014-67-A-0216-0014
Project No. NR 049-153

Moore School Report # 75-01

SUBJECT INDEX TO APPENDIXES

TABLE OF CONTENTS
VOL. II - APPENDIXES

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

APPENDIX A

USER GUIDE

A.1. General

The purpose of this guide is to provide instructions for the indexing and classification system, when used with the UNIVAC Spectra 70/46 Computer. (A revised user guide would be needed for use with other computers.)

General information is provided, in this introductory section, for execution of indexing and classification programs.

Sections A.2 and A.3 contain instructions for particular programs respectively. These instructions are presented in the order they should be performed by the user.

Chapter III contains description of the indexing and classification system. A brief description of the system is given below.

Figure A.1 shows gross information flow in the system. The input to the system is a data-base consisting of text items in English on a computer-readable media in a standard format. The products, or outputs, of the system are four directories and the data-base, rearranged in accordance with classification numbers assigned to the text items. The process is monitored by the user, who can receive reports and oversee the progress, through his use of a time-sharing terminal. The major interactions of the user are shown in figure A.1. Each box, in figure A.1, represents a gross process. Table A.1 lists for each gross process system functions which constitute the process. For those system functions requiring user interactions, the user actions are also indicated. Note that for each user interaction from the terminal illustrated in

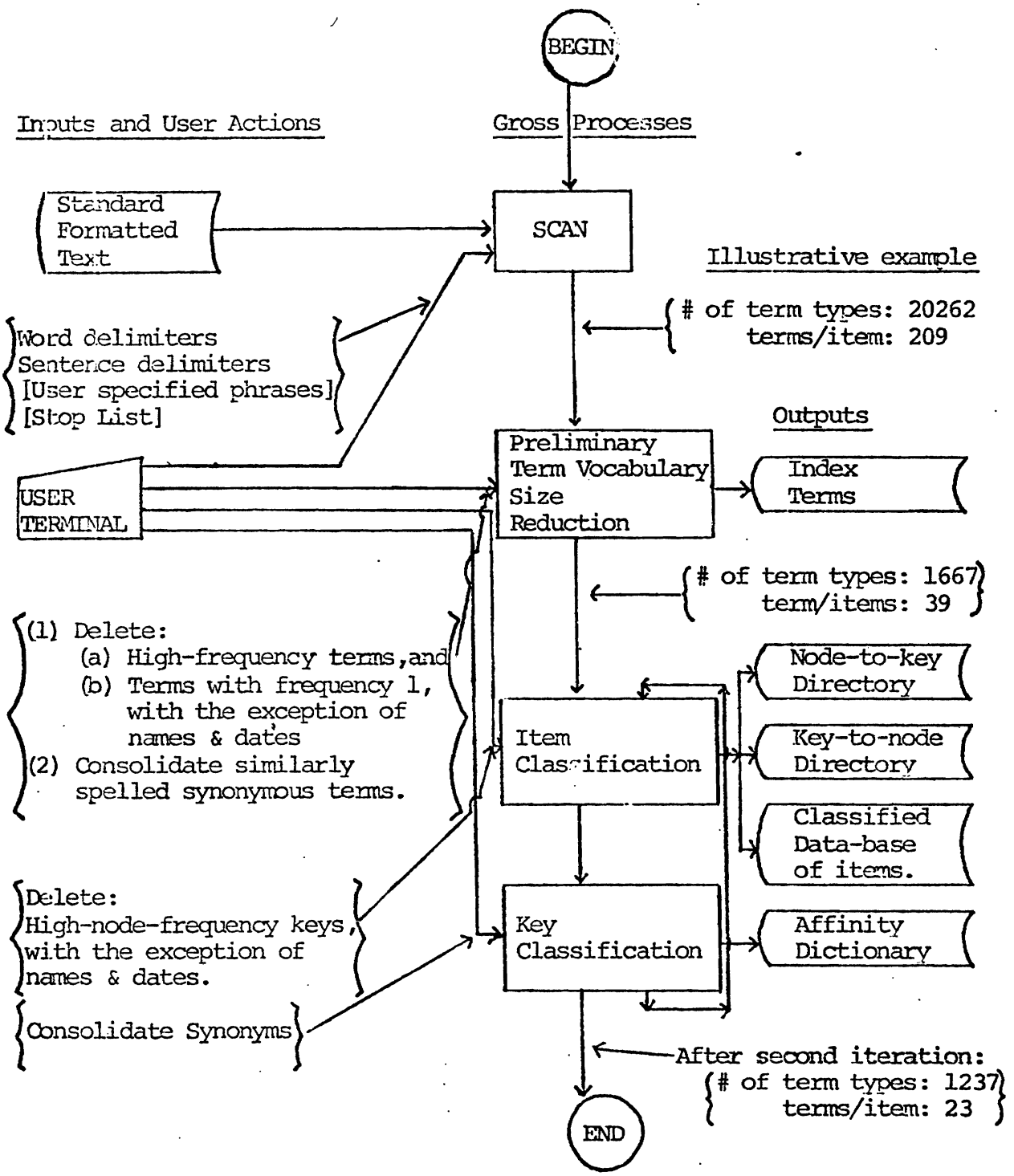


Figure A.1

User Actions, Gross Processes, Inputs and Outputs in the Indexing and Classification of Text Items.

STEP NAME	GROSS PROCESS	SYSTEM FUNCTIONS	USER ACTIONS
INDEXING	PRELIMINARY TERM VOCABULARY SIZE REDUCTION — SCAN	Extraction of words and phrases: Automatic phrase generation. Refinement of phrases. Merge of word and phrase tokens. Production of in-document term types.	1. Enter word and sentence delimiters. [2.] Enter user specified phrases. [3.] Enter stop list.
	PRELIMINARY TERM VOCABULARY SIZE REDUCTION	Reducing the term vocabulary size: Generation of item surrogates.	1. Examine the Summary Report, and Delete: (a) High-frequency terms, and (b) Terms with frequency = 1, with the exception of names and dates. 2. Examine similarly spelled terms and consolidate synonymous terms.
CLASSIFICATION	ITEM-CLASSIFICATION	Item classification Hierarchical classification tree. Node-to-key and Key-to-node Directories: Classified data-base of items.	Examine the report containing the keys ordered by respective node frequencies, and delete high-node-frequency (≥ 15 or 10) keys, with the exception of names & dates.
	KEY-CLASSIFICATION	Generation of key surrogates. Key classification.	
		Affinity Dictionary:	Examine the Affinity Dictionary, locate and consolidate synonyms.

Table A.1

Steps in the Indexing and Classification of Text Items

figure A.1, there is a user action described in table A.1. References to figure A.1 and table A.1 should be made in reading the descriptions of these functions below.

A data-base of 425 text items in world affairs taken from issues of Time magazine published in 1963 has been processed and is used here to illustrate the functions of the system. This is the same collection that has been used by the SMART document retrieval system (Sa 73-10). Some statistics from processing of this illustrative data-base are shown in figure A.1 to indicate the progress during the processing. The computer used for the system operation is a UNIVAC Spectra 70-46, operating with the Univac Virtual Memory Operating System (VMOS) and the Univac Data Management System (DMS). The functions are programmed in Univac version of FORTRAN IV.

As indicated in table A.1, the entire process is divided into two parts: indexing and classification.

In the indexing, candidate index terms are extracted from the text items. Many terms are rejected from further candidacy through the interaction with the user on an on-line time sharing basis. The final output of the indexing process is a directory which contains the remaining index words.

The first part of the indexing process, SCAN, starts with the extraction of candidate index terms from the text items. These terms are words, user-specified phrases and standard phrases, which are extracted from text in a single pass guided by user-terminal, as follows.

A word is defined to be any string of characters set off by word delimiters on either side. Word delimiters are defined and entered by

the user. A word delimiter may be any character which signals the beginning or ending of a word in text. Blanks, left and right parentheses are typically indicated as word delimiters.

A user-specified phrase is defined as a sequence of specific words in a prescribed order. The component words and their relative orderings form a phrase dictionary, which is entered through the user-terminal.

A standard phrase is defined as constituting a composition of words with certain syntactic and structural dependency. Examples of standard phrases are: "APRIL 1, 1974," "U.S. FORCES," "Moore School of E.E.," etc.

In phrase analysis, sentence boundaries are recognized by sentence delimiters, which are defined and entered by the user. A sentence delimiter may be any character which signals the beginning or ending of a sentence in text. For instance, period is typically indicated as a sentence delimiter.

During SCAN, many high-usage words specified by the user in a stop list are rejected and the remaining words are automatically reduced to word stems.

Next, the automatic generation of candidate index phrases, and their merging with previously extracted terms takes place. If the user elects, low-frequency phrases may also be decomposed, into subphrases or words which occur more frequently. These processes are optional and come into action in response to commands from the user. In the final step of SCAN, in-document remaining term types are produced by consolidating multi-occurrences of terms within each item.

Normally, the initial average number of candidate index terms per item and the initial number of candidate index term types are far too

large, and they both have to be greatly reduced for the effectiveness of the subsequent classification and information retrieval. For instance, figure A.1 shows the two statistics of an average of 209 candidate index terms per item and a total of 20262 index term types for the illustrative data-base.

To let the user perform reduction in the term vocabulary size, several reports are provided to the user. These reports are derived in the indexing process, or the subsequent classification process. Preliminary term vocabulary size reduction is performed on the basis of the reports derived in the indexing process only.

As indicated in table A.1, two reports are generated to aid the user in the preliminary term vocabulary size reduction. The first one, Summary Report, contains statistics on frequency distribution of terms. This report informs the user of the impact of actions that he can elect to take, advising him of percentages of reduction in vocabulary size or average number of terms per item resulting from such actions. By specifying frequency limits the user can perform deletion of terms on a mass-basis. The user thus can elect deletion of candidate index terms with frequency 1, or with a high item or total frequency. He can elect to except from the deletion names and dates. The deleted terms are considered to be inefficient in the subsequent classification.

The second report contains lists of terms in which similarly spelled terms are brought together and arranged in groups. This information is useful so as to locate and correct the errors, and find terms which can be consolidated. (Another method of consolidation of terms is applied later in conjuction with the Affinity Dictionary which

is described below). As shown in figure A.1, for the illustrative database, the average number of terms per item and the number of term types have been reduced by such processes from 209 and 20262 to 39 and 1667 respectively.

The result of the preliminary term vocabulary size reduction is a directory of remaining index terms. In this directory, with each index term, are associated identification numbers of the text items and sentences that contain the term.

Next, the remaining index terms are assigned identification codes, and these codes and the identification numbers of items to which the respective terms have been assigned as index terms, are reformatted to be directly acceptable by the classification process. An item identification number and its respective codes are referred to as an item surrogate.

Normally, the indexing process is followed by the classification process. Classification is essentially the result of an attempt to organize a set of objects in a systematic fashion. As indicated in table A.1, the classification processes serve for two purposes: item and key classification.

The objective of item classification is to group alike items together into cells. Cells are similar to the shelves in a library, where sets of alike objects are stored together. Likeness of items is determinable and measurable by the use of common index terms in the respective items. The classification algorithm used here is considered to be of a "divisive" type. It forms a tree by successively dividing the collection of items, on the basis of co-occurrences of index terms

assigned to the items, until desired sized groups are generated. Each node of the tree created in this manner is associated with an exclusive sub-set of the collection of items. The node can also be considered to have associated with a non-exclusive sub-set of the term vocabulary, consisting of the union of terms used to index the respective items. Each node can be assigned a classification number based on its position in the tree. A terminal node of the tree also constitutes a cell of items and its classification number is assigned to all items associated with the cell.

Next, a hierarchical classification tree is formed by intersecting term sets of the nodes and assigning the resulting terms to the parent node (next level up the hierarchy), and in turn deleting these resulting terms from the original nodes.

The hierarchical classification tree with terms at the nodes is described in two classification schedules: Key-to-node and Node-to-key Directories. The index terms, after this point, are referred to as keys, since they will be used to indicate retrieval requirements.

The Key-to-node Directory contains for each key a list of classification numbers of the nodes in the hierarchical classification tree, to which the key is assigned. Vice-versa, the Node-to-key Directory gives the same information in an inverted manner. Namely, it contains for each classification number a list of keys, to which the classification number has been assigned.

As indicated in table A.1, one of the products from the item classification process, is a report which contains the keys ordered by respective node frequencies. If the number of nodes associated with a key is

relatively large, this would indicate that the key is not effective in classifying items and consequently can be deleted from the vocabulary (with the exception of names and dates).

The final product of the item classification process is the Classified (rearranged) Data-base, in which the text items are ordered by respective classification numbers assigned to them by the item classification process. This special ordering reflects the "likeness" of items and consequently facilitates the subsequent retrieval of items.

After the item classification process, the keys and the classification numbers of cells to which they were assigned by the item classification process, are reformatted to be directly acceptable by the key classification process. A key and its respective classification numbers are referred to as a key surrogate.

The objective of key classification is to bring"affinitive" keys together. Affinity of keys is determinable and measurable by the use of respective keys in common text items. The classification algorithm forms a tree by successively dividing the collection of keys, on the basis of co-occurrences of item classification numbers (of cells) to which the keys were assigned, until desired sized groups are generated.

The product of the key classification process is an Affinity Dictionary, in which the keys are ordered by respective key classification numbers. This special ordering reflects the "affinity" of keys and consequently enables the user to determine and consolidate synonymous or near synonymous keys. This dictionary also facilitates searching and browsing through the hierarchical classification tree.

As indicated above, the products of the classification processes enable the user to perform further reduction in the term vocabulary size. For instance, for the illustrative data-base, the average number of terms per item and the number of index term types have been reduced by such products from 39 and 1667 to 23 and 1237 respectively. It should be noted that this reduction cycle must be repeated until the quality of the classification is satisfactory. In this respect, for the illustrative data-base, a ceiling of 10 nodes per key was considered to be satisfactory.

A.1.1 Step Sequence Tables

This section contains two tables which summarize the processes in the order they should be performed to index and classify a collection of text items.

Table A.2 contains the sequence of steps needed to index the collection of text items. Some major steps are divided into sub-steps. They perform the tasks indicated by the respective step names. For instance, major step 1, which is the Extraction of Words and Phrases from the text items, consists of five sub-steps of which the fourth one is optional. Input to and output from each program or process are indicated, and the respective references are given in terms of page or section numbers.

Similarly, table A.3 contains the information for the classification process.

MAJOR STEP NUMBER	STEP NAME	SUB-STEP NUMBER	SUB-STEP NAME	INPUT file no.	INPUT file ref. p	INPUT par. ref. p	PROGRAM name	PROGRAM ref. p	OUTPUT file no.	OUTPUT file ref. p	OUTPUT rep. ref.	OUTPUT stat. ref.
1	Extraction of Words and Phrases	1	Create Standard Formatted Text File	Orig. Coll. of Txt Items			User Written Prog.	342	0	244		
		2	Enter Stop List	22	259		UPDATE or EDITOR	321	22	259		
		3	Enter Word and Sentence Delimiters			292	INDEX	291	1	245		294
		[4]*	Enter Phrase Dictionary			292	INDEX	291	1	245		294
		5	Find All Word Tokens	0 22		292	INDEX	291	1	245		294
2	Automatic Phrase Generation	1	Find All Phrase Tokens	1	245	296	PHRGEN	295	3	247		296
3	Refinement of Phrases	1	Fixed Length of Phrases	3	247	297	VARFIX	296	4	248		
		2	Sort Phrases Alpha-betically	4	248	297	SORT	275	19	248		
		3	Summarize Phrases By Types	19	248	298-9	UNWRDS	298	13 16 23	256-7 258 261		299

* This step is optional

Table A.2
Sequence of Steps to Run the Indexing Programs

MAJOR STEP NUMBER	STEP NAME	SUB-STEP NUMBER	SUB-STEP NAME	INPUT file no.	INPUT file ref.p.	INPUT par. ref.p.	PROGRAM name	PROGRAM ref.p.	OUTPUT file no.	OUTPUT file ref.p.	OUTPUT rep. ref.p.	OUTPUT stat. ref.p.
3	Refinement of Phrases	4	Sort Phrases By Frequency	16	258	299	SORT	275	15	258		
		5	Generate Summary Report	15	258	300-1	FRQWCN	300			301	301
		6	On-line Display of Text	0	244	302	STDFIL	302	0	244		
		7	Display Information	0,24 13,23 15	244,261 256-7, 260,258	303 304-5	PRTON PRTOFF	303				
		8	Refine Phrases	4 13,23	248 256-7 260	306	UPDATE	305				
4	Merge of Word and Phrase Tokens	1*	Sort Phrases by Document Numbers	4	248	306	SORT	275	4	248		
		2	Merge Word and Phrase Tokens	1 3 or 4	245 247-8	307	MRWRPH	306	5	249		307
5	Consolidation of Duplicate Terms Within Each Document	1**	Fixed Length of Words	1	245	308	VARFIX	307	2	246	.	
		2	Sort Terms by Document Numbers	5 or 2 or 11	249 246 254	309	SORT	275	17 or 18 or 21	249 246 254		

* This step is performed iff major step 3 is performed.
** This step is performed iff major steps 2 through 4 are not performed.

Table A.2 (continued)

MAJOR STEP NUMBER	STEP NAME	SUB-STEP NUMBER	SUB-STEP NAME	INPUT file no.	INPUT file ref.p	INPUT par. ref.p	PROGRAM name	PROGRAM ref.p	OUTPUT file no.	OUTPUT file ref.p	OUTPUT rep. ref.p	OUTPUT stat. ref.p
5		3	Summarize In-document Terms by Types	17 or 18 or 21	249 246 254	309-10	ELDID	309	6	250		310
6	Reducing the Key Vocabulary Size	1	Sort Terms Alpha-betically	6	250	311	SORT	275	7	250		
		2	Summarize Term Types	7	250	311-12	UNWRDS	311	11 12 16 23	254 255-56 258 260		312
		3	Sort Terms by Frequency	16	258	314	SORT	275	14	258		
		4	Determine Similarly Spelled Terms	12,23	255-56 260	312-13	ADJCMP	312			313	
		5	Generate Summary Report	14	258	300-1	FRQVCN	300			301	301
		[6]*	Compare Old and New Vocabularies	12(old) 12(new)	255-56	314	COMDIR	314			315	
		7	Display Information	0,24 12,23 14	244,261 255-56 260,258	316 317-19	PRTON PRTOFF	316 317				
		8	Refine Terms	11 12,23	254 255-56 260	319	UPDATE	319				

* This step can be performed, if all the text items have been previously indexed.

Table A.2 (continued)

MAJOR STEP NUMBER	STEP NAME	SUB-STEP NUMBER	SUB-STEP NAME	INPUT file no.	INPUT file ref.p	INPUT par. ref.p	PROGRAM name	PROGRAM ref.p	OUTPUT file no.	OUTPUT file ref.p	OUTPUT rep. ref.p	OUTPUT stat. ref.p
7	Generate Document Surrogates	1	Sort Terms Alpha-betically	6	250	311	SORT	275	7	250		
		2	Summarize Keys By Types	7	250	322	UNWRDS	321	8 or 9 10 10 (w/o low frq)	251, 52 253		312
		3	Sort Keys By Document Numbers	10 [w/o low frq]	253	309	SORT	275	20	253		
		4	Generate Document Surrogates	20 V	253 264	322	DOCSUR	322	I	262		

Table A.2 (continued)

MAJOR STEP NUMBER	STEP NAME	SUB-STEP NUMBER	SUB-STEP NAME	INPUT file no.	INPUT file ref.p	INPUT par. ref.p	PROGRAM name	PROGRAM ref.p	OUTPUT file no.	OUTPUT file ref.p	OUTPUT rep. ref.p	OUTPUT stat. ref.p
1	Document Classification	1	Classify Documents	I	262	323-24	CLASFY	323	II	263	324	325
2	Complete Sets of Keys Within Cells	1	Sort File By Document Numbers	II	263	325	SORT	275	III	263		
		2	Assign In-document Key Types To Categories	III 20	263 253	325	GNTRCD	325	IV	264		
		3	Sort File By Cell Number	IV	264	327	SORT	275	VI	264		
		4	Produce Complete Set of Keys Within a Cell	VI	264	327	GNFKFL	327	VII	265		327
3	Hierarchical Classification Tree	1	Sort File By Cell Number	VII	265	328	SORT	275	VIII	265		
		2	Generate The Tree	VIII	265	328-29	TREE	328	IX	266		329

Table A.3
Sequence of Steps to Run the Classification Programs

MAJOR STEP NUMBER	STEP NAME	SUB-STEP NUMBER	SUB-STEP NAME	INPUT file no.	INPUT file ref.p	INPUT par. ref.p	PROGRAM name	PROGRAM ref.p	OUTPUT file no.	OUTPUT file ref.p	OUTPUT rep. ref.p	OUTPUT stat. ref.p
4	Node-to-key and Key-to-node Directories	1	Sort File By Node Number	IX	266	329	SORT	275	X	266		
		2	Generate Node-to-key Directory	8 or 9 X II	251-52 266 263	330	NDTOKY	330	XI XII	267 268	331	331
		3	Sort Keys Alpha-betically	XII	268	331	SORT	275	XIII	268		
		4	Generate Key-to-node Directory	XIII	268	332	KYTOND	331	XIV	269	332	332
5	Classified Data-base	1	Generate Classified Data-base	III 20 or 6, 0	263 253,250 244	332-33	MRGCLY	332	XV	270		
		2	Sort File By Cell Number	XV	270	333	SORT	275	XVI	270		
		3	Produce Classified Data-base Report	XV	270	334	PRTCLF	333	temp.			
		4	Sort Report By Cell Number	temp.		334	SORT	275			334	

Table A.3 (continued)

MAJOR STEP NUMBER	STEP NAME	SUB-STEP NUMBER	SUB-STEP NAME	INPUT file no.	INPUT file ref.p	INPUT par. ref.p	PROGRAM name	PROGRAM ref.p	OUTPUT file no.	OUTPUT file ref.p	OUTPUT rep. ref.p	OUTPUT stat. ref.p
6	Key Classification	1	Sort File By Codes	IV	264	335	SORT	275	V	266		
		2	Generate Key Surrogates	V	264	322	DOCSUR	322	XVII	262		
		3	Classify Keys	XVII	262	335	CLASFY	335	XVIII	263	324	325
7	Affinity Dictionaries	1	Sort File By Codes	XVIII	263	336	SORT	275	XIX	263		
		2	Generate Dictionary For Retr. System	XIX XIV	263 269	336-37	KTCLND	336	XX XXI	272 273	337	337
		3	Sort File By Cell Number	XXI	273	338	SORT	275	XXII	273		
		4	Generate Dictionary For Manual Use	XXII	273	338-39	PRTCLF	338			339	

Table A.3 (continued)

A.1.2 Program Tables

The indexing part of the system is comprised of fifteen programs which may be run from a remote terminal. These programs and their functions are shown in table A.4. The classification part of the system is comprised of nine programs which may also be run from a remote terminal. These programs and their functions are shown in table A.5.

PROGRAM	SECTION AND PAGE	FUNCTION
INDEX	A.2.2 p.291	Extraction of words and standard and user specified phrases.
PHRGEN	A.2.3 p.295	Automatic phrase generation.
VARFIX	A.2.4 A.2.6.1	Transformation of variable length terms to fixed length terms.
UNWRDS	A.2.4.3 A.2.7.1.2	Production of term or key types.
STDFIL	A.2.8.2 A.2.4.6 p.302	Generation of a system directory to the Standard Formatted Text File.
PRTON	A.2.4.7.1	On-line display of information.
PRTOFF	A.2.7.5.1 A.2.4.7.2	Off-line display of information.
UPDATE	A.2.7.5.2 A.2.4.8 A.2.7.6	(1) To decompose phrases. (2) To add, delete, and change (consolidate and substitute) terms.
DELMER	A.2.7.6.2	To delete and merge document representatives.
MRWRPH	p.320 A.2.5.2	Merging the word and phrase tokens.
ELDID	p.306 A.2.6.3 p.309	Consolidation of multiple occurrences of terms within each document.
ADJCMP	A.2.7.2 p.312	Indication of similarly spelled index terms.
FRQWCN	A.2.4.5 p.300	Generation of Summary Report.
COMDIR	A.2.7.4 p.314	Comparison of the old and new vocabularies of index terms.
DOCSUR	A.2.8.4 p.322	Generating document or key surrogates.

Table A.4

List of Indexing Programs

PROGRAM	SECTION AND PAGE	FUNCTION
CLASFY	A.3.1.1 A.3.2.1.3	Document or key classification.
GNTRCD	A.3.1.3.2 p.325	To assign the in-document key types of each document to its respective catagory.
GNFKFL	A.3.1.3.4 p.327	To generate categories of keys.
TREE	A.3.1.4.2 p.328	To generate Hierarchical Classification Tree.
NDTOKY	A.3.1.5.2 p.330	To generate Node-to-key Directory.
KYTOND	A.3.1.5.6 p.331	To generate Key-to-node Directory.
MRGCLY	A.3.1.6.1 p.332	To generate the Classified Data-base.
PRTCLF	A.3.1.6.3 A.3.2.2.4	(1) To produce a report of the Classified Data-base. (2) To generate an Affinity Dictionary for the user.
KTCLND	A.3.2.2.2 p.336	To generate an Affinity Dictionary for the Retrieval System.

Table A.5

List of Classification Programs

A.1.3 File Description Tables

The input to a program may be one or more files created in a previous step of the process. Table A.6 and table A.7 contain summaries of files which are created in the indexing and classification processes respectively. Detailed description of each file is also given in tables A.8 through A.37.

The user may also have to provide input parameters to control the operation of a program. These parameters can be entered in successive k-bytes (k 80) records from a terminal. The parameters in a record follow the general syntax shown below.

<Record> : : = <parameter> [[Δ] <parameter>] ...

<parameter> : : = Im | nAl

Im : : = m-digit integer
nAl : : = 1 to n characters
Δ : : = one blank character
[ ] : denotes optionality.

Outputs from a program may contain statistics, reports, of files to be input to another program at a later step in the process.

file number	output from program	input to program	reference table	reference page	record form
0 (Standard Formatted Text File)	User Written Program	INDEX STDFIL PRTOFF	A.8	244	V
0 (RA Standard Formatted Text File)	STDFIL	PRTON PRTOFF	A.8	244	V
1	INDEX	PHRGEN MRWRPH VARFIX PRTON PRTOFF	A.9	245	V (≤130)
2	VARFIX	SORT PRTON PRTOFF	A.10	246	F 2(4+MAXTL)
3	PHRGEN	VARFIX MRWRPH PRTON PRTOFF	A.11	247	V (≤132)
4	VARFIX	UPDATE UNWRDS PRTON PRTOFF	A.12	248	F 2(7+MAXTL)
5	MRWRPH	SORT PRTON PRTOFF	A.13	249	F 2(3+MAXTL)

Table A.6

Summary of Indexing Process Files

file number	output from program	input to program	reference		record form
			table	page	
6	ELDID	Retrieval System SORT DELMER PRTON PRTOFF	A.14	250	V
7	SORT	UNWRDS ADJCMP PRTON PRTOFF	A.14	250	V
8	UNWRDS	NDTOKY PRTON PRTOFF	A.15	251	F 2(3+MAXTL)
9	UNWRDS	NDTOKY PRTON PRTOFF	A.16	252	F 2(4+MAXTL)
10	UNWRDS	SORT GNTRCD PRTON PRTOFF	A.17	253	F 2(3+MAXTL)
11	UNWRDS	SORT UPDATE PRTON PRTOFF	A.18	254	V

Table A.6

-CONTINUED-

file number	output from program	input to program	reference table	reference page	record form
12	UNWRDS	UPDATE ADJCMP COMDIR PRTON PRTOFF	A.19 A.20	 255 256	V
13	UNWRDS	UPDATE ADJCMP PRTON PRTOFF	A.20 A.21	256 257	V
14	SORT	FRQWCN PRTON PRTOFF	A.22	258	F 2(6+MAXTL)
15	SORT	FRQWCN PRTON PRTOFF	A.22	258	F 2(6+MAXTL)
16	UNWRDS	SORT PRTON PRTOFF	A.22	258	F 2(6+MAXTL)
17	SORT	ELDID PRTON PRTOFF	A.13	249	V
18	SORT	ELDID PRTON PRTOFF	A.10	246	F 2(4+MAXTL)

Table A.6

-CONTINUED-

file number	output from program	input to program	reference table	reference page	record form
19	SORT	UNWRDS PRTON PRTOFF	A.12	248	F 2(7+MAXTL)
20	SORT	DOCSUR MRGCLY PRTON PRTOFF	A.17	253	F 2(3+MAXTL)
21	SORT	ELDID PRTON PRTOFF	A.18	254	V
22 (stop list)	Created by user	INDEX UPDATE PRTON PRTOFF	A.23	259	F
23 (ISAM Key Directory)	UNWRDS	ADJCMP UPDATE PRTON	A.24	260	F (4)
24 (DN/ISAM Key Directory)	STDFIL	PRTON	A.25	261	F (8)

Notes: MAXTL denotes maximum number of characters per term.
SORT denotes a SORT package.

Table A.6

-CONTINUED-

file number	output from program	input to program	reference table	reference page	record form
I	DOCSUR	CLASFY	A.26	262	V(≤528)
II	CLASFY	SORT	A.27	263	F (8)
III	SORT	GNTRCD MRGCLY	A.27	263	F (8)
IV	GNTRCD	SORT	A.28	264	F (4)
V	SORT	DOCSUR	A.28	264	F (4)
VI	SORT	GNFKFL	A.28	264	F (4)
VII	GNFKFL	SORT	A.29	265	V
VIII	SORT	TREE	A.29	265	V
IX	TREE	SORT	A.30	266	V
X	SORT	NDTOKY	A.30	266	V
XI	NDTOKY	Retrieval System	A.31	267	V
XII	NDTOKY	SORT	A.32	268	V
XIII	SORT	KYTOND	A.32	268	V
XIV	KYTOND	Retrieval System KTCLND	A.33	269	V
XV	MRGCLY	SORT PRTCLF	A.34 A.35	270 271	V
XVI	SORT	Retrieval System	A.34 A.35	270 271	V
XVII	DOCSUR	CLASFY	A.26	262	V(≤528)
XVIII	CLASFY	SORT	A.27	263	F (8)
XIX	SORT	KTCLND	A.27	263	F (8)
XX	KTCLND	Retrieval System	A.36	272	V
XXI	KTCLND	SORT	A.37	273	V
XXII	SORT	PRTCLF	A.37	273	V

Table A.7 Summary of Classification Process Files

field #	length (in bytes)	format	Description
1	5	Zoned decimal (I5)	The length of the third field.
2	6	Zoned decimal (I5)	A sequentially generated number used to identify each document. This is referred to as document identification number.
3	Not more than (L-11)*	EBCDIC characters ('A' format)	Any information that the user wishes to associate with the document may be placed into this field, which is referred to as the text field.

*'L' represents the maximum allowable physical record size.(File 0)

Table A.8

Description of a Record on the (RA) Standard Formatted Text File

field #	length (in bytes)	format	Description
1	2	FPB*	Length of the term (in characters)= LENGTH
2	2* LENGTH	EBCDIC	The term extracted from the text
3	4	FPB	The identification number of the document containing the term.
4	2	FPB	The identification number of the sentence containing the term.
5	2	FPB	The term-address assigned to the term

*FPB denotes "fixed point binary".

Table A.9

Description of a Record on File 1.

field #	length (in bytes)	format	Description
1	2*MAXTL*	EBCDIC	The term extracted from the text.
2	4	FPB	The identification number of the document containing the term.
3	2	FPB	The identification number of the sentence containing the term.
4	2	FPB	The term-address assigned to the term.

*MAXTL denotes the maximum number of characters per term.

Table A.10

Description of a Record on File 2 or 18

field #	length (in bytes)	format	Description
1	2	FPB	Length of the phrase (in characters)= PHRLEN
2	2*PHRLEN	EBCDIC	The phrase generated.
3	4	FPB	The identification number of the document containing the phrase.
4	2	FPB	The identification number of the sentence containing the phrase.
5	2	FPB	The number of phrase components.
6	2	FPB	Phrase-address.

Table A.11

Description of a Record on File 3.

field #	length (in bytes)	format	Description
1	4	FPB	The token code assigned to the phrase token.
2	2*MAXPHR*	EBCDIC	The phrase generated.
3	4	FPB	The identification number of the document containing the phrase token.
4	2	FPB	The identification number of the sentence containing The phrase token.
5	2	FPB	The number of phrase components.
6	2	FPB	Phrase-address.

* MAXPHR denotes maximum number of characters per phrase.

Table A.12

Description of a Record on File 4 or 19

field #	length (in bytes)	format	Description
1	2*MAXKL*	EBCDIC	Term token
2	4	FPB	The identification number of the document containing the term token.
3	2	FPB	The identification number of the sentence containing the term token.

* MAXKL denotes maximum number of characters per term.

Table A.13

Description of a Record on File 5 or 17.

field #	length (in bytes)	format	Description
1	2*MAXTL*	EBCDIC	In-document term type
2	4	FPB	Identification number of the document containing the term.
3	2	FPB	In-document frequency of the term.
4	2	FPB	Identification number of the sentence containing the term.
[5]	2	FPB	Identification number of the sentence containing the term.
.	.	.	" "
.	.	.	" "
.	.	.	" "

* MAXTL denotes maximum number of characters per term.

Table A.14

Description of a Record on File 6 or 7

field #	length (in bytes)	format	Description
1	2*MAXKL*	EBCDIC	Key type
2	2	FPB	Type code assigned to the key type.
3	4	FPB	Total frequency of the key.

* MAXKL denotes maximum number of characters per key.

Table A.15

Description of a Record on File 8.

field #	length (in bytes)	format	Description
1	2*MAXKL*	EBCDIC	Key type.
2	2	FPB	Type code assigned to the key type.
3	4	FPB	Total frequency of the key.
4	2	FPB	Key indicator: 0- The key is excluded from classification process. However, it will be added to the classification-produced directories. 1- The key is included in classification process.

* MAXKL denotes maximum number of characters per key.

Table A.16

Description of a Record on File 9.

field #	length (in bytes)	format	Description
1	2*MAXKL*	EBCDIC	In-document key type.
2	4	FPB	Identification number of the document containing the key.
3	2	FPB	Type code assigned to the key type.

* MAXKL denotes maximum number of characters per key.

Table A.17

Description of a Record on File 10 or 20.

field #	length (in bytes)	format	Description
1	4	FPB	The token code assigned to the in-document term type.
2	2*MAXKL *	EBCDIC	In-document term type.
3	4	FPB	Identification number of the document containing the in-document term type.
4	2	FPB	In-document frequency of the term.
5	2	FPB	Identification number of the sentence containing the in-document term type.
[6]	2	FPB	" "
.	.	.	" "
.	.	.	" "
.	.	.	" "

* MAXKL denotes maximum number of characters per term.

Table A.18

Description of a Record on File 11 or 21.

field #	length (in bytes)	format	Description
1	2	FPB	Record Indicator: 0- more of same type of record to follow. 1- record of different type to follow.
2	4	FPB	Token code assigned to the in-document term type.
3	4	FPB	Identification number of the document containing the in-document term type.
4	2	FPB	In-document frequency of the term.
5	2	FPB	Identification number of the sentence containing the in-document term type.
[6]	2	FPB	" "
.	.	.	" "
.	.	.	" "
.	.	.	" "

Table A.19

Description of the Token Code Record on File 12

field #	length (in bytes)	format	Description
1	4	FPB	Type code assigned to the term type.
2	2*MAXKL*	EBCDIC	The term type.
3	4	FPB	Document frequency of the term.
4	4	FPB	Total frequency of the term.

*MAXKL denotes the maximum number of characters per term.

Table A.20

Description of the Type Code Record on File 12 or 13

field #	length (in bytes)	format	Description
1	2	FPB	Record Indicator: 0- more of same type of record to follow. 1- record of different type to follow.
2	4	FPB	The token code assigned to the phrase token.
3	4	FPB	The identification number of the document containing the phrase token.
4	2	FPB	The identification number of the sentence containing the phrase token.

Table A.21

Description of the Token Code Record on File 13.

field #	length (in bytes)	format	Description
1	4	FPB	Type code assigned to the term type.
2	2*MAXTL*	EBCDIC	Term type.
3	4	FPB	Document frequency of the term.
4	4	FPB	Total frequency of the term.

* MAXTL denotes maximum number of characters per term.

Table A.22

Description of a Record on File 14 or 15 or 16

field #	length (in bytes)	format	Description
1	MAXWL*	EBCDIC characters ('A' format)	A word on the user's stop list. This should be left justified.
2	MAXWL	EDCDIC characters	" "
.	.	.	" "
.	.	.	" "
.	.	.	" "

*MAXWL denotes maximum number of characters per word.

Table A.23

Description of the Stop List (File 22)

field #	length (in bytes)	format	Description
1	4	FPB	Pointer to the first of the term phrase) token code records associated with the first term (phrase) type.
.	.	.	" "
.	.	.	" "
N	4	FPB	Pointer to the first of the term (phrase) token code records associated with the Nth term (phrase) type.
.	.	.	" "

Table A.24

Description of the ISAM Key Directory (File 23)

record #	field #	length (in bytes)	format	Description
1	1	4	FPB*	Number of documents on the standard formatted text file.
	2	4	FPB	Unused
N+1 (N>0)	1	4	FPB	Identification number of the Nth document on the standard formatted text file.
	2	4	FPB	Pointer to the Nth document on the standard formatted text file.

Table A.25

Description of the DN/ISAM Key Directory (File 24)

field #	length (in bytes)	format	Description
1	4	FPB	The identification number of the document (or type code assigned to the key).
2	2	FPB	Number of Codes (representing keys or cell numbers) in the surrogate (≤ 250)= NDKY
3	4	FPB	Number of surrogates in this group.
4	2	FPB	Terminal node (cell) flag 0- not a terminal node. 1- terminal node.
5	2	FPB	Level in which terminal node occurred.
6	2	FPB	Node number of terminal node.
7	2	FPB	First code in surrogate.
6+NDKY	2	FPB	Last code in surrogate.

Record Length = (NDKY*2 + 16)

Notes:

1. The third field of first record initially contains the actual number of surrogates on the out put file.
2. The fourth and following fields are not used by Program DOCSUR.

Table A.26

Description of a Record on the Document (or Key) Surrogate Directory (File I or XVII)

field #	length (in bytes)	format	Description
1	4	FPB	Identification number of the document of type code assigned to the key type.
2	2	FPB	Document of key classification number.
3	2	FPB	Number of codes in the document (or key) surrogate.

Table A.27

Description of a Record on File II or III or XVIII or XIX

field #	length (in bytes)	format	Description
1	2	FPB	Document classification number of the cell.
2	2	FPB	Type code (in-document key type) in the cell.

Table A.28

Description of a Record on File IV or V or VI

record type	field #	length (in bytes)	format	Description
header-record	1	2	FPB	Document classification number of the cell.
	2	2	FPB	Sequence number of the record (associated with the terminal node).
	3	2	FPB	Number of keys in the terminal node.
key-record	1	2	FPB	Document classification number of the cell.
	2	2	FPB	Sequence number of the record.
	3	2	FPB	Type code (in-document key type).
	4	2	FPB	In-cell frequency of the key.
	5	2	FPB	Type code.
	6	2	FPB	In-cell frequency
	.	.	.	.
	.	.	.	.
	.	.	.	.
	.	.	.	.
	maximum allowed			

Table A.29

Description of File VII or VIII

field #	Length (in bytes)	format	Description
1	2	FPB	Length of text to follow.
2	4	FPB	Document identification number
3	2	FPB	Document classification number
4	2	FPB	Record indicator: 0- More of same type of record to follow. 1- Record of different type to follow.
5	2	FPB	Sequence number of record.
6	2	EBCDIC	Text of Document
.	.	.	"
.	.	.	"
.	.	.	"
2042*			Up to 2036 characters in a record.

* Maximum Record Length is 4086 bytes.

Table A.34

Description of the Text-Record on File XV or XVI

field #	length (in bytes)	format	Description
1	2	FPB	Number of keys in record.
2	4	FPB	Document identification number.
3	2	FPB	Document classification number.
4	2	FPB	Record indicator: 0- more of same type of record to follow. 1- record of different type to follow.
5	2	FPB	Sequence number of record.
6	2*MAXKL*	EBCDIC	Key.
.			.
.			.
.			.
Maximum Allowed			Up to maximum number allowed in a record.

*MAXKL denotes maximum number of characters per key.

Table A.35

Description of the Key-Record on File XV or XVI.

record type	field #	length (in bytes)	format	Description
header-record	1	2*MAXKL*	EBCDIC	Key type.
	2	2	FPB	Key classification number assigned to the key type.
	3	2	FPB	Node frequency of the key.
key-record	1	2	FPB	Document classification number.
	2	2	FPB	Number of the following fields plus one.
	3	2	FPB	In-cell frequency of the key.
	[4]	2	FPB	" "
	.	.	.	" "
	.	.	.	" "
	.	.	.	" "

* MAXKL denotes maximum number of characters per key.

Table A.36

Description of File XX

field #	length (in bytes)	format	Description
1	2*MAXKL	EBCDIC	Key type. ('MAXKL' characters)
2	2	FPB	The first digit in the canonical key classification number.
3	2	FPB	The second digit in the canonical key classification number.
.	.	.	
.	.	.	
K+1	2	FPB	The last digit in the canonical key classification number.
K+2	2	FPB	Zero
K+3	2	FPB	Zero
.	.	.	.
.	.	.	.
Level2+2	2	FPB	Number of digits (K) in the key classification number.
Level2+3	2	FPB	The first digit in the first canonical document classification number.
	2	FPB	Number of digits in the last canonical document classification number.

Notes: (1) LEVEL 1 specifies the number of levels in the document classification tree.
(2) LEVEL 2 specifies the number of levels in the key classification tree.

Table A.37 Description of a Record on File XXI or XXII

A.1.4 FILE Command

Any file used by a program must be predefined by a FILE command which makes it accessible through the Data Management System (DMS) access methods. The user must provide the required information about the file through the parameters in the corresponding FILE command. The programs may require the user to specify one to six parameters in a FILE command. However, other additional parameters may also be required by the DMS. The general syntax of a FILE command is given below, where parameters are not positional and may be specified in any order.

/FILE <name>,<parameter> [,<parameter>]...

<name> :: = A name under which the file is cataloged.

<parameter> :: = A FILE command parameter.

The FILE command parameters that may be required by a program are listed and described below.

LINK = DSET nn — nn is a two digit reference number associated with the file.

OPEN = {INPUT, OUTPUT, INOUT, OUTIN, EXTEND} — specifies the manner in which the file is to be opened.
INPUT - in put processing only.
OUTPUT - output processing only
INOUT - input processing with the ability to write the file.
OUTIN - output processing with the ability to retrieve records from the file.
EXTEND - output processing only at the end of the file.

FCBTYPE = {SAM, ISAM, BTAM} — specifies the access method in processing the file.
SAM - Sequential Access Method
ISAM - Indexed Sequential Access Method
BTAM - Basic Tape Access Method

RECFORM = {F / V}	specifies the record form. F - a file of fixed-length records V - a file of variable-length records
RECSIZE = M	Defines the record size when RECFORM = F and defines the maximum allowable size of records when RECFORM = V
BLKSIZE = k	defines the largest block to be read or/and written to the file. It establishes the minimum bugger size for the DMS.

As an example, the FILE command:

```
/FILE file-0, LINK = DSET 21, RECFORM = V,
/BLKSIZE = 4096, OPEN = INPUT, CBTYPE = BTAM
```

indicates that the file cataloged under the name file-0 will be used only for input processing through the BTAM. Furthermore, it also indicates that the record form is variable-length and the block size is 4096.

A.1.5 SORT Package

Some programs may require the user to prearrange the data on the input file(s) in a specified order. Since different generalized SORT utility software are available at most computer installations, a sort package is not included in the indexing and classification system.

A sort application can be achieved by invoking the SORT routine and then providing the required statements for which the general syntax and interpretation is given below.

```
/EXEC [UTE]     SORT
SORT FIELDS = (<field> [,<field>]...)
              [, FORMAT = <format>]

[RECORD LENGTH = (<len>), TYPE = V]
END
```

<field> : Describes a control field in a sort application. Each character of a term in a record should be specified as a separate two-byte field. Up to 255 fields may be specified. The significance of a field is determined by its position in the SORT FIELDS statement, thus, the most significant control field is the one first specified. The control field has the following format:

<position ,<length>,<sequence>[,<format-1>]

<position> : Specifies the beginning of a control field relative to the beginning of the record. For variable-length records, the logical record includes the four-byte record length indicator. Also, the logical record includes a four-byte GCW (Green Control Word). The latter is prefixed by the Univac 70/46 FORTRAN to every unformatted (binary) record.

<length> : Specifies the length of the control field.

<sequence> : Specifies how the control field is to be ordered through one of the following values:

A - Ascending Sequence

D - Descending Sequence

<format-1> : This optional operand should only be specified if the fields vary in format. (Refer to the following paragraph.)

<format> : Specifies the data format of the control field, and can be one of the following two character combinations:

FI - Fixed-Point

CH - Character

<len> : Specifies the (maximum) length of input records. The RECORD LENGTH statement is needed, if the input records are of variable length.

The user must also predefine the unsorted and sorted files by FILE commands. In the FILE command defining the unsorted file, LINK = SORTIN should be specified as one of the FILE command parameters. In the FILE command defining the sorted file, LINK = SORTOUT should be specified as one of the parameters. Other FILE command parameters may also be specified as required.

A.1.6 Running A Program

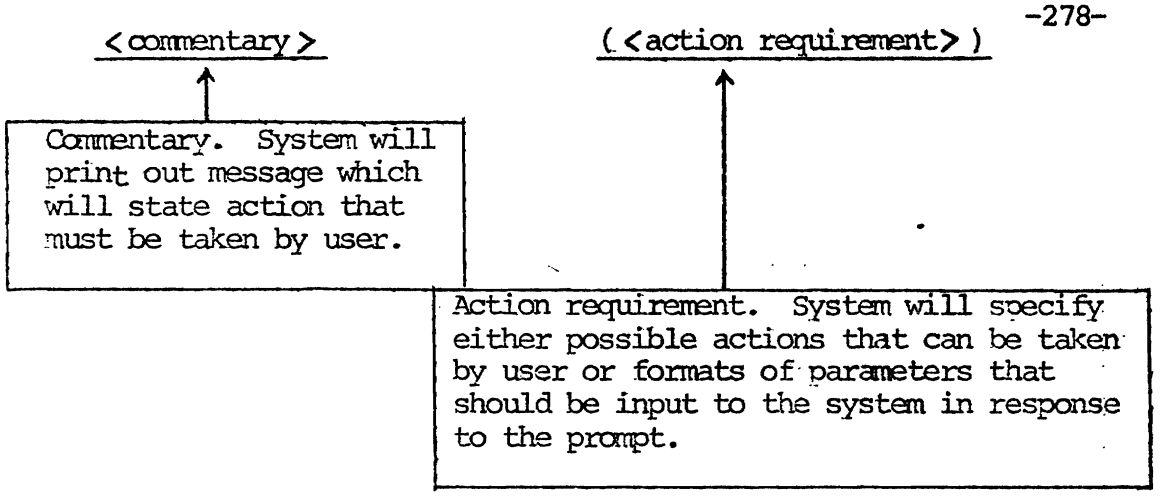
EXECUTE command enables the user to invoke a program from a remote terminal or background task. The command can be issued as follows:

/EXEC [UTE] <Module>

where <Module> denotes the name of the load module of the program. The module name may consist of any string of one to eight characters specified by the programmer.

A.1.7 Prompts and Error Messages

Messages to and from the system during operations of some programs consist of either prompts or error messages. Error messages are listed in table A.45. Prompts are listed in table A.38. All prompts follow the same general syntax which is shown below.



<action requirement> may consist of a list of fields, as described below.

<action requirement> :: = <parameter> [,1X, <parameter>]...

<parameter> :: = Im | kAl | <letter>

<letter> :: = P | T | Y | N | F

m :: = 1 | 2 | 3 | 4 | 5 | 6

k :: = a positive integer

Field Interpretation Table

FIELD NAME	DESCRIPTION
Im	m-digit integer
1X	any character
kAl	one to k alphanumeric characters (left justified)
P	specifies the line-printer
T	specifies a teletype
Y	indicates 'YES'
N	indicates 'NO'
F	indicates the first of records (or fields) which are candidates for display.

PROMPT	MEANING
Specify the file (I2)	The file number of the file to be displayed or updated.
Specify the maximum key length (I2)	Maximum number of characters per key.*
Specify the document number (I6)	The identification number of the document. This is needed when document is being displayed or a key is being added to the document.
Sentence numbers (I4)	The identification numbers of the sentences to be displayed. Each number should be specified in a separate line. Specification of zero signals the end of input in response to the prompt. If zero is specified at first, the whole document is displayed.
Specify the text letter case parameter and the sentence deliniter (I1, 1X, A1)	See page 293
Give the number of word delimiters to be specified (I2)	See page 293
Specify the word delimiters (30A1)	See page 293

* See page 293 for the interpretation of maximum key length, when displaying file-0.

Table A.38

List of Prompts

PROMPT	MEANING
Specify the output device and the line size (T or P, 1X, I3)	Line size specifies the length of a printed line in characters. Note: (1) 'Line size' ≤ 130 for the line-printer. (2) 'Line size' ≤ 62 for the teletype. Line size specification is optional and the followings are default values. (1) 'Line size' = 125 for the line-printer. (2) 'Line size' = 60 for the teletype.
Do you want the sentence numbers printed (Y,N,F)	F means the first of the sentence identification numbers in each record will be printed.
Range of {display / deletion / change} codes for (I6,1X,I6,1X,A1)	A range of codes. If a non-blank character is specified for the last parameter, the codes are interpreted as token codes; otherwise they are interpreted as type codes. See table A.39 for possible display strategies. See table A.40 for possible deletion strategies, and see table A.41 for possible change strategies.
A Search Criterion (Y,N)	Y - A search criterion is to be specified by the following three prompts.
search criterion ↓ Total frequency range (I5, 1X, I5)	A range of total frequencies.
search criterion ↓ Document frequency range (I5, 1X, I5)	A range of document frequencies
search criterion ↓ {Printing / Deletion} by alpha-numeric keys (Y,N)	The response to this prompt completes the specification of a search criterion. See table A.42 and table A.43 for possible display and deletion strategies respectively, in accordance with any possible search criterion.

Table A.38 -CONTINUED-

PROMPT	MEANING
Specify the output device (T,P)	Device for display of information
Do you want the records of codes printed (Y,N,F)	F - The first of the records associated with each type code (key type) on file 12 or 13 will be displayed.
Specify the sentence numbers in ascending order (I4)	The identification numbers of the sentences in the document to which the key is to be assigned. Each number should be specified in a separate line. Specification of zero signals the end of input in response to the prompt. If zero is specified at first, the key is assigned to the first sentence of the document.
Specify the maximum type code on file 13 (I6)	The maximum type code number on file 13. That is, the total number of phrase types.
Specify the maximum token code on file 13 (I6)	The maximum token code number on file. That is, the number of records in file.
Specify the key for replacement	A key should be specified for CHANGE command.
Specify the key to be added.	A key should be specified for ADD command.
Specify the command name (A,E,C)	A - Add command E - Erase (delete) command C - Change command

Table A.38

-CONTINUED-

PROMPT	MEANING
Specify the document numbers (I6)*	The identification numbers of these documents in which the respective in-document key types are to be changed or deleted. This prompt is issued, if only one type code is specified in response to the previous prompt. Each number should be specified in a separate line, and zero terminates the input. If zero is specified at first, all respective in-document key types are changed or deleted.
Any exceptions (I6)*	This prompt is issued when performing deletion by a search criterion, or performing automatic decomposition. The type codes of key types to be excluded from the deletion or decomposition process should be specified in successive lines. Zero can terminate the input. If zero is specified at first, no exception is made.
Specific codes (I6)*	This prompt is issued when performing deletion without a search criterion. Specific codes (in the range) can further be specified in successive lines for deletion. Zero can terminate the input. If zero is specified at first, all codes in the range are deleted.
Automatic Decomposition (Y,N)	This prompt is issued when performing deletion by a search criterion in which the upper bound for total frequency range is 1. See section 3.2.4 for the description of automatic decomposition process.
Range of codes to be decomposed (I6,1X,I6,1X,A1)	A range of codes. If a non-blank character is specified for the last parameter, the codes are interpreted as token codes; otherwise they are interpreted as type codes. The following prompt asks for the sub-phrases in the phrase.

Table A.38 -CONTINUED-

* The user can change the j^{th} code typed (from the end) by specifying '-j' and then the code for substitution. He can then proceed with specification of further codes.

PROMPT	MEANING
Beginning and Ending positions of sub-phrases (I2,1X,I2)	The beginning position specifies the beginning of sub-phrase relative to the beginning of the phrase. The ending position specifies the ending of sub-phrase relative to the beginning of the phrase. The user may specify as many position pairs as required. Zero can terminate the input.
Range of codes to be erased (I6, 1X, I6)	The code of the word (to be changed) in stop list should be specified.
Specify the word for replacement	A stop list word should be specified for change command.
Range of codes to be displayed (I6, 1X, I6)	This prompt is issued when performing display of any file except files 4,11,12, and 13. In file I, where I ≠ 4,11,12,13, the Nth record is assigned code N. See table A.44 for display strategies.

Note: Blank or zero (unless otherwise stated), or a negative number, in response to a prompt causes the system to reissue a previous prompt. Specification of EOF (End of File mark), at any point, terminates the operation of program.

Table A.38 -CONTINUED-

	CODE RANGE	MEANING
token codes	code-1 > 0 code-2 = 0	The in-document term type defined by code-1 is changed.
	0 < code-1 ≤ code-2	A set of in-document term types defined by code range are changed.
	otherwise	ERROR
type codes	code-1 > 0 code-2 = 0	In-document term types, which will be further specified, are changed.
	0 < code-1 ≤ code-2	A set of in-document term types defined by code range are changed.
	otherwise	ERROR

Note: (code-1, code-2) defines a code range.

Table A.41

Changes Strategies

FREQUENCY RANGE	DISPLAY STRATEGY	
	Display	Display by alpha-numeric keys,
a. 0≤KLFRQ≤KHFRQ b. 0≤DLFRQ≤DHFRQ	1 Among the terms which are candidates for display, those whose total and document frequencies fall in the ranges defined by (a) and (b) respectively, are displayed.	The same as for (1) except that only alpha-numeric terms are considered for display.
a. 0≤KLFRQ≤KHFRQ b. 0=DLFRQ=DHFRQ	2 Among the terms which are candidates for display, those whose total frequencies fall in the range defined by (a), are displayed.	The same as for (2) except that only alpha-numeric terms are considered for display.
a. 0=KLFRQ=KHFRQ b. 0≤DLFRQ≤DHFRQ	3 Among the terms which are candidates for display, those whose document frequencies fall in the range defined by (b), are displayed.	The same as for (3) except that only alpha-numbric terms are considered for display.
a. 0=KLFRQ=KHFRQ b. 0=DLFRQ=DHFRQ	All the terms that are candidates for display, are displayed.	All alpha-numeric terms among those which are candidates for printing, a are displayed.
Otherwise	ERROR	ERROR

Note: (KLFRQ,KHFRQ) and (DLFRQ, DHFRQ) define frequency ranges.

Table A.42

Display by a Search Criterion.

FREQUENCY RANGE	DELETION STRATEGY	
	Deletion	Deletion by alpha-numeric keys.
a. 0≤KLFRQ≤KHFRQ b. 0≤DLFRQ≤DHFRQ	1 Among the terms which are candidates for deletion, those whose total and document frequencies fall in the ranges defined by (a) and (b) respectively, and not excluded specifically, are deleted from the file.	The same as for (1) except that only alpha-numeric terms are considered for deletion.
a. 0≤KLFRQ≤KHFRQ b. 0=DLFRQ=DHFRQ	2 Among the terms which are candidates for deletion, those whose total frequencies fall in the range defined by (a), and not excluded specifically, are deleted from the file.	The same as for (2) except that only alpha-numeric terms are considered for deletion.
a. 0=KLFRQ=KHFRQ b. 0≤DLFRQ≤DHFRQ	3 Among the terms which are candidates for deletion, those whose document frequencies fall in the range defined by (b), and not excluded specifically, are deleted from the file.	The same as for (3) except that only alpha-numeric terms are considered for deletion.
a. 0=KLFRQ=KHFRQ b. 0=DLFRQ=DHFRQ	All the terms that are candidates for deletion, are deleted from the file.	All alpha-numeric terms among those which are candidates for deletion, and not excluded specifically, are deleted from the file.
Otherwise	ERROR	ERROR

Note: (KLFRQ, KHFRQ) and (DLFRQ, DHFRQ) define frequency ranges.

Table A.43 Deletion of Terms by a Search Criterion.

	CODE RANGE	MEANING
file 12, or 13	code-1 $\succ$ 0 code-2= 0	Records defined by code-1 are candidates for display.
	0 $\prec$ code-1 $\preceq$ code-2	A range of records defined by the code range are candidates for display.
	code-1 = 0 code-2 $\succ$ 0	Records associated with the first 'code-2' type codes in file are candidates for display.
	code-1 = code-2 = 0	All records in file are candidates for display.
	otherwise	ERROR
file I (I=4,11,12,13)	code-1 $\succ$ 0 code-2 = 0	Record defined by code-1 is displayed.
	0 $\prec$ code-1 $\preceq$ code-2	A range of records defined by the code range are displayed.
	code-1 = 0 code-2 $\succ$ 0	The first 'code-2' records of file are displayed.
	code-1 = code-2 = 0	All records in file are displayed.
	otherwise	ERROR

Table A.44

Display Strategies

ERROR MESSAGE	REMARK
Doc #(s) equal to (and greater than) N does (do) not belong to code = M	The term defined by code M does not appear in the indicated documents.
Doc # = N does not belong to code = M	The term defined by code M does not appear in the indicated document.
Invalid sub-phrase specification	Sub-phrase specification for manual decomposition is invalid.
Invalid "range of codes"	See table A.39, A.40, A.41, or A.44.
Invalid "frequency range"	See table A.42, or A.43.
Invalid file specification	An invalid file number. See table A.6.
Code(s) equal to (and greater than) M does (do) not exist in file.	The indicated code(s) does (do) not exist in file.
The code = M does not exist in file	The indicated code does not exist in file.
Too many codes requested	The user asked for display of non-existent terms.
Command name X is invalid	An invalid command name.
ADD command is not defined on file 4.	ADD command is not defined on random access phrase directory.
File K cannot be updated	The indicated file cannot be updated.

Note: N,M,K are integers. X is an alphabetic character.

Table A.45

List of Error Messages

A.2 Indexing Programs

A.2.1 Input Text

The original collection of text items must be placed on a computer-readable media in a format acceptable to the system. This format is referred to as Standard Format and the storage media of text items is referred to as Standard Formatted Text File.

Since all collections of text items are somewhat unique, it is the user's or the programmer's responsibility to write a computer program required to place his collection of text items into the Standard Formatted Text File (see appendix B).

The text items in the user's original collection may consist of titles, abstracts, full texts, index terms, or any combination of these. If the collection of text items has already been indexed, then the user needs only to place the index forms on the Standard Formatted Text File; otherwise he may place the full texts, abstracts, titles or any combination of these on the Standard Formatted Text File.

A.2.2 Extraction of Words and Phrases

Program: INDEX

Issue file commands for input and output files.

```
/FILE file-0,LINK = DSET 21, FCBTYPE = BTAM, RECFORM = V,
/ BLKSIZE = 4096, OPEN = INPUT
/FILE file-22, LINK = DSET 22, FCBTYPE = ISAM, RECFORM = V,
/OPEN = INPUT
/FILE file-1,LINK = DSET 23, FCBTYPE = BTAM, RECFRM = V,
/RECSIZE = 130, BLKSIZE = 4094, OPEN = INOUT
```

Invoke the load module of program.

```
/EXEC  LMINDEX
```

Enter input parameters.

<no of doc processed>Δ<no of doc to be processed>
<max term len>Δ<no of rec on out file>Δ<no of words found in stop list>Δ<restart>
<sen del>Δ<text letter case>Δ<no of sent>
<max word len>Δ<stop list>Δ<phr dic>
<no of word del>
<word dels>
Phrase Diction-ary: [<no of entry><len of entry><st and phr letter case>]
[<first field><following fields><cont field>]
. . . . . .
. . . . . .
<st and phr letter case>

Parameter Interpretation Table, INDEX

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<no of doc processed>	Number of documents processed so far. This is one of the statistics produced. '0' should be specified at the first run.	6-digit integer
<no of doc to be processed>	Number of documents to be processed. This field is ignored, if '0' is specified for field restart .	6-digit integer
<max term len>	Maximum length (in characters) of terms extracted so far. This is one of the statistics produced. '0' should be specified at the first run.	2-digit integer
<no of rec on out file>	Number of records written to the output file so far. This is one of the statistics produced. '0' should be specified at the first run.	10-digit integer
<no of words found in stop list>	Number of word tokens found to be in stop list so far. This is one of the statistics produced. '0' should be specified at the first run.	

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<restart>	0-All the remaining (unprocessed) documents are to be processed at the current run. 1-Only a number of documents indicated by the field: no of doc to be processed are to be processed.	1-digit integer
<sen del>	Sentence delimiter	1 character
<text letter case>	1-The text consists of both lower and upper case letters. 2-The text consists of only upper case letters.	2-digit integer
<no of sent>	Number of sentences not to be processed from the beginning part of each document. For instance, '0' means that all sentences of each document will be processed, and '1' means that all sentences except the first one will be processed.	2-digit integer
<max word len>	Upper bound for the length of words to be extracted from the text. This field defines also the fixed length of words in stop list.	5-digit integer
<stop list>	0-A stop list is used. 1-No stop list is used.	1-digit integer
<phr dic>	0-A phrase dictionary is used. 1-No phrase dictionary is used.	1-digit integer
<no of word del>	Number of word delimiters to be specified	5-digit integer
<word dels>	Word delimiters	1 to 30 characters (left justified)
<no of entry>	Number of entries in the phrase dictionary	5-digit integer

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<len of entry>	The (fixed) length of the first field of each dictionary entry (=20).	2-digit integer
<st and phr letter case>	0-Stop list words and user specified phrases consist of both lower and upper case letters. 1-They consist of only upper case letters.	1-digit integer
<first field>	The first field of a dictionary entry. The word in this field must be left-justified.	20 characters
<following fields>	The second and following fields of the dictionary entry. Information in these fields may be typed in free-format.	1-59 characters
<cont field>	Entry continuation field: Blank- The next record will commence a new entry. Nonblank- The next record will be used for the specification of the same entry.	1 character at 80th column

Output Statistics, INDEX

(1) Number of documents processed up to now.

(2) Maximum length of terms (in characters) extracted up to now.

(3) Number of records (term tokens) written to output file.

(4) Number of word tokens found to be in stop list up to now.

EXAMPLE 1.

Suppose that a user wants to process the first 100 documents on the Standard Formatted Text File, and wants also to define a phrase

dictionary consisting of the phrases: "computer index", "manual index', and "north atlantic treaty". Then, the input parameters would be specified as follows:

```
000000Δ000100
00Δ0000000000Δ0000000000Δ1
.Δ02Δ00
00020Δ0Δ0
00009
<>( ) % @ ' " Δ
000022001
INDEXΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ = 2 COMPUTER + & 1  MANUAL +#
TREATYΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ = 3 NORTH + ATLANTIC + #
```

EXAMPLE 2.

Suppose that the statistics produced by example 1 would be:

```
max term len  = 23
no of rec on out file  = 50000

no of words found in stop list  = 24000,
```

and the user wants to process the remaining (unprocessed) documents on the Standard Formatted Text File, without using a phrase dictionary. Then, the input parameters would be specified as follows:

```
000100
23Δ0000050000Δ00000024000Δ0
.Δ02Δ00
00020Δ0Δ1
00009
< > ( ) % @ ' " Δ
1
```

A.2.3 Automatic Phrase Generation

Program: PHRGEN

Issue file commands for input and output files.

```
/FILE file-1, LINK = DSET 21, FCBTYPE = BTAM, RECFORM = V,
/RECSIZE = 130, BLKSIZE = 4024, OPEN = INPUT
/FILE file-3, LINK = DSET 22, FCBTYPE = BTAM, RECFORM = V,
/RECSIZE = 132, BLKSIZE = 4084, OPEN = OUTPUT
```

Invoke the load module of program.

/EXEC LMPHRGEN

Enter input parameters.

<upper bound>Δ<device>

Parameter Interpretation Table, PHRGEN

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<upper bound>	Upper bound for the length of any phrase to be generated. This should be less than or equal to 60 characters.	2-digit integer
<device>	Device specification for display of statistics produced. T - terminal P - line printer	1 character

Output Statistics, PHRGEN

(1) Maximum length (in characters) of phrases generated.
(2) Maximum number of components in any generated phrase.
(3) Number of records (phrase tokens) written to output file.

EXAMPLE

Suppose that the output statistics will be displayed on a terminal, and no phrase with length longer than 50 characters will be generated. Then, the input parameters should be specified as follows:

50ΔT

A.2.4 Refinement of Phrases

A.2.4.1 Transformation of Variable Length Phrases to Fixed Length Phrases

Program: VARFIX

Issue file commands for input and output files.

```
/FILE file-3, LINK=DSET21, FCBTYPE=BTAM, RECFORM=V,
/RECSIZE=132, BLKSIZE=4085, OPEN=INPUT
/FILE file-4, LINK=DSET22,FCBTYPE=ISAM, RECFORM=V,
/RECSIZE=44, BLKSIZE=2048, OPEN=OUTPUT
```

Invoke the load module of program.

```
/EXEC LMVARFIX
```

Enter input parameters.

<max phrase len>Δ<option>

Parameter Interpretation Table, VARFIX

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<max phrase len>	Maximum length of phrases generated.	2-digit integer
<option>	Zero should be specified.	1-digit integer

EXAMPLE

Assume that the maximum length of phrases generated by program PHRGEN is 48. Then, the input parameters would be specified as follows:

48Δ0

A.2.4.2 Alphabetical Sorting

File 4 should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file 19 are given below.

Sort of file 4: (A,DN,SN)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	CH,A	2	first character of term
2	CH,A	2	second character of term
.	.	.	.

Sort of file 4: (A,DN,SN) -CONTINUED-

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
.	.	.	.
.	.	.	.
MAXPL	CH,A	2	last character of term
MAXPL+1	FI,A	4	document identification number
MAXPL+2	FI,A	2	sentence identification number

Note: MAXPL denotes the maximum number of characters per phrase.

A.2.4.3 Production of Phrase Types

Program: UNWRDS

Issue file commands for input and output files.

```
/FILE file-19, LINK=DSET21,FCBTYPE=SAM,RECFORM=V,
/RECSIZE=144,BLKSIZE=2048,OPEN=INPUT
/FILE file-13,LINK=DSET22,FCBTYPE=ISAM,RECFORM=V,
/OPEN=OUTPUT
/FILE file-16,LINK=DSET25,FCBTYPE=SAM,RECFORM=V,
/RECSIZE=140,BLKSIZE=2048,OPEN=OUTPUT
/FILE file-23,LINK=DSET23,FCBTYPE=ISAM,RECFORM=F,
/RECSIZE=16,BLKSIZE=2048,OPEN=OUTPUT
```

Invoke the load module of program

```
/EXEC LMUNWRDS
```

Enter input parameters

```
<option>Δ<max phrase len>
<sub-option>
```

Parameter Interpretation Table, UNWRDS

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<option>	1 should be specified	1-digit integer
<max phrase len >	Maximum length of phrases generated	2-digit integer
<sub-option>	2 should be specified	1-digit integer

Output Statistics, UNWRDS

(1) Number of phrase types produced.

EXAMPLE

Assume that the maximum length of phrases is 48. Then, the input parameters would be specified as follows:

1Δ48
2

A.2.4.4 Sorting by Total Frequency

File 16 should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file 15 are given below.

Sort of file 16: (TF,DF,A)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	FI,A	4	Total frequency
2	FI,A	4	Document frequency
3	FI,A	4	Type code

Note: Ordering of data by type codes is identical to alphabetical ordering of data.

A.2.4.5 Generating A Summary Report

Program: FRQWCN

Issue file command for input file.

/FILE {file-14 / file-15},LINK=DSET21,FCBTYPE=SAM,RECFORM=V,RECSIZE=140,OPEN=INPUT

Issue file command for a temporary file to be used by program.

/FILE TEMP,LINK=DSET22,FCBTYPE=ISAM,RECFORM=V,OPEN=OUTIN

Invoke the load module of program.

/EXEC LMFRQWCN

Enter input parameters.

<max term len>Δ<option>Δ<order>Δ<cut-1>Δ<cut-2>Δ<no of docs>

Parameter Interpretation Table, FRQWCN

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<max term len>	Maximum number of characters per term (word or phrase).	2-digit integer
<option>	1 should be specified	1-digit integer
<order>	0-Order of data: (TF,DF,A) 1-Order of data: (DF,TF,A)	1-digit integer
<cut-1> = a	Specifies a frequency pair (TF,DF). In order to delete a % of term types from the beginning part of file 15 (or 14) the user must perform deletion by a search criterion in which upper bounds for total and document frequency ranges are TF, and DF, respectively. A full summary report is produced up to the point corresponding to (TF,DF). If '0' is specified, only frequency distribution of terms is generated up to the corresponding point.	

Parameter Interpretation Table, FRQNCN -CONTINUED-

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<cut-2> = b	Specifies a frequency pair (TF_2,DF_2). In order to delete b % of term types from the ending part of file 15 (or 14), the user must perform deletion by a search criterion in which lower bounds for total and document frequency ranges are TF_2 and DF_2 respectively. A full summary report is generated after the point corresponding to (TF_2,DF_2). If'0' is specified, only frequency distribution of terms is generated after the corresponding point.	2-digit integer
<no of docs>	Number of document representatives.	6-digit integer

Output Report and Statistics, FRQNCN

A Summary Report is generated. See section 3.2.7.3 for a full description of a Summary Report. This report contains many useful statistics.

EXAMPLE

Let us suppose the following:

(1) Number of document representatives=425. Note that if no document representative has been deleted or merged, then this total is equal to number of documents in the data-base.
(2) Input file is ordered as: (TF,DF,A).
(3) Maximum length of phrases is 48.

If we want to generate a full Summary Report, the input parameters would have to be specified as follows:

48Δ|Δ0Δ99Δ00Δ000425

For instance, table 3.16 shows the beginning portion of a Summary Report generated for the illustrative Time Data-base. However, for the illustrative data-base, the input file has been ordered as; (DF,TF,A).

A.2.4.6 Generating a System Directory to the Standard Formatted Text File

Program: STDFIL

Issue file commands for input and output files.

```
/FILE file-0,LINK=DSET21,FCBTYPE=BTAM,RECFORM=V,BLKSIZE=4096,OPEN=INPUT
/FILE RA-file-0,LINK=DSET22,FCBTYPE=ISAM,RECFORM=V,OPEN=OUTPUT
/FILE file-24,LINK=DSET23,FCBTYPE=ISAM,RECFORM=F,RECSIZE=20,OPEN OUTPUT
```

Invoke the load module of program.

```
/EXEC  LMSTDFIL
```

Enter input parameters.

<block size>

Parameter Interpretation Table

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<block size>	Maximum allowable physical block size (in bytes).	5-digit integer

A.2.4.7 Display of Information

A.2.4.7.1 On-line Display

Program: PRTON

Issue file commands for input files.

```
/FILE RA-file-0, LINK=DSET30,FCBTYPE=ISAM,RECFORM=V,OPEN=INPUT
/FILE file-24,LINK=DSET21,FCBTYPE=ISAM,RECFORM=F,RECSIZE=20,OPEN=INPUT

/FILE file-4,LINK=DSET34,FCBTYPE=ISAM,RECFORM=V, RECSIZE=144,OPEN=INPUT
/FILE file-13,LINK=DSET43,FCBTYPE=ISAM, RECFORM=V,OPEN=INPUT
/FILE file-23,LINK=DSET23,FCBTYPE=ISAM, RECFORM=F,RECSIZE=16,OPEN=INPUT

/FILE {file-15},LINK=DSET46,FCBTYPE=ISAM, RECFORM=V,RECSIZE=140,
      {file-16}
/OPEN=INPUT

/FILE file-1,LINK=DSET31, FCBTYPE=ISAM,RECFORM=V,OPEN=INPUT

/FILE file-3,LINK=DSET33,FCBTYPE=ISAM,RECFORM=V,OPEN=INPUT

/FILE file-22,LINK=DSET44,FCBTYPE=ISAM,RECFORM=V,OPEN=INPUT
```

Invoke the load module of program.

```
/EXEC LMPRTON
```

Input parameters should be entered in response to prompts issued by the system. See table A.38 for the description of all possible prompts. See tables 3.18 and 3.19 for examples of display of information.

A.2.4.7.2 Off-line Display

Program: PRTOFF

Issue file command for input file.

```
/FILE <name>,LINK=DSET21,OPEN=INPUT
```

where <name> denotes the name under which the input file is cataloged.

Enter the load module of program.

```
/EXEC LMPRTOFF
```

Enter input parameters, in accordance with the input file.

(1) Input file = file-0, or RA-file-0

<option>[Δ<no of docs>]
<line size>
[<doc no>]
.
.
.
[<doc no>]

(2) Input file = file-13

<option>Δ<no of phrase types>Δ<max phrase len>
<sen option>
<rec option>

(3) Input file = file-I, where I=4,15,16,19

<option>Δ<no of recs>Δ<max phrase len>

(4) Input file = file-I (I=1,3)

<option>Δ<no of recs>

(5) Input file = file-22

<option>Δ<no of recs>Δ<max word len>

Parameter Interpretation Table, PRTOFF

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<option>	Denotes the number of the file to be displayed.	2-digit integer
<no of docs>	Number of text items to be printed from the beginning part of the file. If zero is specified, all text items are printed.	6-digit integer
<line size>	The length of a printed line in characters (≤125).	3-digit integer

Parameter Interpretation Table, PRTOFF

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<doc no>	Specific text items can be printed by specifying respective document identification numbers. In this case, the parameter <no of docs> is not needed.	6-digit integer
<no of phrase types>	Number of phrase types [along with associated information] to be printed from the beginning of the file. If zero is specified, all phrase types are printed.	6-digit integer
<max phrase len>	Maximum length of phrases generated	2-digit integer
<sen option>	Y- Sentence identification numbers within each record associated with phrase (term) type will be printed. N- They will not be printed	1 character
<rec option>	F- The first of the records associated with each phrase (term) type will be printed Y- All associated records will be printed. N- No associated record will be printed.	1 character
<no of recs>	Number of records to be printed from the beginning part of file. If zero is specified, all records are printed.	6-digit integer
<max word len>	see page	2-digit integer

A.2.4.8 Processes To Delete and Decompose Phrases

Program: UPDATE

Issue file commands for input files.

```
/FILE file-4,LINK=DSET34,FCBTYPE=ISAM,RECFORM=V,RECSIZE=144,OPEN=INOUT
/FILE file-13,LINK=DSET43,FCBTYPE=ISAM,RECFORM=V,OPEN=INOUT
/FILE file-23,LINK=DSET23,FCBTYPE=ISAM,RECFORM=F,RECSIZE=16,OPEN=INPUT
```

Invoke the load module of program.

/EXEC LMUPDATE

Input parameters should be entered in response to prompts issued by the system (see table A.38). Table 3.20 shows several examples illustrating the use of updating functions.

A.2.5 Merging of Word and Phrase Tokens

A.2.5.1 Sorting By Document Identification Numbers

The file containing refined phrase tokens should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file 4 are given below.

Sort of file 4 : (DN,SN,P-A)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	FI,A	4	Document identification number
2	FI,A	2	Sentence identification number
3	FI,A	2	Phrase-address

A.2.5.2 Merging of Word and Phrase Tokens

Program:MRWRPH

Issue file commands for input and output files.

/FILE {file-3 / file-4} ,LINK=DSET22,FCBTYPE=BTAM,RECFORM=V,OPEN=INPUT
/FILE file-1,LINK=DSET21,FCBTYPE=BTAM,RECFORM=V,RECSIZE=130,
/BLKSIZE=4024,OPEN=INPUT
/FILE file-5,LINK=DSET23,FCBTYPE=BTAM,RECFORM=V,RECSIZE=134,
/BLKSIZE=4002,OPEN=OUTPUT

Invoke the load module of program.

/EXEC LMMRWRPH

Enter input parameters

<max phrase len>Δ<max term len>Δ<sub-option>Δ<device>

Parameter Interpretation Table, MRWRPH

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<max phrase len =MXP>	Maximum length of phrases generated.	2-digit integers
<max term len =MXP>	See page . Max (MXP,MXT) defines the maximum number of characters per term on output file.	2-digit integer
<sub-option>	0- Specifies file-4 as one of the input files. 1- Specifies file-3 as one of the input files.	1-digit integer
<device>	Device specification for display of statistics produced. T- terminal P- line printer	1 character

Output Statistics, MRWRPH

(1) Number of term tokens written to output file.
(2) Max (MXP,MXT) is printed out for information of the user.

EXAMPLE

Suppose that file-4 is one of the input files, and maximum length of phrases is 48, and statistics will be displayed on a terminal. Then, the input parameters would be as: 48Δ00Δ0ΔT

A.2.6 Consolidation of Multiple Occurrences of Terms Within Each Document

A.2.6.1 Transformation of Variable Length Terms to Fixed Length Terms

This step should be performed, if no phrase generation has been performed.

Program: VARFIX

Issue file commands for input and output files.

```
/FILE file-1,LINK=DSET21,FCBTYPE=BTAM,RECFORM=V,RECSIZE=130,
/BLKSIZE=4024,OPEN=INPUT
/FILE file-2,LINK=DSET22,FCBTYPE=BTAM,RECFORM=V,RECSIZE=130,
/BLKSIZE=4024,OPEN=OUTPUT
```

Invoke the load module of program.

```
/EXEC LMVARFIX
```

Enter input parameters.

<max term len>Δ<option>

Parameter Interpretation Table, VARFIX

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<max term len>	See page 292. This parameter defines the maximum number of characters per term on output file.	2-digit integer.
<option>	2 should be specified.	1-digit integer

EXAMPLE

Suppose that maximum length of phrases extracted (max term len) is 23. Then, the input parameters would be specified as follows: 23Δ2

A.2.6.2 Alphabetical Sorting Within Each Document Representative

File 2,5or11 should be ordered using SORT package. Control fields specifying the ordering of data in the sorted file 18, 17, or 21 respectively are given below.

Sort of file 2,5, or 11: (DN,A)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	FI,A	4	Document identification number.
2	CH,A	2	First character of term.
3	CH,A	2	Second character of term.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
MAXTL+1	CH,A	2	Last character of term.
[MAXTL+2]	FI,A	2	Sentence identification number.

Note: MAXTL denotes the maximum number of characters per term.

A.2.6.3 Production of In-document Term Types

Program: ELDID

Issue file commands for input/output and temporary files.

```
/FILE {file-17} ,LINK=DSET21,FCBTYPE=BTAM,OPEN=INPUT
      {file-18}
      {file-21}
/FILE file-6,LINK=DSET22,FCBTYPE=SAM,RECFORM=V,OPEN=OUTPUT
/FILE TEMP,LINK=DSET23,FCBTYPE=ISAM,RECFORM=V,OPEN=OUTIN
```

Invoke the load module of program.

```
/EXEC L*ELDID
```

Enter input parameters

<overindexed><underindexed><max term len><option>

Parameter Interpretation Table, ELDID

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<overindexed>	Specifies a maximum for the number of in-document index term types assigned to any document. Any document containing more in-document index term types than this maximum is an overindexed document.	3-digit integer
<underindexed>	Similarly, this parameter specifies a minimum, and consequently defines the underindexed documents.	2-digit integer
<max term len>	Maximum number of characters per term.	2-digit integer
<option>	0- Specifies file 17 or 18 as the input file. 1- Specifies file 21 as the input file.	1-digit integer

Output Statistics, ELDID

(1) Number of document representatives processed.
(2) Number of records written to output file.
(3) List of over indexed documents.
(4) List of under indexed documents.

EXAMPLE

Suppose that the parameters 100 and 10 define the overindexed and underindexed documents respectively. Suppose also that maximum number of characters per term is 48 and file 17 is the input file. Then, the input parameters would be specified as follows:

10010480

A.2.7 Reducing the Key Vocabulary Size

A.2.7.1 Production of Term Types

A.2.7.1.1 Alphabetical Sorting

File 6 should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file 7 are given below.

Sort of file 6 : (A,DN)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	CH,A	2	First character of term.
2	CH,A	2	Second character of term.
.	.	.	.
.	.	.	.
.	.	.	.
MAXTL	CH,A	2	Last character of term.
MAXTL+1	FI,A	4	Document identification number.

A.2.7.1.2 Production of Term Types

Program: UNWRDS

Issue file commands for input and output files.

```
/FILE file-7,LINK=DSET21,FCBTYPE=SAM,RECFORM=V,OPEN=INPUT
/FILE file-11,LINK=DSET24,FCBTYPE=ISAM,RECFORM=V,OPEN=OUTPUT
/FILE file-12,LINK=DSET22,FCBTYPE=ISAM,RECFORM=V,OPEN=OUTPUT
/FILE file-16,LINK=DSET25,FCBTYPE=SAM,RECFORM=V,RECSIZE=140,OPEN=OUTPUT
/FILE file-23,LINK=DSET23,FCBTYPE=ISAM,RECFORM=F,RECSIZE=16,OPEN=OUTPUT
```

Invoke the load module of program.

```
/EXEC LMUNWRDS
```

Enter input parameters

```
<option>Δ<max term len>
<sub-option>
```

Parameter Interpretation Table, UNWRDS

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<option>	0- production of term types 2- production of key types	1-digit integer
<max term len>	Maximum number of characters per term.	2-digit term
<sub-option>	2 should be specified.	1-digit integer
<freq cut-off>	Frequency cut-off parameter described in section 3.2.8	2-digit integer

Output Statistics, UNWRDS

(1) Number of term types produced.

EXAMPLE

Assume that the maximum number of characters per term is 48.

Then, the input parameters would be specified as follows:

0Δ48
2

A.2.7.2 Similarly Spelled Terms

Program: ADJCMP

Issue file commands for input files.

/FILE (file-12),LINK=DSET22,FCBTYPE=ISAM,RECFORM=V,OPEN=INPUT
      (file-13)
/FILE file-23,LINK=DSET22,FCBTYPE=ISAM,RECFORM=F,RECSIZE=16,OPEN=INPUT

Invoke the load module of program.

/EXEC LMADJCMP

Enter input parameters
<dev>
<max term len>Δ<similarity>Δ<dissimilarity>Δ<option>

Parameter Interpretation Table, ADJCMP

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<max term len>	Maximum number of characters per term.	3-digit integer
<similarity>	Minimum number of characters that two similar terms must have in common on a left to right comparison.	3-digit integer
<dissimilarity>	Maximum number of characters between the first pair of corresponding characters that do not match and the end of the longer of the two similar terms being compared.	3-digit integer
<option>	2 should be specified	1-digit integer
<dev>	P - line-printer T - terminal	1 character

Output Report, ADJCMP

Lists of term types in which similarly spelled terms are brought together and arranged in groups. Each term type is associated with its respective type code, and document and total frequencies. Table 3.15 shows a portion of such a report for the illustrative data-base.

EXAMPLE

Below are example input parameters which have been used in processing the illustrative data-base (see table 3.15).

048Δ003Δ002Δ2

A.2.7.3 Summary Report

A.2.7.3.1 Sorting By Document Frequency

File 16 should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file 14 are given below.

Sort of file 16 : (DF,TF,A)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	FI,A	4	Document frequency
2	FI,A	4	Total frequency
3	FI,A	4	Type Code

A.2.7.3.2 Generating A Summary Report

See section A.2.4.5

A.2.7.4 Comparison of Old and New Vocabularies of Index Terms

Program: COMDIR

Issue file commands for input files.

```
/FILE old-file-12,LINK=DSET21,FCBTYPE=ISAM, RECFORM=V,OPEN=INPUT
/FILE new-file-12,LINK=DSET22,FCBTYPE=ISAM,RECFORM=V,OPEN=INPUT
```

Invoke the load module of program.

```
/EXEC LMCOMDIR
```

<old max term len>Δ<new max term len>Δ<block factor>

Parameter Interpretation Table, COMDIR

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<old max term len>	Maximum number of characters per term on the old vocabulary of index terms.	2-digit integer

Parameter Interpretation Table, COMDIR, (continued)

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<new max term len>	Maximum number of characters per term on the new vocabulary of index terms	2-digit integer
<block factor>=B	(B-4)/ 4 specifies the maximum block size in bytes	5-digit integer

Output Report, COMDIR

The program generates a report which contains the following information:

(1) Common index terms in the old and new vocabularies.
(2) Index terms, which exist in the old but not in the new vocabulary.
(3) Index terms, which are present in the new but not in the old vocabulary.

In addition, with each index term are associated the identification numbers of the documents to which it belongs. Table 3.17 shows a portion of such a report for the illustrative data-base.

EXAMPLE

Below are example input parameters which have been wed in processing the illustrative data-base.

48Δ48Δ00500

A.2.7.5 Display of Information

A.2.7.5.1 On-line Display

Program: PRTON

Issue file commands for input files.

```
/FILE RA-file-0,LINK=DSET30,FCBTYPE=ISAM,RECFORM=V,OPEN=INPUT
/FILE file-24,LINK=DSET21,FCBTYPE=ISAM,RECFORM=V,RECSIZE=20,OPEN=INPUT

/FILE file-11,LINK=DSET41,FCBTYPE=ISAM,RECFORM=V,OPEN=INPUT
/FILE file-12,LINK=DSET42,FCBTYPE=ISAM,RECFORM=V,OPEN=INPUT
/FILE file-23,LINK=DSET22,FCBTYPE=ISAM,RECFORM=F,RECSIZE=16,OPEN=INPUT

/FILE file-14,LINK=DSET46,FCBTYPE=ISAM,RECFORM=V,RECSIZE=140,OPEN=INPUT

/FILE {file-6}
      {file-7},LINK=DSET36,FCBTYPE=ISAM,RECFORM=V,OPEN=INPUT

/FILE {file-2 }
      {file-18},LINK=DSET32,FCBTYPE=ISAM,OPEN=INPUT

/FILE {file-5 }
      {file-17},LINK=DSET35,FCBTYPE=ISAM,OPEN=INPUT

/FILE file-8,LINK=DSET38,FCBTYPE=ISAM,OPEN=INPUT

/FILE file-9,LINK=DSET39,FCBTYPE=ISAM,OPEN=INPUT

/FILE {file-10}
      {file-20} ,LINK=DSET40,FCBTYPE=ISAM,OPEN=INPUT
```

Invoke the load module of program

```
/EXEC LMPRTON
```

Input parameters should be entered in response to prompts issued by the system. See table A.38 for the description of all possible prompts, and see tables 3.18 and 3.19 for examples of display of information.

A.2.7.5.2 Off-line Display

Program : PRTOFF

Issue file command for input file.

/FILE name ,LINK=DSET21,OPEN=INPUT

where <name> denotes the name under which the input file is cataloged.

Enter the load module of program.

/EXEC LMPRTOFF

Enter input parameters, in accordance with the input file.

(1) Input file = file-12

<option>Δ<no of term types>Δ<max term len>
<sen option>
<rec option>

(2) Input file = file-I, where I=2,5,8,9,14,17,18

<option>Δ<no of recs>Δ<max term len>

(3) Input file = file-11

<option>Δ<no of recs>Δ<max term len>
tok sen option

(4) Input file = file -I, where I=6,7,10,20

<option>Δ{<no of recs> / <no of doc repr>}Δ<max term len>
[<tok sen option>]*
[<doc repr option>]
[{<doc no> / <low bound>Δ<upper bound>}]
.
.
.

*This parameter should not be specified for file 10 or 20.

Parameter Interpretation Table, PRTOFF

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<option>	Denotes the number of the file to be displayed.	2-digit integer
<no of term types>	Number of term types [along with associated information] to be printed from the beginning part of the file. If zero is specified, all term types are printed.	6-digit integer
<max term len >	Maximum number of characters per term.	2-digit integer
<sen option>	See page 305	1 character
< rec option >	See page 305	1 character
< no of recs >	See page 305	6-digit integer
<tok sen option>	Y- Sentence identification numbers associated with each in-document term type will be printed. N- They will not be printed.	1 character
<no of doc repr>	Number of document representatives to be printed from the beginning of the file. If zero is specified, all document representatives are printed.	1-digit integer
<doc repr option>	If this parameter is not entered, then the first '<no of recs>' records are printed. If the parameter is entered, then the value interpretation is as follows: 1-The first '<no of doc repr>' document representatives are printed 2-Document representatives specified by parameter doc no are printed 3-Document representatives specified by parameters <low bound> and <upper bound> are printed.	1-digit integer

Parameter Interpretation Table, PRTOFT

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<doc no>	Identification number of a document whose representative will be printed.	6-digit integer
<low bound>=a <upper bound> =b	These two parameters define a range (a,b). Any document representative containing N in-document term types, where a≤N≤b, will be printed.	2-digit integers

A.2.7.6 Updating Functions

A.2.7.6.1 Updating of Index Terms

These processes enable the user to delete, consolidate, change, and add index terms.

Program : UPDATE

Issue file commands for input files.

```
/FILE file-11,LINK=DSET41,FCBTYPE=ISAM,RECFORM=V,OPEN=INOUT
/FILE file-12,LINK=DSET42,FCBTYPE=ISAM,RECFORM=V,OPEN=INOUT
/FILE file-23,LINK=DSET22,FCBTYPE=ISAM,RECFORM=F,RECSIZE=16,OPEN=INPUT
```

Invoke the load module of program.

```
/EXEC LMUPDATE
```

Input parameters should be entered in response to prompts issued by the system (see table A.38). Table 3.20 contains several examples illustrating the use of updating functions for index terms.

A.2.7.6.2 Updating of Document Representatives

These processes enable the user to delete or merge document representatives.

Program : DELMER

Issue file commands for input and output files.

```
/FILE file-6,LINK=DSET21,RECFORM=V,OPEN=INPUT
/FILE M.file-6,LINK=DSET22,RECFORM=V,OPEN=OUTPUT
```

where M.file-6 is the name under which the output file is cataloged. The output file is identical in structure to the input file, and can therefore be renamed as file-6.

Invoke the load module of program.

```
/EXEC LMDELMER
```

Enter input parameters

```
<max term len>
<command name>
<doc no>
   .
   .
   .
```

Parameter Interpretation Table, DELMER

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<max term len>	Maximum number of characters per term	2-digit integer
<command name>	D - Delete M - Merge	1 character
<doc no>	Identification number of some document to be deleted or merged.	6-digit integer

A.2.7.6.3 Updating of Stop List Words

These processes enable the user to delete, change, and add stop list words.

Program: UPDATE

Issue file command for input file.

/FILE file-22,LINK=DSET44,FCBTYPE=ISAM,RECFORM=V,OPEN=INOUT

Invoke the load module of program.

/EXEC LMUPDATE

Input parameters should be entered in response to prompts issued by the system (see table A.38).

A.2.8 Document and Key Surrogates

To create document surrogates, the processes described in sections A.2.6.2 and A.2.6.3 should first be performed for (updated) file-11. Then proceed in accordance with instructions provided below.

A.2.8.1 Alphabetical Sorting

File 6 (resulted from file-21) should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file 7 are given in page 311.

A.2.8.2 Production of Key Types

Program: UNWRDS

Issue file command for input and output files.

```
/FILE file-7,LINK=DSET21,FCBTYPE-SAM,RECFORM=V,OPEN=INPUT
/FILE (file-8),LINK=DSET22,FCBTYPE=ISAM,RECFORM=V,OPEN=OUTPUT
      (file-9)
/FILE file-10,LINK=DSET23,FCBTYPE=SAM,RECFORM=V,OPEN=OUTPUT
[/FILE file-10-Without-low-frequency-keys,LINK=DSET23,FCBTYPE=SAM,
/RECFORM=V,OPEN=OUTPUT]
```

Invoke the load module of program.

/EXEC LMUNWRDS

<option>Δ<max term len>Δ<freq cut-off>

See page312 for the interpretation of parameters.

A.2.8.3 Alphabetical Sorting Within Each Document Representative

File 10 [without low-frequency keys] should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file 20 are given in page 309

A.2.8.4 Generating Document or Key Surrogates

Program: DOCSUR

Issue file commands for input and output files.

/FILE {file-20 / file-V},LINK=DSET21,FCBTYPE=SAM,OPEN=INPUT
/FILE {file-I / file-XVII},LINK=DSET22,FCBTYPE=ISAM,RECFORM=V,RECSIZE=528,
/OPEN=OUTPUT

Invoke the load module of program.

/EXEC LMDOCSUR

Enter input parameters

<no of doc repr>Δ<option>Δ<max no keys>Δ<max term len>

Parameter Interpretation Table, DOCSUR

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<no of doc repr>	Number of document representatives, or key types.	6-digit integer
<max no keys>	250 should be specified.	3-digit integer
<max term len>	Maximum number of characters per term. This parameter is not needed if file-V is input.	2-digit integer
<option>	0-file V as the input file. 1-file 20 as the input file.	1-digit integer

A.3 Classification

A.3.1 Document Classification

A.3.1.1 Classification Tree

Program : CLASFY

Issue file commands for input/output and temporary files.

```
/FILE file-I,LINK=DSET21,FCBTYPE=ISAM,RECFORM=V,RECSIZE=528,
/OPEN=INOUT
/FILE file-II,LINK=DSET20,FCBTYPE=ISAM,RECFORM=V,RECSIZE-24,
/OPEN=OUTPUT
/FILE temp-1,LINK=DSET22,FCBTYPE=ISAM,RECFORM=V,OPEN=OUTIN
/FILE temp-2,LINK=DSET23,FCBTYPE=ISAM,RECFORM=V,OPEN=OUTIN
/FILE temp-(I+2),LINK=DSET(23+I),FCBTYPE=ISAM,RECFORM=V,OPEN=OUTIN
```

where I = 1,2N, and N denotes the stratification number for the classification tree to be generated.

Invoke the load module of program.

```
/EXEC LMCLASFY
```

Enter input parameters, for the first run of program.

```
<no of doc to be processed>
<strat>Δ<sens> <group size>Δ<block fac>Δ<block fac 1>Δ<date>
```

Enter input parameters, for the restart of program.

```
<no of doc to be processed>
<sens>Δ<group size>Δ<date>
```

Parameter Interpretation Table, CLASFY

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<no of doc to be processed>	Specifies the number of document surrogates to be processed at the current run.	10-digit integer
<strat>	Specifies the stratification number for the classification tree to be generated.	2-digit integer

Parameter Interpretation Table, CLASFY

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<sens =a>	Specifies the sensitivity factor that prevents the number of keys in any one group from differing from that in any other group by more than a. See section 3.3.1.2	4-digit integer
<group size>	Specifies a maximum group size which determines a terminal node or cell. See section 3.3.1.1.	5-digit integer
<block fac>=b	2xb specifies the maximum block size (in bytes) on the computer system.	5-digit integer
<block fac1> = b1	b1 = b-10	5-digit integer
<date>	Specifies the date at which the program is run.	6-digit integer

Output Report, CLASFY

For each partition in the classification process, a summary report is printed. An example of such a report for the illustrative Time data-base is given in table A.46. The circled numbers indicate each of the elements in the report, which are further described below.

(1) Date of the run.
(2) Total number of document surrogates in the current node being partitioned.
(3) Integer node number assigned to the node being partitioned
(4) Stratification number for the tree.
(5) Sensitivity factor.
(6) Maximum group size.
(7) This occurs as a result of the sensitivity factor test. Namely, the number of times the test: $n_i + a_i = n_j + a_j + E$ (see section 3.3.1.2), is performed.
(8) The number of document surrogates that are not assigned to a group in PASS 2, but are instead written onto a temporary file for input to PASS 3.
(9) Identifies the group in the particular partition.
(10) Number of keys in the group.
(11) Number of document surrogates in the group.

(12) Integer node number assigned to this particular group.
(13) If the group forms a terminal node or cell, then this factor is indicated underneath the group number.
(14) If a node is artificially partitioned (see section 3.3.1.2), it is also indicated in the report. However, this did not occur in the case of illustrative data-base.

Output Statistics, CLASFY

(1) Number of levels in the classification tree.
(2) Number of nodes in the classification tree.
(3) Number of records written to file II or XVIII.

A.3.1.2 Classification Algorithm

A.3.1.3 Generating Complete Set of Keys Within A Cell

A.3.1.3.1 Sorting By Document Identification Numbers

File II should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file III are given below.

Sort of file II : (DN)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	FI,A	4	Document identification number

A.3.1.3.2 Assignment of In-document Key Types To Respective Cells

Program : GNTRCD

Issue file commands for input and output files.

```
/FILE file-III,LINK=DSET21,FCBTYPE=ISAM,RECFORM=V,RECSIZE=24,OPEN=INOUT
/FILE file-20,LINK=DSET22,FCBTYPE=SAM,RECFORM=V,OPEN=OUTPUT
/FILE file-IV,LINK=DSET23,FCBTYPE=SAM,RECFORM=V,RECSIZE=12,OPEN=OUTPUT
```

Invoke the load module of program.

```
/EXEC LMGNTRCD
```

Enter input parameters

<max term len>Δ<no of doc repr>

See page 322 for the interpretation of input parameters.

(1) CLASSIFICATION DATE 01/15/74

(2) TOTAL ITEMS 11 (3) NODE 14

(4) PARTITIONS 3, E IS (5) 25, CELL SIZE (6) 4 ITEMS

(7) PHASE 1 CELL SWITCHES 10

(8) PHASE 2 REDUNDANCY 0

(9) GROUP	1	(10) NO. OF KEYS	204	(11) NO. OF ITEMS	2	(12) NODE	41	(13) TERMINAL CELL	
GROUP	2	NO. OF KEYS	179	NO. OF ITEMS	6	NODE	42		
GROUP	3	NO. OF KEYS	155	NO. OF ITEMS	3	NODE	43	TERMINAL CELL	

Table A.46

An example summary report generated by classification.

A.3.1.3.3 Sorting By Classification Numbers

File IV should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file VI are given below.

Sort of file IV : (CN,TYC)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
	FI,A	2	Integer classification number
1	FI,A	2	Type code

A.3.1.3.4 Complete Set of Keys Within A Cell

Program : GNFKFL

Issue file commands for input/output and temporary files.

```
/FILE fileVI,LINK=DSET21,FCBTYPE=SAM,RECFORM=V,RECSIZE=12,OPEN=INPUT
/FILE fileVII,LINK=DSET23,FCBTYPE=SAM,RECFORM=V,OPEN=OUTPUT
/FILE temp,LINK=DSET22,FCBTYPE=ISAM,RECFORM=V,RECSIZE=24,OPEN=OUTIN
```

Invoke the load module of program.

```
/EXEC LMGNFKFL
```

Enter input parameters.

<block factor>

Parameter Interpretation Table, GNFKFL

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<block factor> = b	4 * b + 4 specifies the maximum block size (in bytes) on the computer system.	5-digit integer

Output Statistics, GNFKFL

(1) Number of terminal nodes (cells) in the tree.
(2) Average number of keys per cell.

A.3.1.4 Hierarchical Classification Tree

A.3.1.4.1 Sorting By Classification Numbers

File VII should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file VIII are given below.

Sort of file VII : (CN)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	FI,D	2	Integer classification number
2	FI,A	2	Sequence number

A.3.1.4.2 Generation of Hierarchical Classification Tree

Program:TREE

Issue file commands for input/output and temporary files.

```
/FILE file-VIII,LINK=DSET12,FCBTYPE=SAM,RECFORM=V,OPEN=INPUT
/FILE file-IX,LINK=DSET15,FCBTYPE=SAM,RECFORM=V,OPEN=OUTPUT
/FILE temp-1,LINK=DSET13,FCBTYPE=ISAM,RECFORM=V,OPEN=OUTIN
/FILE temp-2,LINK=DSET14,FCBTYPE=ISAM,RECFORM=V,OPEN=OUTIN
/FILE temp-(2+I),LINK=DSET(15+I),FCBTYPE=ISAM,RECFORM=V,OPEN=OUTIN
```

where I=1,N, and N denotes the stratification number for the tree.

Invoke the load module of program.

```
/EXEC LMTREE
```

Enter input parameters.

<strat>Δ<level>Δ<no of rec>Δ<block factor>

Parameter Interpretation Table, TREE

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<strat>	See page 323	3-digit integer
<level>	Number of levels in the tree. This is one of the statistics produced by program CLASFY.	3-digit integer
<no of rec>	Number of records in file VIII. This can be obtained from the output of SORT routine.	6-digit integer
<block factor>	See page 327. The same blocksize should be specified.	

Output Statistics, TREE

(1) Number of keys in the hierarchical classification tree, counting duplicates.
(2) Number of nodes in the hierarchical classification tree.
(3) Number of records written to file IX.

Note: The first and third statistics are provided for the user for general information purposes, and may be disregarded if the user finds them of no interest. The second statistics is precisely the number of nodes in the classification tree generated by program CLASFY. In fact, the user should always compare these two control totals.

A.3.1.5 Node-to-key and Key-to-node Directories

A.3.1.5.1 Sorting By Classification Numbers

File IX should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file X are given below.

Sort of file IX : (NDN)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	FI,A	2	Integer classification number
2	FI,A	2	Sequence number

A.3.1.5.2 Generation of Node-to-key Directory

Program : NDTOKY

Issue file commands for input and output files.

```
/FILE {file-8} ,LINK=DSET14,FCBTYPE=ISAM,RECFORM=V,OPEN=INPUT
      {file-9}
/FILE file-X,LINK=DSET12,FCBTYPE=SAM,RECFORM=V,OPEN=INPUT
/FILE file-II,LINK=DSET16,FCBTYPE=ISAM,RECFORM=V,RECSIZE=24,
/OPEN=INPUT
/FILE file-XI,LINK=DSET15,FCBTYPE=SAM,RECFORM=V,OPEN=OUTPUT
/FILE file-XII,LINK=DSET13,FCBTYPE=SAM,RECFORM=V,OPEN=OUTPUT
```

Invoke the load module of program.

```
/EXEC LMNDTOKY
```

Enter input parameters.

```
<dev>
<level>Δ<strat>Δ<max term len>Δ<field pack>Δ<option>
```

Parameter Interpretation Table, NDTOKY

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<dev>	6- line printer M- some other device. (m>16) e.g. magnetic tape, disc.	2-digit integer
<level>	See page 329	3-digit integer
<strat>	See page 329	3-digit integer
<max term len> = L	Maximum number of characters per term.	3-digit integer
<field pack> = f	(2*L+4)*f+2 Specifies the maximum block size in bytes.	3-digit integer
<option>	0- Only Node-to-key table is produced. 1- Node-to-key table ordered by frequencies is also produced.	1-digit integer

Output Reports, NDTOKY

(1) Node-to-key Table in which the keys in each node are ordered alphabetically.
(2) Node-to-key Table in which the keys in each node are ordered by respective total frequencies.

Note: The second report is optional.

Output Statistics, NDTOKY

(1) Number of nodes in the hierarchical classification tree.
(2) Number of terminal nodes(cells) in the tree.
(3) Number of records written to file XII.

A.3.1.5.3 Alphabetical Sorting

File XII should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file XIII are given below.

Sort of file XII : (A,NDN)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	CH,A	2	First character of term.
2	CH,A	2	Second character of term.
.	.	.	.
.	.	.	.
.	.	.	.
MAXTL	CH,A	2	Last character of term.
MAXTL+1	FI,A	2	Integer classification number.

A.3.1.5.4 Generation of Key-to-node Directory

Program: KYTOND

Issue file commands for input/output and temporary files.

```
/FILE file-XIII,LINK=DSET12,FCBTYPE=SAM,RECFORM=V,OPEN=INPUT
/FILE file-XIV,LINK=DSET13,FCBTYPE=ISAM,RECFORM=V,OPEN=OUTPUT
/FILE temp,LINK=DSET14,FCBTYPE=ISAM,RECFORM=V,OPEN=OUTIN
```

Invoke the load module of program.

/EXEC LMKYTOND

Enter input parameters.

<dev>
<level>Δ<strat>Δ<max term len>

See page 330 for the interpretation of input parameters.

Output Reports, KYTOND

(1) Key-to-node Table.
(2) List of key types ordered by respective node frequencies. Each key type is associated with its respective node frequency and type code.

Output Statistics, KYTOND

(1) Number of key types.

A.3.1.6 Classified Data-base

A.3.1.6.1 Generation of Classified Data-base

Program: MRGCLY

Issue file commands for input and output files.

/FILE file-III,LINK=DSET12,FCBTYPE=ISAM,RECFORM=V,OPEN=INPUT
/FILE {file-20} ,LINK=DSET14,FCBTYPE=SAM,RECFORM=V,OPEN=INPUT
      {file-6 }
/FILE file-0,LINK=DSET12,FCBTYPE=BTAM,RECFORM=V,BLKSIZE=4096,OPEN=INPUT
/FILE file-XV,LINK=DSET15,FCBTYPE=BTAM,RECFORM=V,BLKSIZE=4096,OPEN=OUTPUT

Invoke the load module of program

/EXEC LMMRGCLY

Enter input parameters.

<max term len>Δ<fld pack>
[<doc no>]
.
.
.
.

Parameter Interpretation Table, MRGCLY

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<max term len> = L	Maximum number of characters per term.	3-digit integer
<fld pack>=F	(2+L)*F+12 Specifies the maximum block size in bytes.	5-digit integer
<doc no>	Identification number of the document whose representative has been deleted or merged. Note that the corresponding document representative does not exist in file 6 or 20.	6-digit integer.

A.3.1.6.2 Sorting By Classification Numbers

File XV should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file XVI are given below.

Sort of file XV : (CN,DN)

CONTROL FILED #	FORMAT	LENGTH	DESCRIPTION
1	FI,A	2	Integer Classification Number
2	FI,A	4	Document identification number
3	FI,A	2	Sequence number

A.3.1.6.3 Generating The Classified Data-base Report

Program: PRTCLF

Issue file commands for input and output files.

```
/FILE file-XV,LINK=DSET21,FCBTYPE=BTAM,RECFORM=V,BLKSIZE=4096,OPEN=INPUT
/FILE temp,LINK=OUT
```

Note: Temp denotes the file into which the report is to be stored.

Invoke the load module of program.

/EXEC LMPRTCLF

Enter input parameters.

<max term len>Δ<option>
<strat>

Parameter Interpretation Table, PRTCLF

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<max term len>	Maximum number of characters per term.	2-digit integer
<option>	1 should be specified.	1-digit integer
<strat>	See page 323	2-digit integer

A.3.1.6.4 Sorting By Canonical Classification Numbers

File Temp containing the report generated should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file are given below. Note that the sorted file can be printed using the PRINT command with the option SPACE=E

Sort of Classified Data-base Report: (CAN(CN),DN)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	FI,A	16	Canonical classification number.
2	FI,A	2	Document identification number.
3	FI,A	2	Sequence number.

Note: Control fields for this sort application are all contiguous in the physical record and their physical ordering is the same as the byreal ordering shown in the table. Control field 1 starts at byte 139.

A.3.2 Key Classification

A.3.2.1 Classification Tree For Key Classification

A.3.2.1.1 Sorting By Type Codes

File IV should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file V are given below.

Sort of file IV : (TYC,CN)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	FI,A	2	Type code.
2	FI,A	2	Integer classification number.

A.3.2.1.2 Generating Key Surrogates

See section A.2.8.4

A.3.2.1.3 Classification Tree

Program: CLASFY

Issue file command for input and output files.

```
/FILE file-XVII,LINK=DSET21,FCBTYPE=ISAM,RECFORM=V,RECSIZE=528,
/OPEN=INOUT
/FILE file-XVIII,LINK=DSET20,FCBTYPE=SAM,RECFORM=V,RECSIZE=16,
/OPEN=OUTPUT
```

Issue file commands for temporary files.

See page 323

Invoke the load module of program.

```
/EXEC LMCLASFY
```

Enter input parameters.

See page 323 for the specification and interpretation of input parameters. (The user is advised to specify a very small sensitivity factor).

A.3.2.2 Affinity Dictionaries

A.3.2.2.1 Sorting By Type Codes

File XVIII should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file XIX are given below.

Sort of file XVIII : (TYC)

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	FI,A	4	Type code

A.3.2.2.2 Generating Affinity Dictionary For Automatic Search And Retrieval System

Program: KTCLND

Issue file commands for input and output files.

```
/FILE file-XIX,LINK=DSET22,FCBTYPE=SAM,RECFORM=V,RECSIZE=16,OPEN=INPUT
/FILE file-XIV,LINK=DSET21,FCBTYPE=ISAM,RECFORM=V,OPEN=INOUT
/FILE file-XXI,LINK=DSET23,FCBTYPE=SAM,RECFORM=V,OPEN=OUTPUT
```

Invoke the load module of program.

```
/EXEC LMKTCLND
```

Enter input parameters.

```
<dev>
<max term len>Δ<level-1>Δ<level-2>Δ<strat-1>Δ<strat-2>
```

Parameter Interpretation Table, KTCLND

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<dev>	See page 330	2-digit integer
<max term len>	Maximum number of characters per term.	2-digit integer
<level-1>	Number of levels in the document classification tree.	2-digit integer
<level-2>	Number of levels in the key classification tree.	2-digit integer
<strat-1>	Stratification number for the document classification tree.	2-digit integer
<strat-2>	Stratification number for the key classification tree.	2-digit integer

Output Report, KTCLND

(1) Key-cell-node Table.

Output Statistics, KTCLND

(1) Number of key types.

A.3.2.2.3 Sorting By Canonical Classification Numbers

File XXI should be sorted using SORT package. Control fields specifying the ordering of data in the sorted file XXII are given below.

CONTROL FIELD #	FORMAT	LENGTH	DESCRIPTION
1	FI,A	2	First digit in the canonical number.
2	FI,A	2	Second digit in the canonical number.
.	.	.	.
.	.	.	.
.	.	.	.
<level-2>	FI,A	2	Last digit (or zero when padded)

Note: See page 330 for the interpretation of <level-2>.

A.3.2.2.4 Generating Affinity Dictionary For Manual Use

Program: PRTCLF

Issue file command for input file.

/FILE file-XXII,LINK=DSET21,FCBTYPE=SAM,RECFORM=V,OPEN=INPUT

Invoke the load module of program.

/EXEC LMPRTCLF

Enter input parameters.

<max term len>Δ<option>
<dev>Δ<level-1>Δ<level-2>

Parameter Interpretation Table, PRTCLF

PARAMETER NAME	DESCRIPTION	INPUT FORMAT
<max term len>	Maximum number of characters per term.	2-digit integer
<option>	0 should be specified.	1=digit integer
<dev>	See page 330	2-digit integer
<level-1>	See page 337	2-digit integer
<level-2>	See page 337	2-digit integer

Output Report, PRTCLF

(1) Cell-key-node Table. This report is referred to as Affinity Dictionary.

EXAMPLE

The following input parameters have been used in generating an Affinity Dictionary for the Illustrative Data-base.

48Δ0
06Δ07Δ08

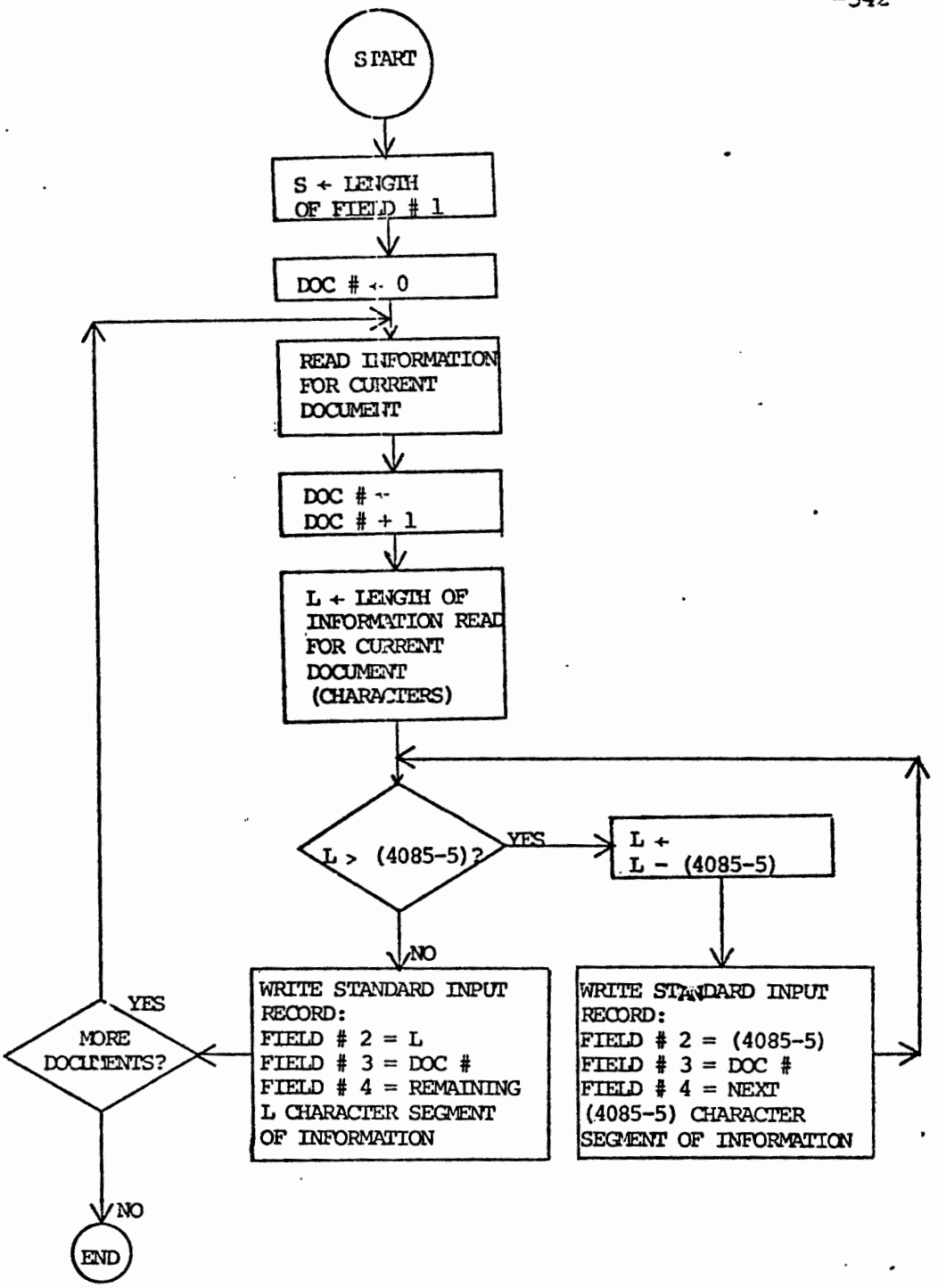


Figure 3.1

General Flowchart of User Written Program
To Create The Standard Formatted Text File

Subroutine number	Subroutine name	Subroutine number	Subroutine name
1	ADD	25	PHRSUB
2	BSRH1	26	PPFIL
3	BSRH2	27	PRCBUC
4	BSRH3	28	PRCODE
5	BSRT1	29	PRDATA
6	BSRT2	30	PRDIR
7	BSRT3	31	PRDSEN
8	CHANGE	32	PRFILE
9	CHGPHR	33	PRINTREC
10	CHRANG	34	PRSRCD
11	CNSEN	35	PRSUR
12	CONV	36	PRTEXT
13	CONVER	37	PRTFIL
14	DECOMP	38	PRTSNO
15	DELETE	39	RECOD
16	ERASE	40	SRTCOD
17	EXTPHR	41	SRTDNO
18	EXTRCT	42	SORTST
19	KEYLEN	43	SRBUF
20	NORM	44	SRSTOP
21	NORMAL	45	VALID
22	PDIREC	46	WRBUF
23	PFIL	47	WRWD
24	PHRASE		

Table B.1 Subroutine Library

convention enables the programmer to locate the source code of any program in a simple manner.

In order to be able to invoke a (main) program, the programmer must create a load module for the program. To achieve this, he must first read the source code of the program and the source codes of the subroutines it calls for, if any, from the tape to the disc. Once, the program and the subroutines it calls for are on disc, they can be compiled and link edited through the operating system of his computer to create a load module, which may be given any name consisting of any string of one to eight characters. Now, the programmer may present the name of the load module to the user so that he can invoke the program.

APPENDIX C

Program Descriptions and Flowcharts

C.1 Introduction

This appendix contains descriptions of operations of a series of programs that have been devised and written for the implementation of the indexing and classification system described in Chapter III. All programs have been written in FORTRAN IV of the Univac Series 70/46 Computer, and are input/output bound, using unformatted data sets.

Each section describes the operation of a particular program and gives also the flowchart of the program.

C.2 Program STDFIL

Purpose of Program: To generate a system directory to the Standard Formatted Text File.

Program Description:

The program reads the standard formatted text records of each document and writes them onto the output file as ISAM records. At the same time, it also writes the document's number and the first of the ISAM keys of its correspoinding standard text records onto a temporary file. On encountering the file end mark of input file, the program writes the number of documents in the

collection onto the Standard Formatted Text File and copies the temporary file generated to the Standard Formatted Text File.

In order to take advantage (in terms of input/output time) of the computer system on which the program is being run, the program writes each standard text record into a physical block whose size is specified by the user. Figure C.1 presents a general flowchart of program STDFIL. Figure C.2 shows also the access relationship between the system directory and the Standard Formatted Text File.

C.3 Program INDEX.

Purpose of Program: To extract words and standard and user specified phrases.

Program Description:

The program starts by repositioning the input and output files. If no document has been processed yet, the repositioning consists of simply rewinding both files. If some documents have been processed yet, then it repositions the input and output files by means of the first and third statistics generated previously and supplied as input parameters. The repositioning of input file is achieved by skipping the required number of

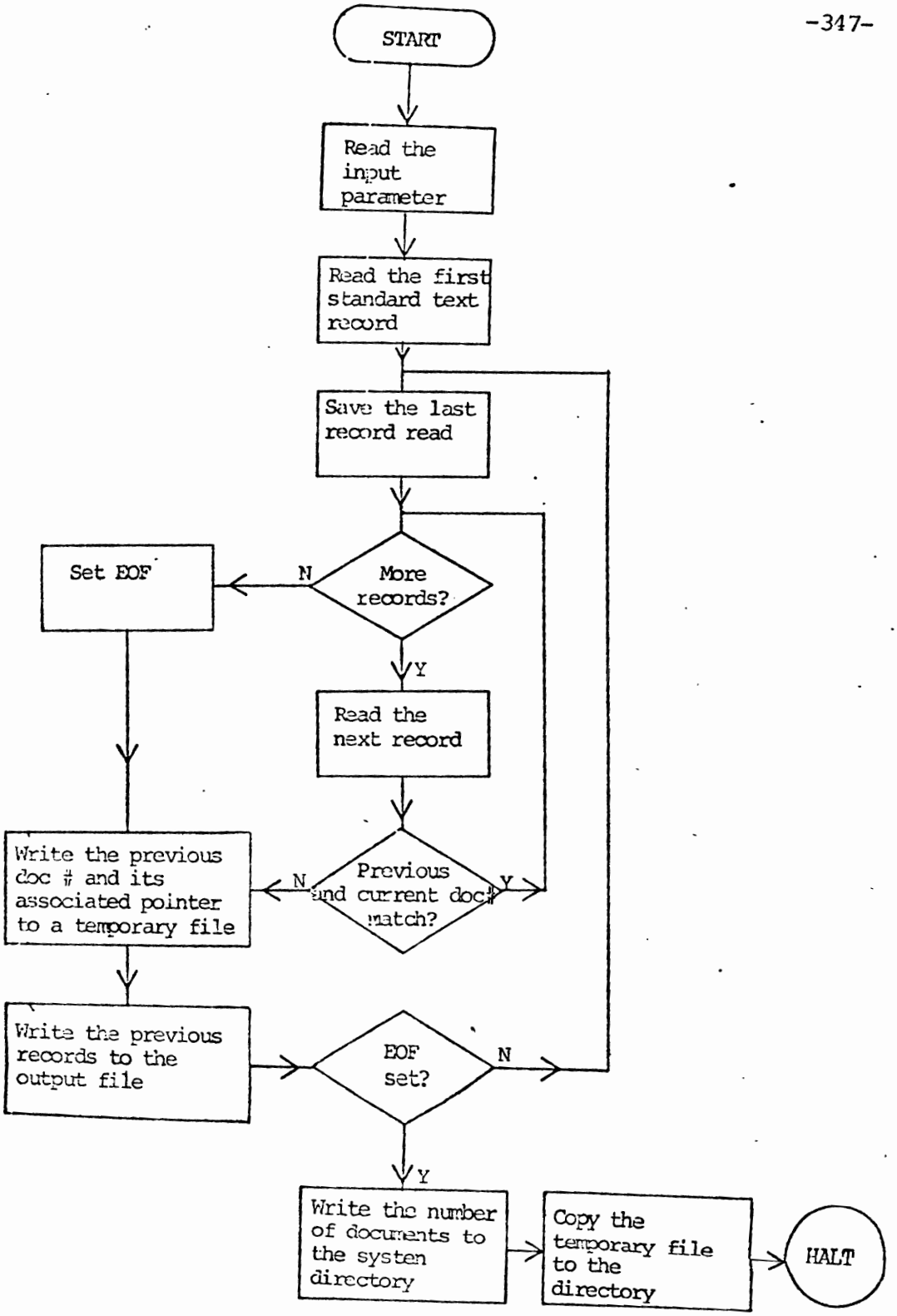


Figure C.1

General Flowchart of Program STDFIL

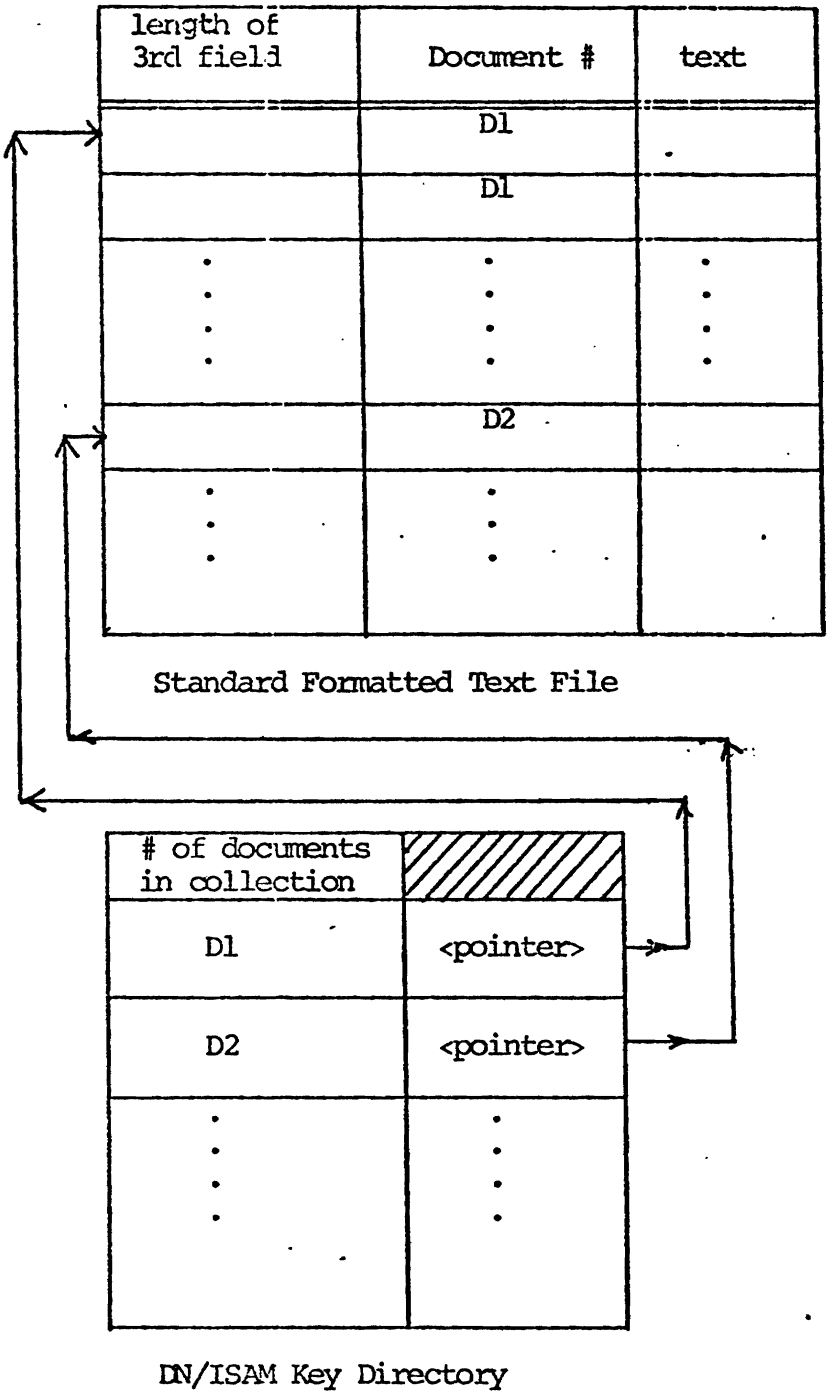


Figure C.2

Illustration of Access Methods

standard formatted text records specified by one of the input parameters. The output file is repositioned by skipping the required number of records and overwriting the file end mark.

After the repositioning is succeeded, the program reads the next standard text record on input file into the main memory as a string of characters, and starts decomposing it into words or phrases. Each extracted word (i.e. a string of non-word delimiting characters) is compared against the list of month names. If a match is found, the program parses the environment of the extracted word in text and substitutes the extracted date-phrase for the word in the word buffer. If there is no match and analysis of capitalization is possible, the program checks if the first letter of the word is a capital letter. If this is the case, it parses the text and extracts the sequence of adjacent words starting with capital letters. The word buffer is in turn overwritten by this phrase. If the word is not part of a phrase (i.e. date-phrase, or a sequence of adjacent words starting with capital letters), several tests are performed to determine whether or not it is to be retained as a candidate index word. The first of these tests is to examine its length. The word is ignored if its length is less than three characters; otherwise the

postfixing characters are examined and dropped if they are "special characters" (i.e. any character other than the 26 alphabetics, 10 numerics, and the "blank character" is defined to be a special character). If the truncation of "special characters" reduced its length to less than three characters, it is rejected from further consideration.

After the above tests the word is compared to the Stop List of function words, and ignored if it is found to be in the Stop List. However, the Stop List comparison test is optional and controlled by the user. Next, there is a process of stem analysis which drops or modifies the suffix of the word. The suffix dropping routine, which is a slightly improved version of the one used by General Inquirer (St66-2) and Litofsky(Li69), removes a number of different suffixes. A flowchart of this routine is given in Figure C.3, which shows the set of suffixes and the exact context in which they are dropped off. The normalized word is then written into the word buffer.

If the program encounters an occurrence of period while parsing the text, it checks its environment in an attempt to extract an abbreviated name-phrase (e.g. MRS. BROWN, J. HOWARD), or a date-phrase. The occurrence of period in a name phrase is distinguished from its

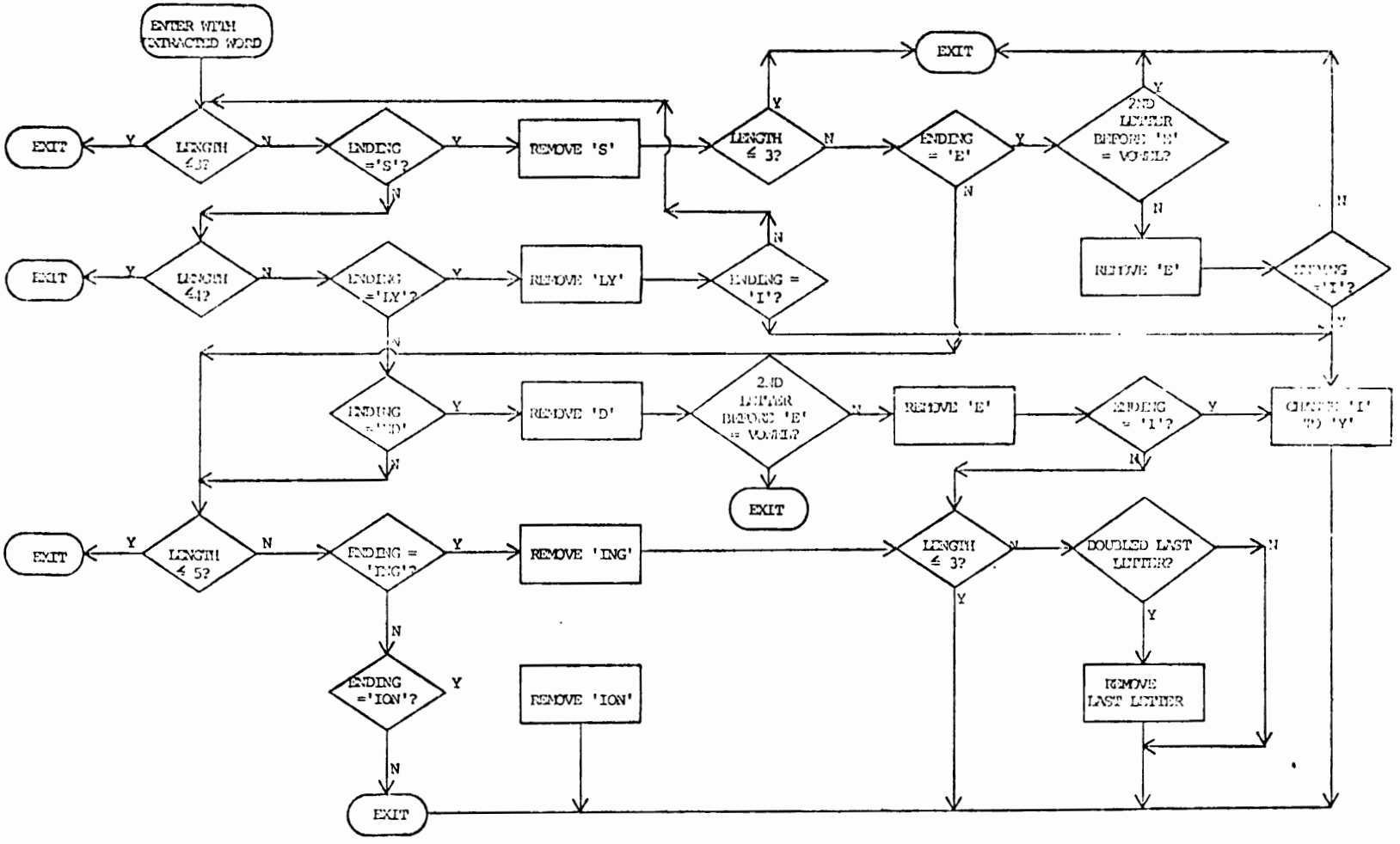


Figure C.3

Suffix Deletion Routine

occurrence as a sentence delimiter by the convention that the last word of a sentence should not be a single character or an abbreviated name-phrase. Time-phrases are recognized by testing the environment of colons in text.

If the word buffer contains a phrase, it is written into the sentence buffer and a new indexing cycle (i.e. extraction of a word or phrase) is started. If it contains a (normalized) word and the system is provided with a phrase dictionary, indexing cycle continues with phrase analysis. If no dictionary is supplied, the word is written into the sentence buffer and the new indexing cycle is started.

Phrase analysis in conjunction with a Phrase Dictionary is accomplished by directions that are part of the phrase dictionary entry whose first field contains the word extracted from the text. Therefore, first a binary search is done to determine if there is an entry of the dictionary whose first field contains the extracted word. If there is no such entry, the phrase analysis is aborted and the word is written into the sentence buffer; otherwise further analysis is done in an attempt to form a phrase combining the extracted word and required words from the sentence buffer in accordance with the directions in the entry. If this second

analysis results positively, the component words in the sentence buffer are replaced by the phrase itself; otherwise the phrase analysis is aborted and the word is written into the sentence buffer. For instance, the dictionary entry:

INDEX = 2 COMPUTER + & 1 MANUAL + #

would enable the system to recognize all occurrences of the phrases: "Computer Indexing", "Computer Aided Indexing", and "Manual Indexing". Similarly, by means of the entry:

TREATY = 3 NORTH + ATLANTIC + ALLIANCE + #

all occurrences of the phrase "North Atlantic Alliance" can be recognized.

Upon encountering the end of a sentence in text (i.e. the occurrence of a sentence-delimiter that signals the end of a sentence), each term in the sentence buffer is written to the output file with the following four fields of information.

1. Length of the term (in characters).
2. The identification number of the document containing the term. This number is taken from the input record. Namely, the standard text

record.

3. The identification number of the sentence (relative to the beginning of the document) containing the term.
4. The term-address assigned to the term.

Each term is written to the output file as a variable length field of information, since the system does not know the maximum term length to be attained prior to the extraction of all terms from the collection of documents. However, the term fields of variable length will have to be transformed to fields of a "fixed length", since fixed length terms are indispensable for the purpose of alphabetical sorting.

The assignment of sentence (identification) numbers starts with the first sentence of the document, which is assigned number 1, and ends with the last sentence of the document. A sentence number register is used for this purpose, which is reset to 0 at the beginning of the document and incremented by 1 upon the occurrence of each sentence-delimiter which signals the end of a sentence.

In the same manner, a term-address register is used for the assignment of term-addresses, which is reset to 0 on encountering a new sentence and incremented by 1 after the extraction of each term. The current value of the register

is then assigned to the term. The register is always incremented by 1 on the occurrence of a word-delimiter (consequently after the extraction of a word even though it is not retained as a candidate index word). However, before the next indexing cycle starts the program updates the register's value by incrementing it again by 1, if the word-delimiter that caused the extraction of the word was not a "blank", or by M(M≥1) if the extracted term is immediately followed by "special character(s)".

Figure C.4 shows the flowchart of the program and subroutines involved.

C.4 Program PHRGEN.

Purpose of Program: Automatic phrase generation.

Program Description:

The program compares each record of the input file with the one next to it and generates the phrase by concatenating the second fields (i.e. the fields containing the terms) of those records whose third and fourth fields are identical and whose correspoinding term-addresses (i.e. the contents of last fields) differ only by 1. The program does not attempt to generate a phrase whose length will be longer than the user-specified upper bound. After the phrase is generated, the program checks if it starts or/and ends with the prepositions "the" or "of". If it does, then the

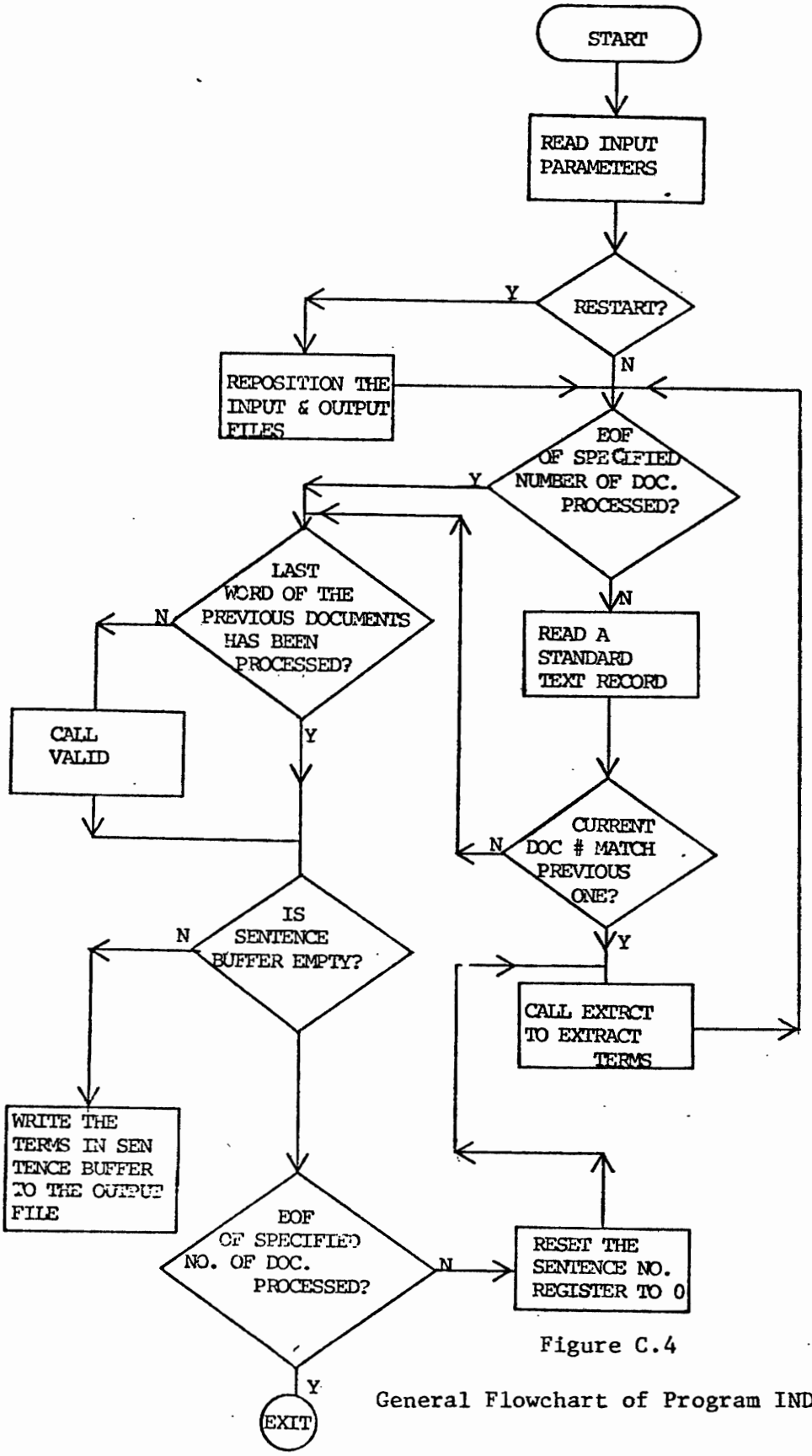


Figure C.4

General Flowchart of Program INDEX

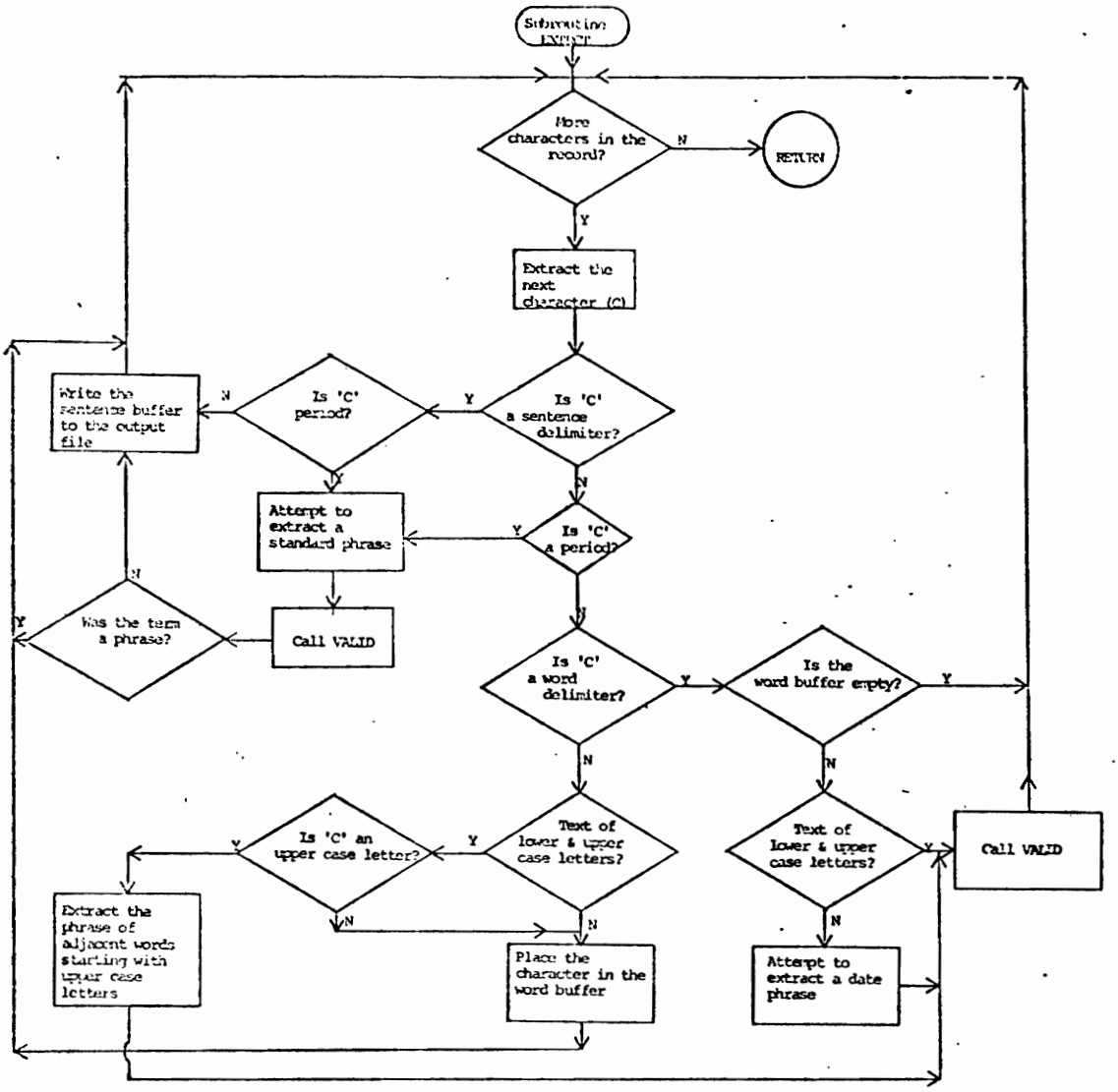


Figure C.4

-CONTINUED-

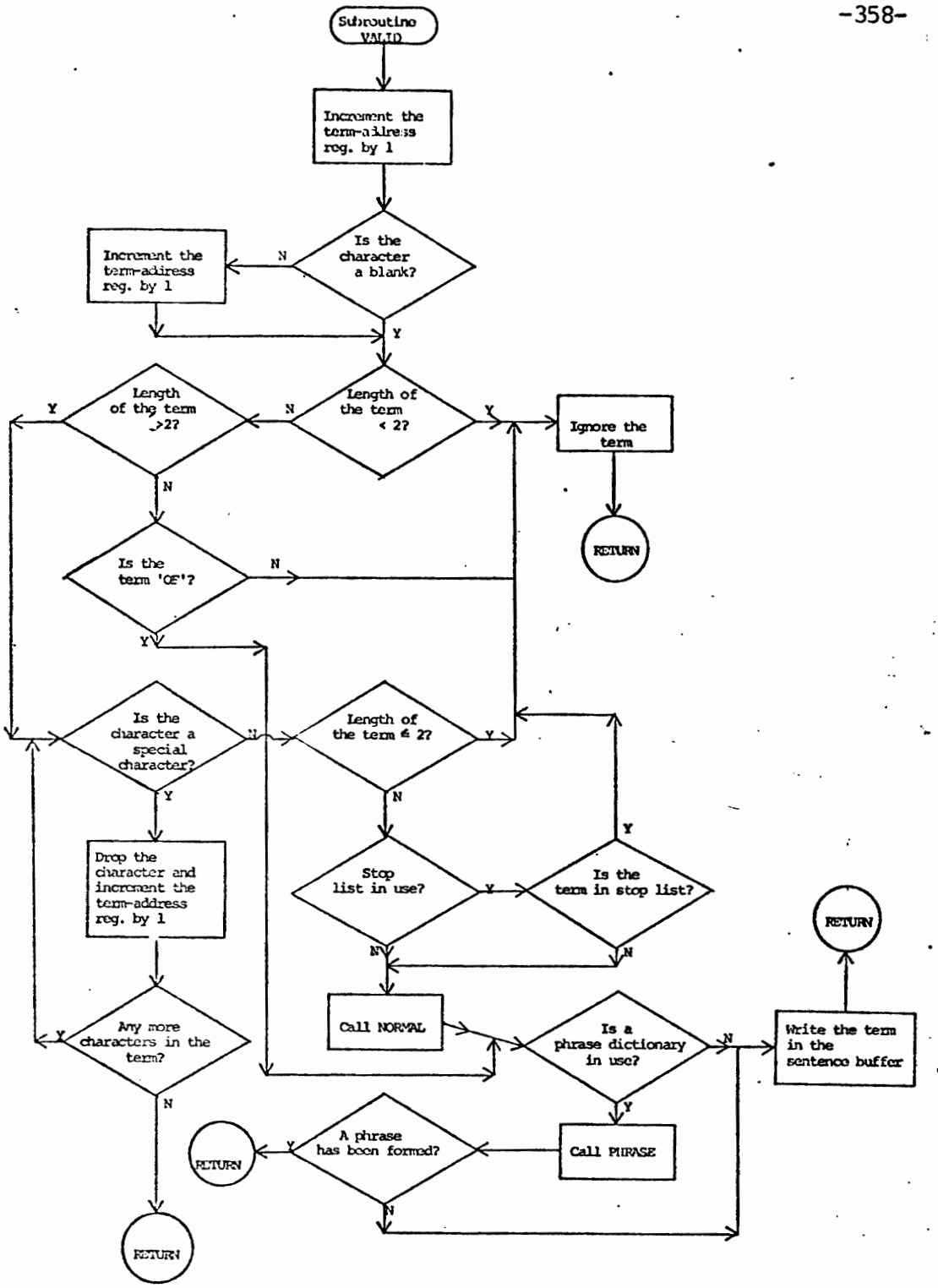


Figure C.4
-CONTINUED-

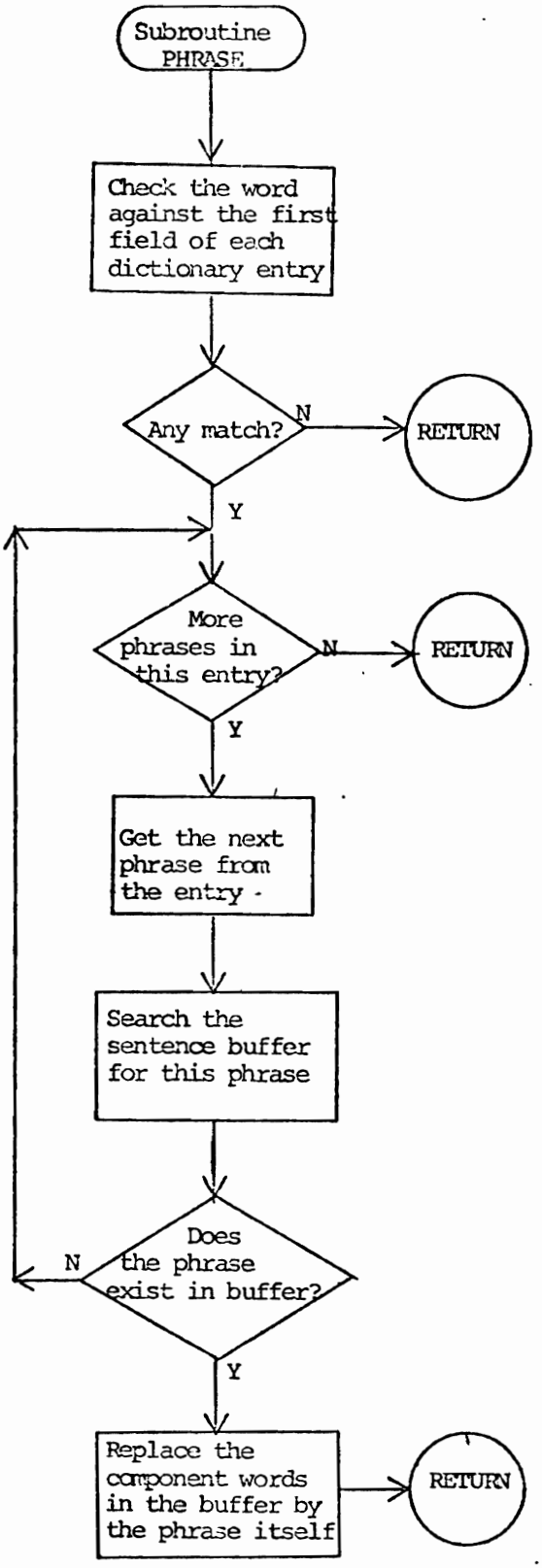


Figure C.4 -CONTINUED-

preposition(s) is (are) chopped off from the phrase and the number of remaining components is examined. If the preposition dropping transforms the phrase into a single word, it is rejected from the index phrase candidacy.

The candidate index phrase is written onto the output file with the associated information. Document and sentence identification numbers, and the phrase-address are taken from the input records while the number of phrase components is determined simply by counting the terms forming the phrase.

Each phrase is written to the output file as a variable length field, since the program does not know the maximum phrase length to be attained prior to the phrase generation process. Later, the phrase fields of variable length will be transformed to fields of a "fixed length", since fixed length phrases are indispensable for the purpose of alphabetical sorting.

Figure C.5 shows the flowchart of Program PHRGEN.

C.5 Program VARFIX

Purpose of Program: Transformation of variable length terms to fixed length terms.

Program Description:

Transformation from variable length term fields into fixed length fields is achieved by postfixing the second

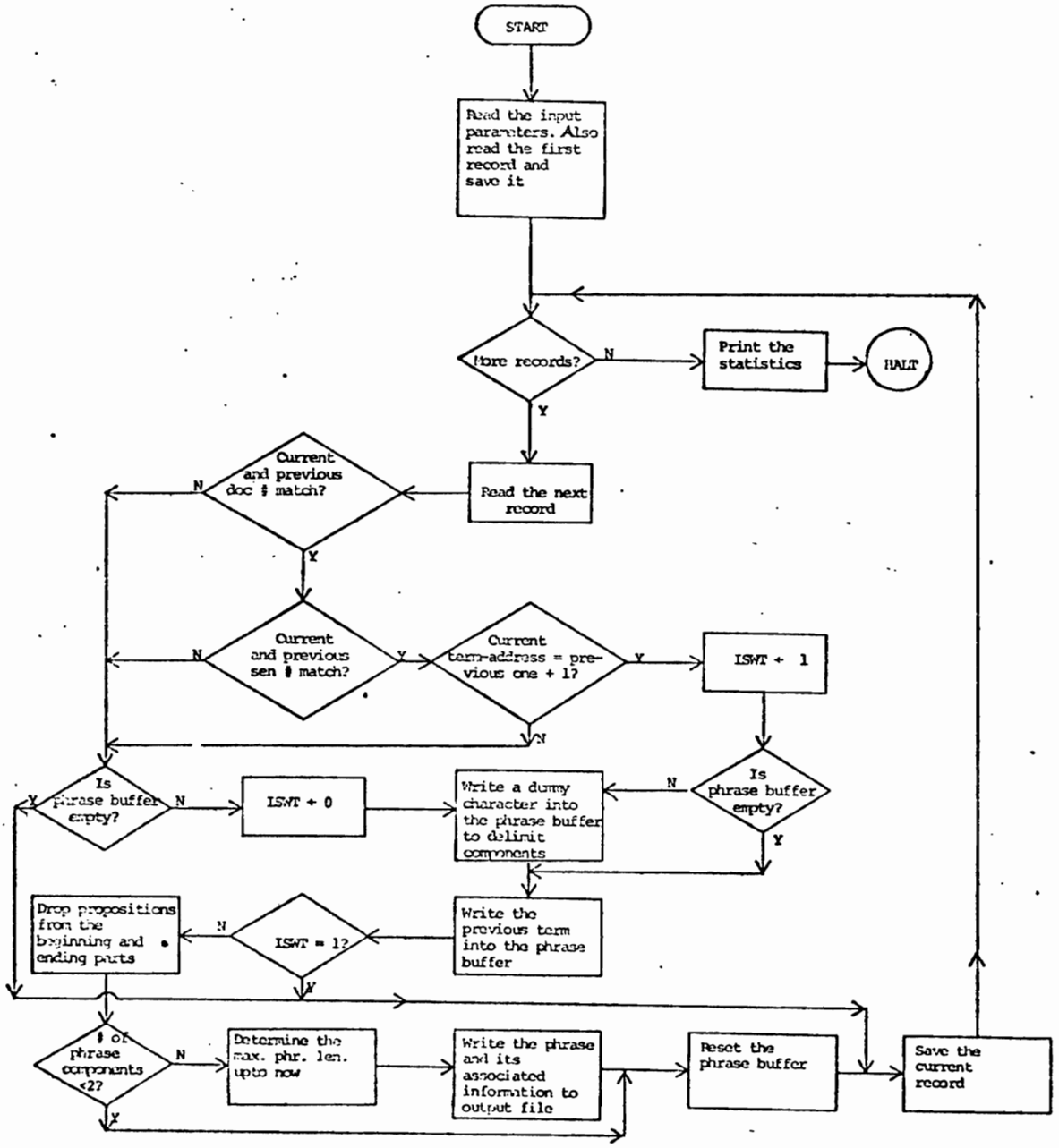


Figure C.5

General Flowchart of Program PHRGEN

field (i.e. the field containing the term) of each input record with a required number of blanks. That is, if the "maximum term length" is denoted by "MAXTL", then a term of length "L" is postfixed by "MAXTL-L" blanks.

If the input file is the Phrase Token Directory, each phrase token is also assigned a sequentially generated code. The program performs the latter task by prefixing the input record with a "code" field, containing the so-called token code. The characteristics of such a code from the point of system is that the value of a token code associated with a record is identical to the value of ISAM key which is assigned to the record by the Data Management System. This feature will enable the system to access the phrase tokens through respective token codes by using ISAM access method.

Figure C.6 shows the flowchart of Program VARFIX.

C.6 Program UNWRDS

Purpose of Program: Production of term or key types.

Program Description:

The operation of the program is controlled by three input parameters: <option>=PRM, <sub-option>=DIR, and <freq cut-off>=FRLMT. The program compares each record of input file with the one next to it and accumulates the total frequency and document frequency as long as the current and previous keys match. In addition, if PRM=0 or 1, the token

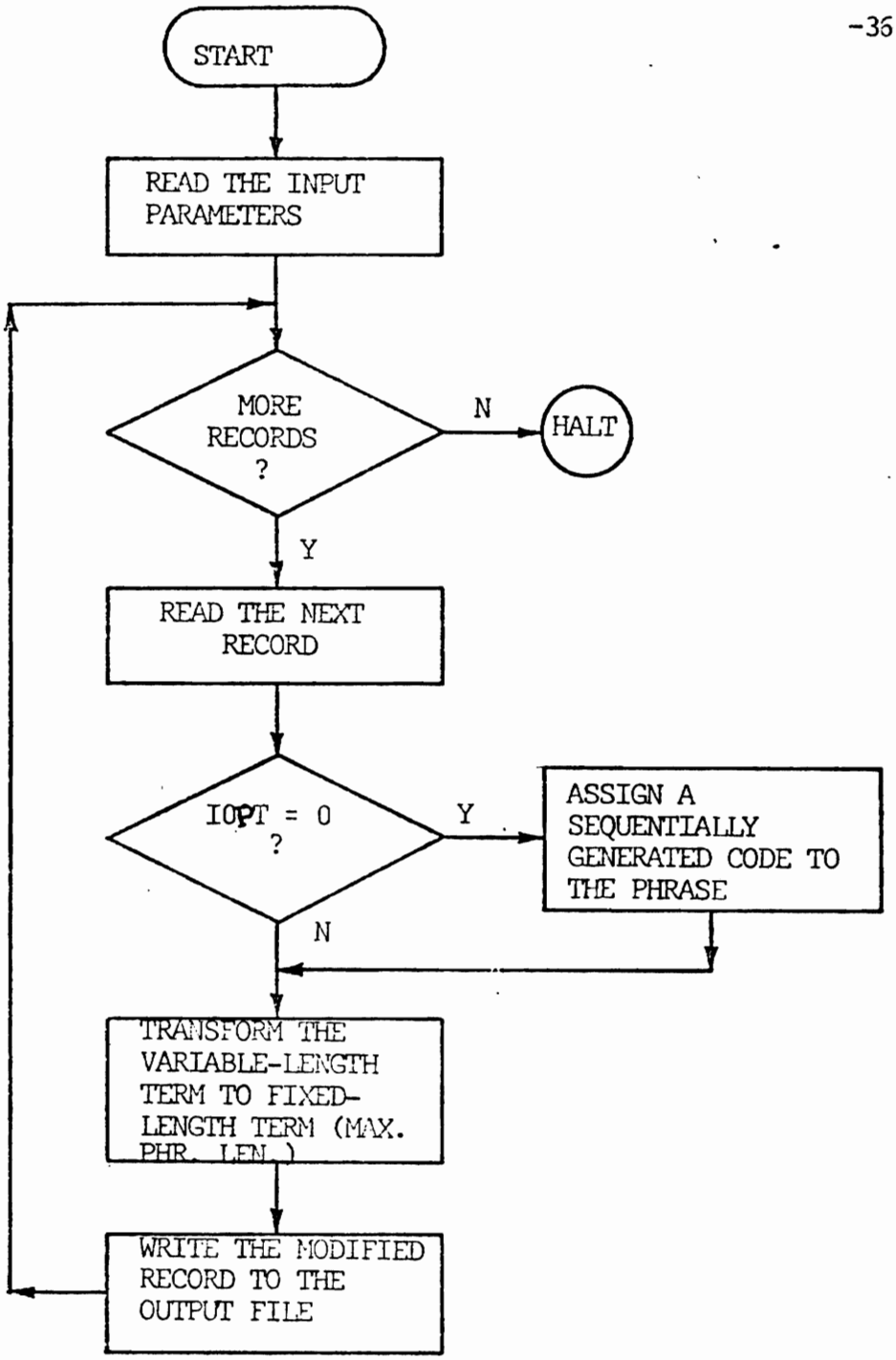


FIGURE C.6
GENERAL DESCRIPTION OF PROGRAM VARFIX

codes, document and sentence identification numbers corresponding to the key are saved in a stack. Whenever, adjacent keys do not match, a sequentially generated type code is assigned to the key type and both frequency counters are reset. At the same time the key type, the type code, frequencies, and information in the stack, if any, are written to the output file containing key types. If PRM=0 or 1, the program also writes the address (i.e. ISAM key generated by the Data Management System) of the first of the output records associated with the key type to the ISAM Key Directory. If DIR=3 (PRM=1) is specified, the program disregards those records of the input file, which have a 'delete mark', and generates an updated version of the input file in that each record is assigned a new sequentially generated token code. If PRM=0 is specified, a modified version of input file is further generated by prefixing each record with a code field containing the sequentially generated token code.

If PRM=2 is specified, the first three fields of every record in the input file are output to file 10 after the frequency field has been replaced by the respective type code of the key. Moreover, if also a($a \neq 0$) is specified for the parameter FRLMT, then a second version of file 10 is also generated in which keys with total frequencies less than or equal to A are not written.

The flowchart of program UNWRDS is given in Figure C.7. Figures C.8 and C.9 describe also access relationship between the respective files.

C.7 Program MRWRPH.

Purpose of Program: To merge the word and phrase tokens.

Program Description:

The program starts by reading a phrase token from file 3 or 4 and copying all the term tokens from the Word Token Directory to the output file until it comes over the term token which constitutes the first component of the phrase. It, in turn, ignores the component terms of the phrase by skipping the required number (i.e. the number of components of the phrase) of records from the Word Token Directory. However, if the record containing the phrase token read contains a "delete mark", it is ignored and the next record is read from file 4.

The program also determines the maximum length of terms written onto the output file and compares it against the one specified as an input parameter. If they do not match, then this new maximum is printed out for the information of the user.

Figure C.10 contains the flowchart of program MRWRPH.

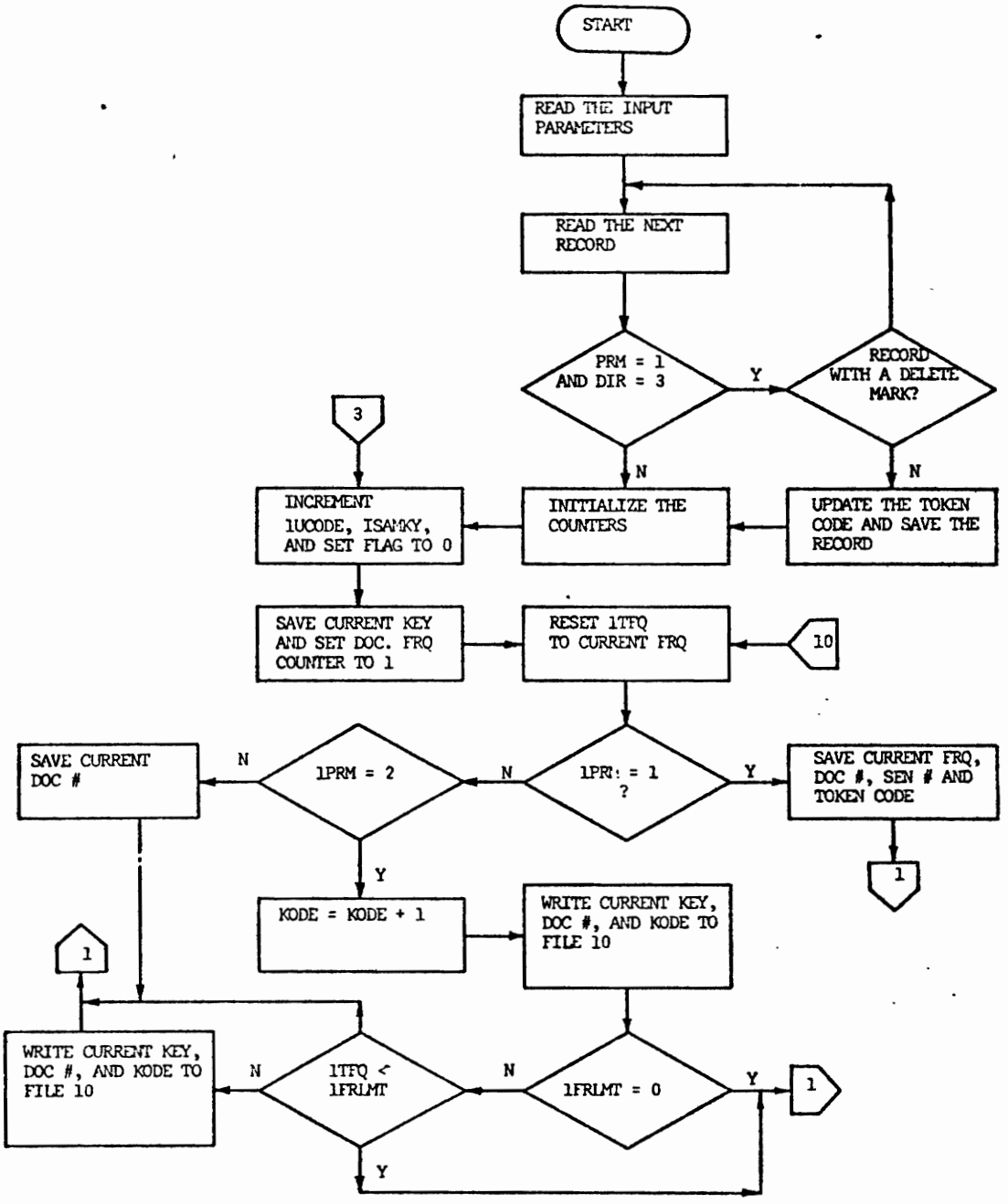


Figure C.7

General Flowchart of Program UNWRDS

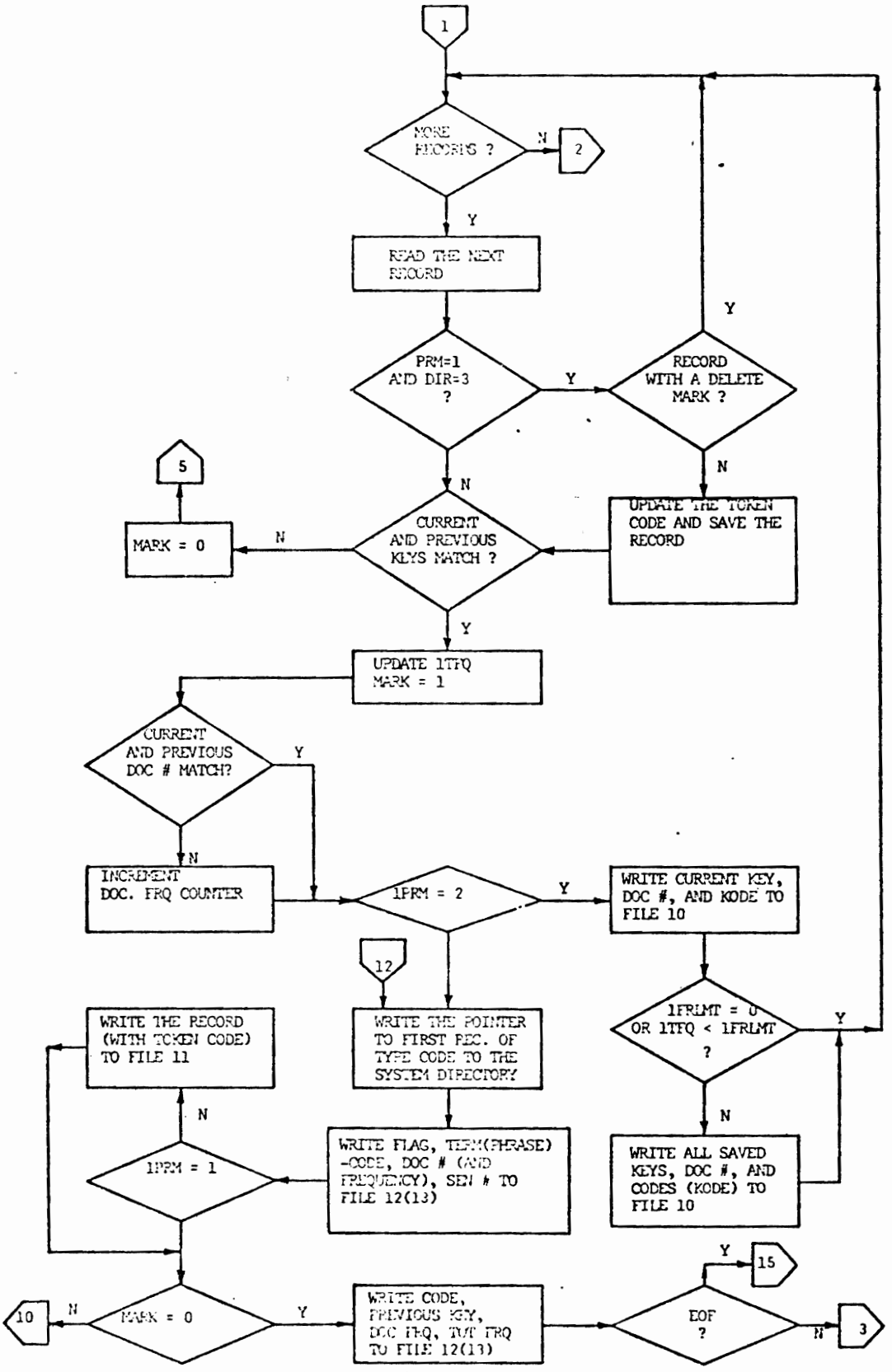


Figure C.7 -CONTINUED-

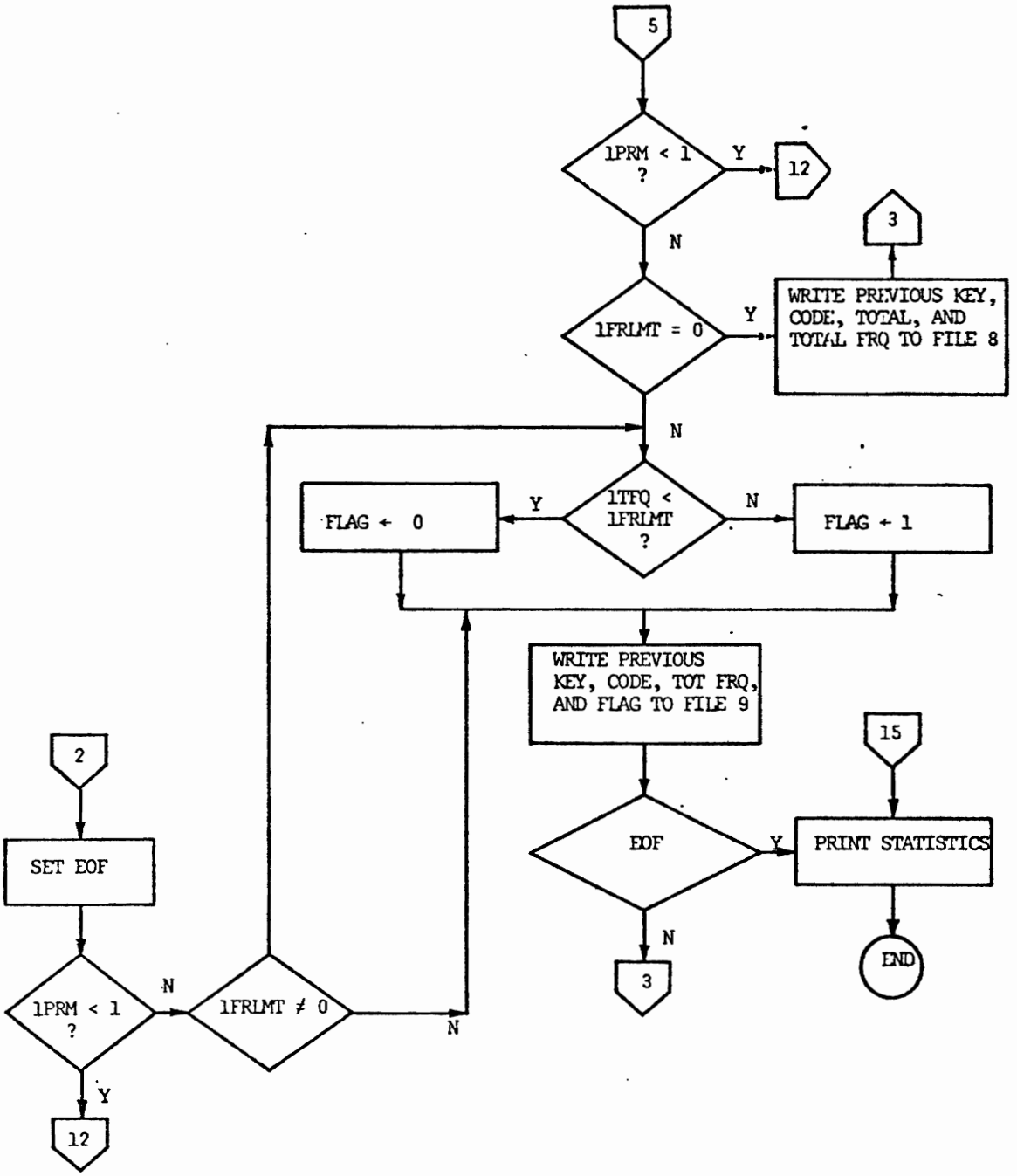


Figure C.7
-CONTINUED-

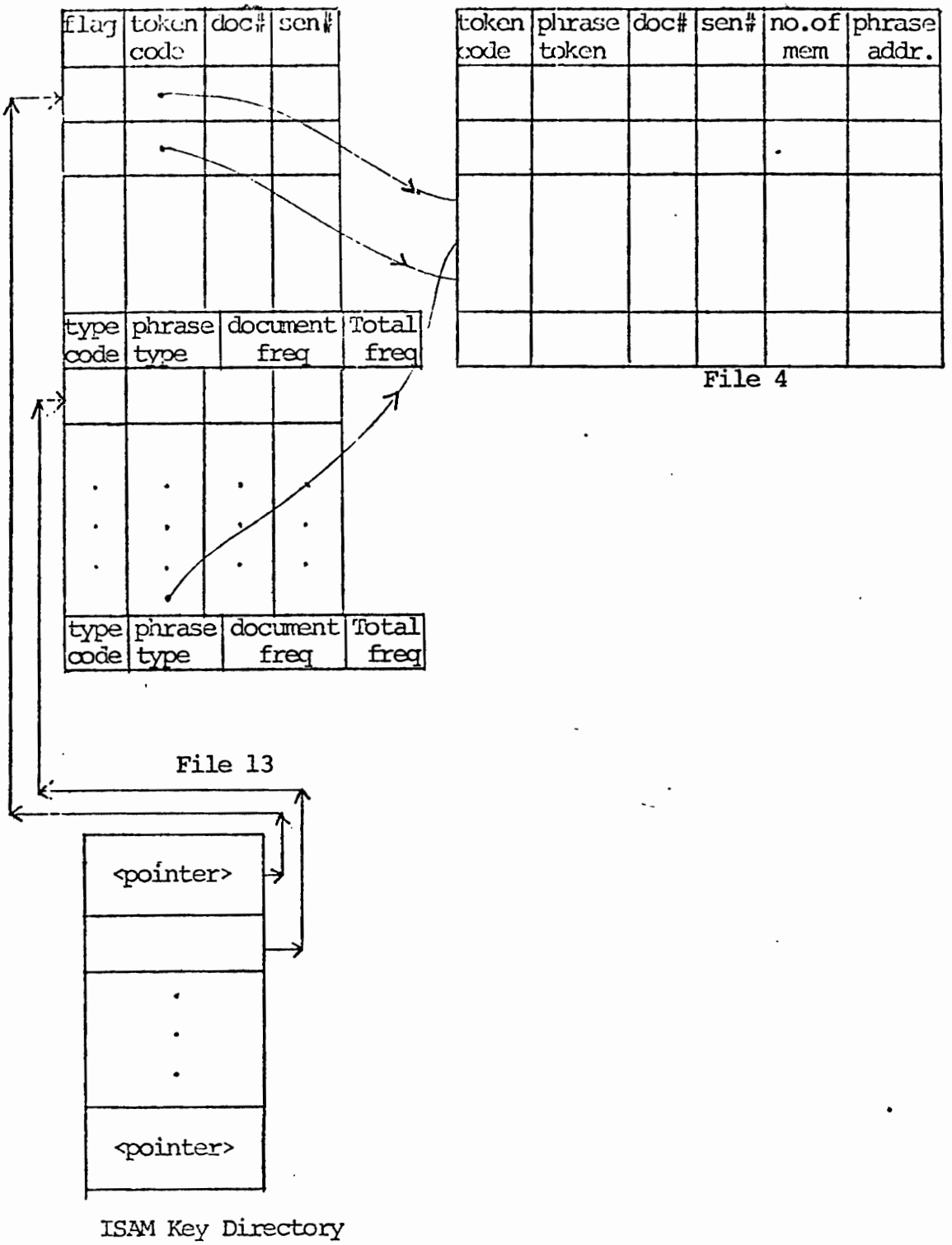


Figure C.8

Illustration of Access Methods

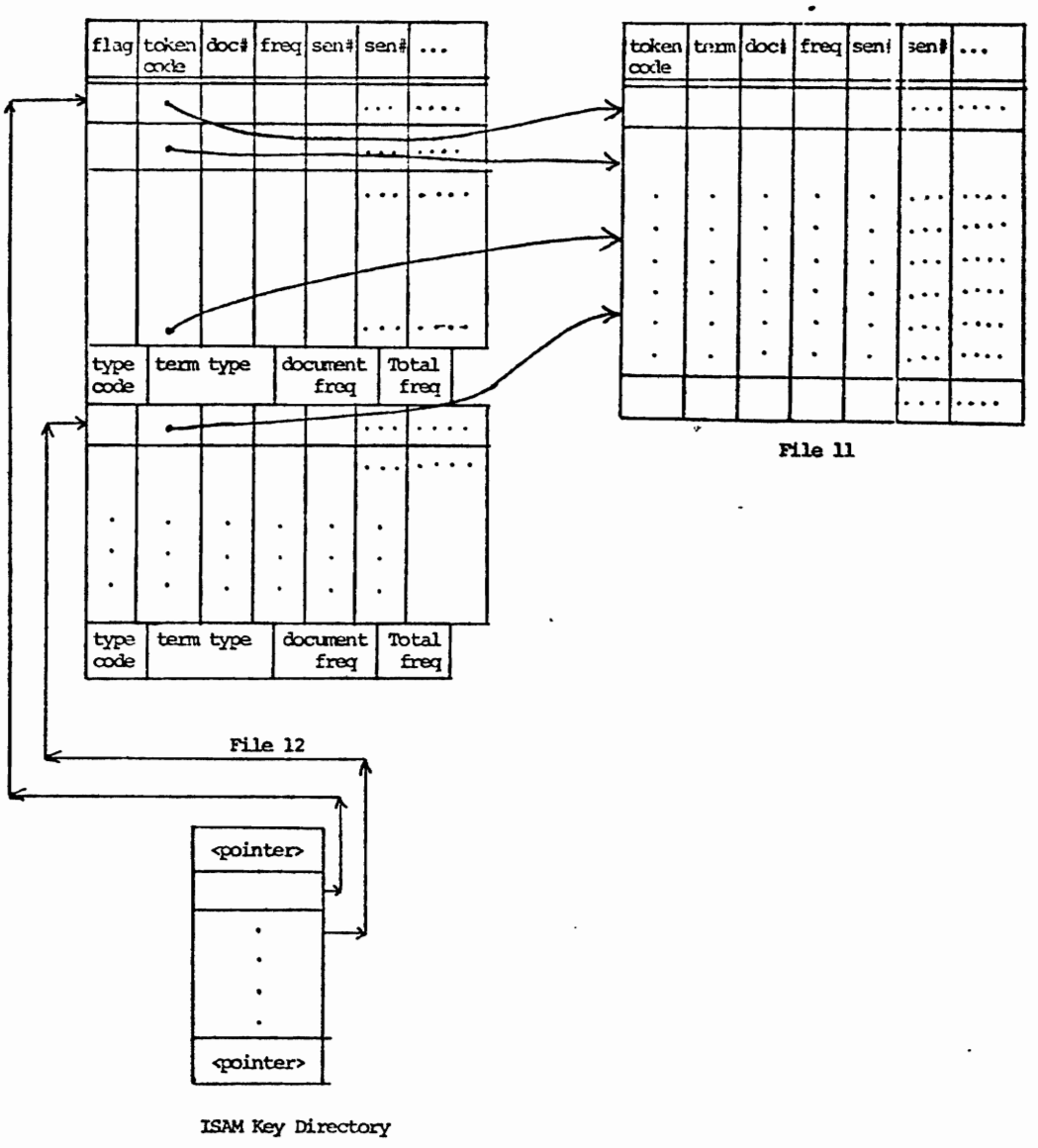


Figure C.9

Illustration of Access Methods

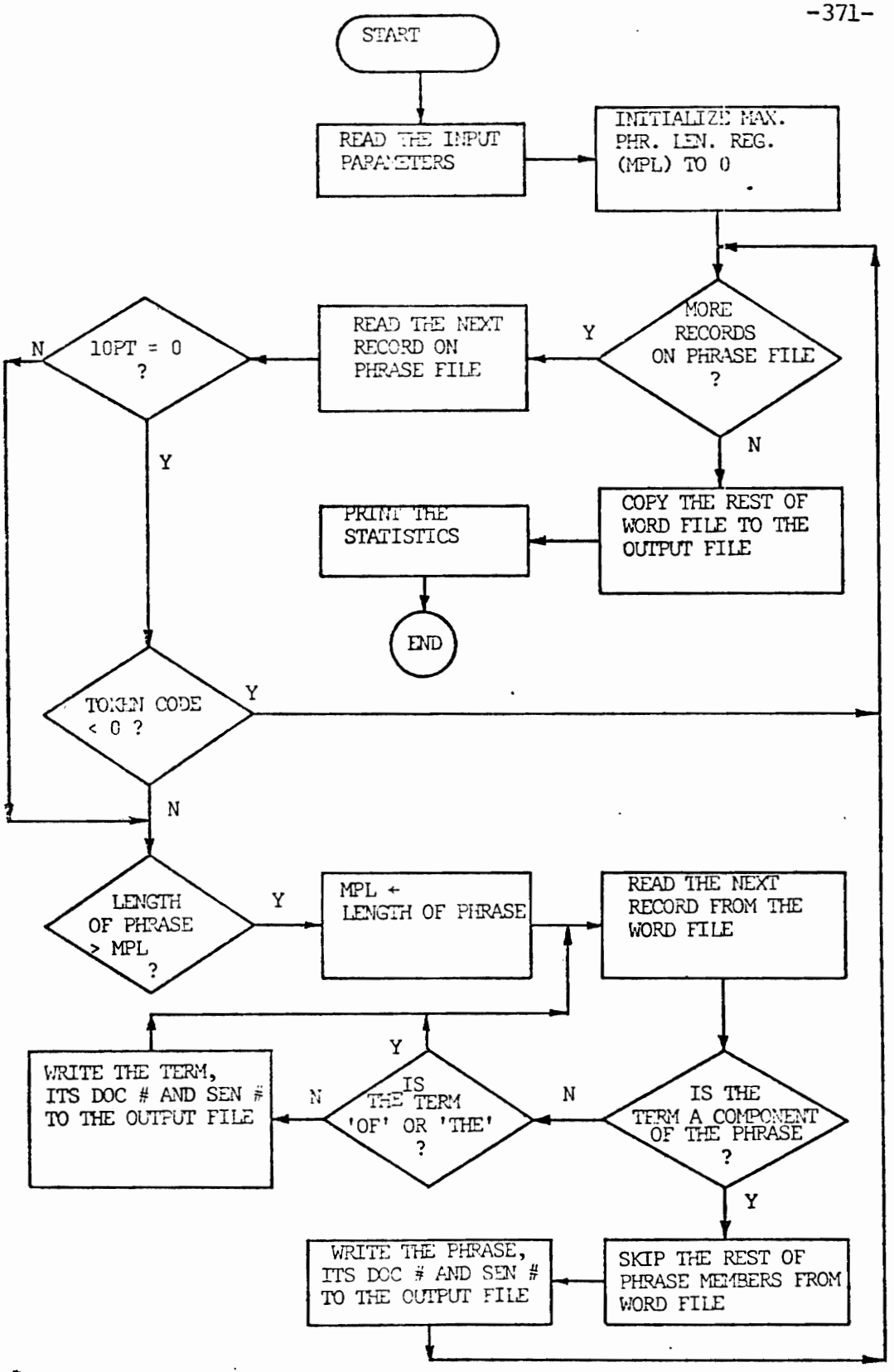


Figure C.10 General Flowchart of Program MRWRPH

C.3 Program ELDID.

Purpose of Program: Consolidation of multiple occurrences of terms within each document.

Program Description:

The program starts by reading a record from the input file. If the record read has a "delete mark", it is ignored and the next input record is read. If the current term matches the previous term in the document, a frequency counter is incremented and its correspoinding sentence identification numbers are saved in a stack, and then the next input record is read. If the current term does not match the previous term in the document; the previous term, its document identification number, its in-document frequency, and sentence identification numbers in the stack are written to the output file.

The program also accumulates the number of terms in each document and compares it against the user-specified maximum and/or minimum. If this accumulated total is greater than the user-indicated maximum, then the document identification number along with the actual number of keys in the document are printed out. If this total is less than the user-indicated minimum, then this information is saved in a temporary file rather than being printed out immediately. However, the information in the temporary file is printed

out after all input records are processed.

The flowchart of program ELDID is given in Figure C.11.

C.8 Program DELMER.

Purpose of Program: To delete and/or merge document representatives.

Program Description:

The program begins by reading the identification numbers of those documents which are to be deleted or merged, and putting them in a stack after sorting these numbers in the order of their magnitudes. It then starts reading the input file. Each input record read is saved as long as the document identification number contained in it (i.e. the second field) matches the one of the previous record. Whenever a record with a dismatching document number is read, the previous document number is checked against the top cell of the stack. If it is not found to be in the stack, then the saved records are written to the output file; otherwise they are either ignored, if deletion of the document has been specified, or written to a temporary file, if merging of the document has been specified. It also removes the matched document number from the stack by popping the stack off by one cell.

The emptiness of stack is the signal to the program for the completion of deletion or merging operation. As soon as

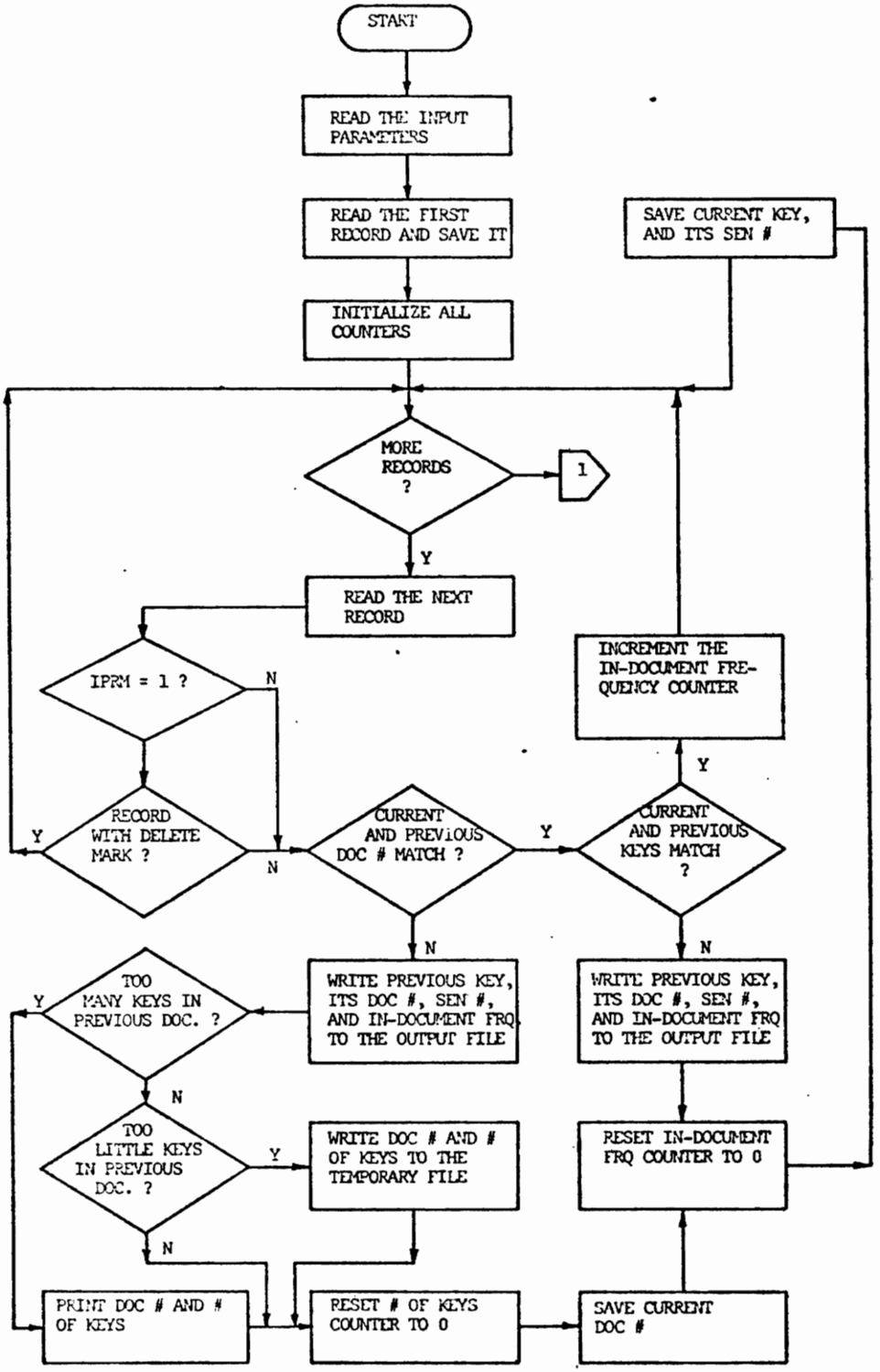


Figure C.11 General Flowchart of Program ELDID

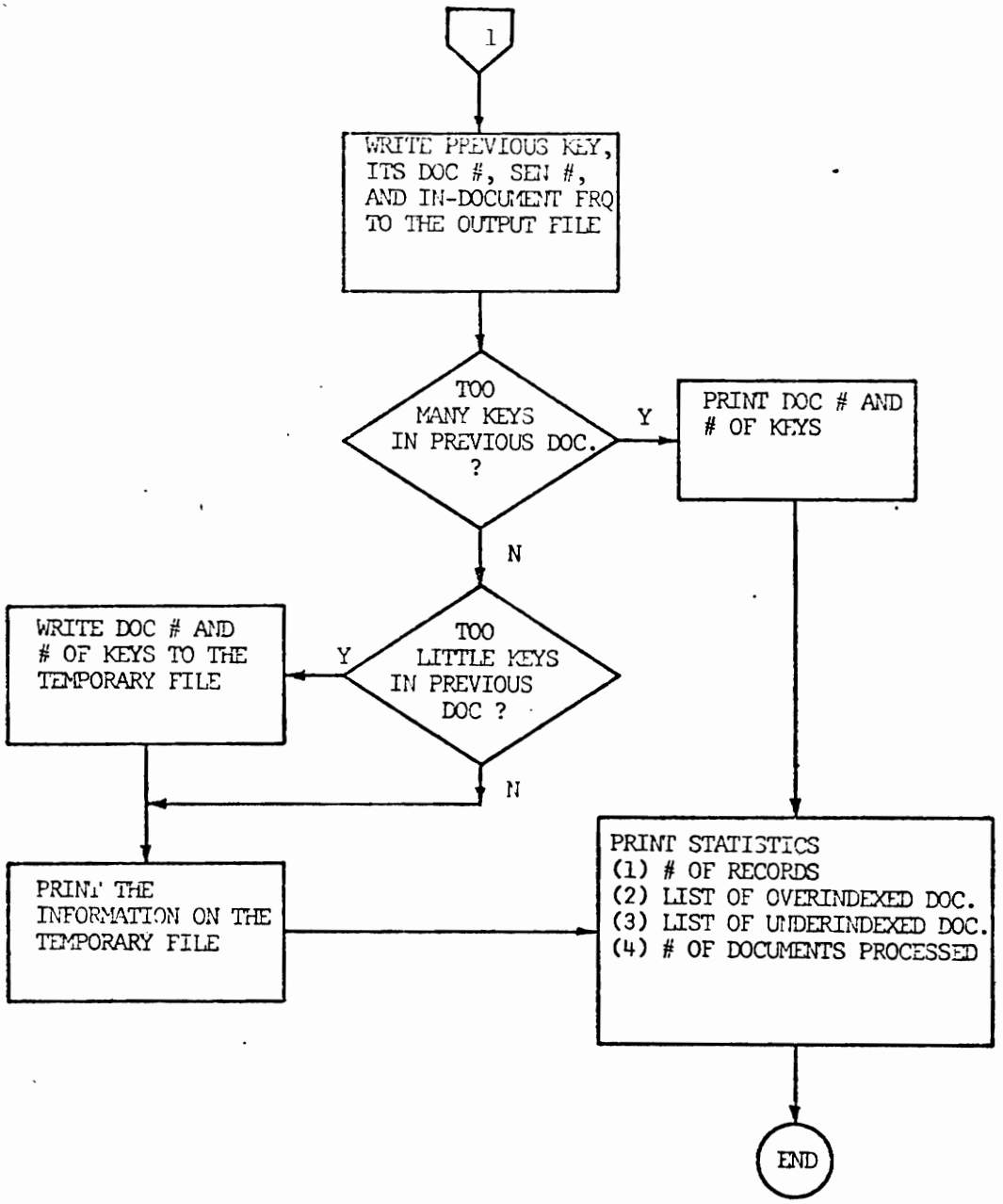


Figure C.11
-CONTINUED-

the stack becomes empty, the remaining input records are copied to the output file, and information in the temporary file, if any, is also copied to the end of output file.

The flowchart of program DELMER is given in Figure C.12.

C.10 Program ADJCMP.

Purpose of Program: To determine and indicate similarly spelled terms.

Program Description:

The "similarity" of terms is defined by two parameters: the similarity parameter and dissimilarity parameter. The first of these represents the minimum number of characters that two adjacent terms must have in common on a left-to-right scan. The stipulation by this parameter is a necessary but not a sufficient condition for two terms being similar. The second necessary condition is given by the dissimilarity parameter which represents the maximum length of disagreement. That is, the maximum number of characters between the first pair of corresponding characters that do not match and the end of the longer of the two adjacent terms being compared.

The program compares corresponding characters of adjacent terms on the input file from left to right. If the first necessary condition is not satisfied, the program rejects them from similarity candidacy; otherwise it further

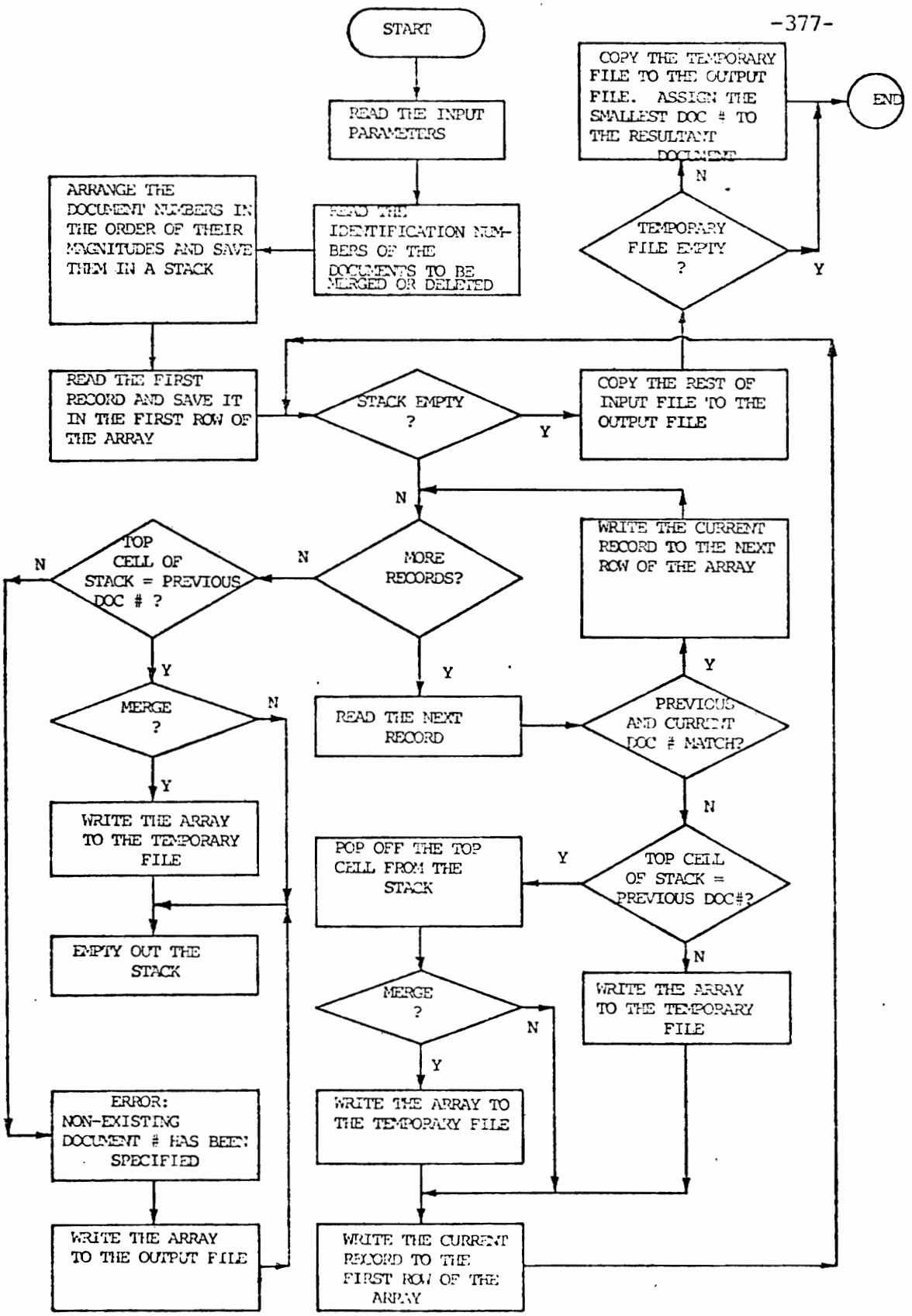


Figure C.12 General Flowchart of Program DELMER

checks if the second necessary condition is satisfied. If the second condition is also satisfied, then the terms are printed out in the same group of terms, i.e. the group containing similar terms. If the second condition is not satisfied, the program concludes that they cannot be similar, and starts comparing the next term with the current one for similarity candidacy.

Figure C.13 contains the flowchart of program ADJCMP.

C.11 Program COMDIR.

Purpose of Program: To compare the old and new vocabularies of index terms.

Program Description:

The program starts by reading on old term type (i.e. a term type from the old file 12) and a new term type (i.e. a term type from the new file 12) and comparing them lexicographically. If all the corresponding characters of these two terms match, it writes the term, its type code and corresponding document numbers (from the new file) to the temporary common terms file, and continues comparison after reading the next term types from the files. If these term types do not match, it takes appropriate action according to their positions in the alphabatized collating sequence. If the old term type is lexicographically smaller than the new term type, the program writes the old term type, its type

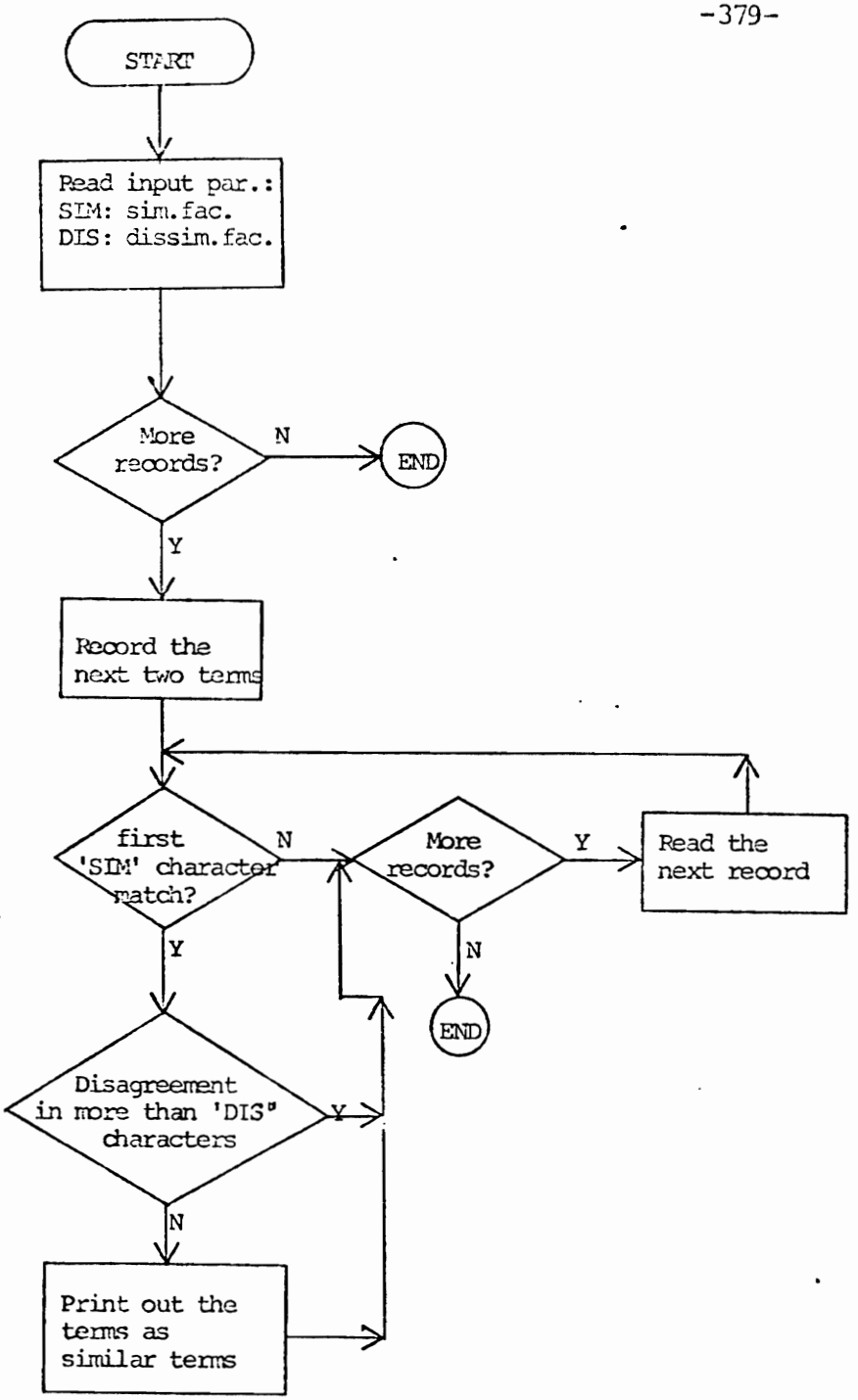


Figure C.13

General Flowchart of Program ADJCMP

code and corresponding document numbers (from the old file) to the temporary outgoing terms file, and restarts comparison after reading the next old term type. If the old term type is lexicographically greater than the new term type, then it writes the new term type, its type code and corresponding document numbers (from the new file) to the temporary incoming terms file, and continues comparison after reading the next new term type.

If the new file is exhuasted while the old file still contains more terms, the required information in the old file is copied to the temporary outgoing terms file. If the old file is exhuasted first, then the required information in the new file is copied to the temporary incoming terms file.

After both files are completely processed, the program prints out the three temporary files one after the other.

The flowchart of program COMDIR is given in Figure C.14.

C.12 Program DOCSUR

Purpose of Program: To generate document or key surrogates.

Program Description:

The operation of the program is controlled by the option parameter option =IOPT. If IOPT=0, file V is processed and key surrogates are generated. If IOPT=1, file 20 is processed and document surrogates are generated. The

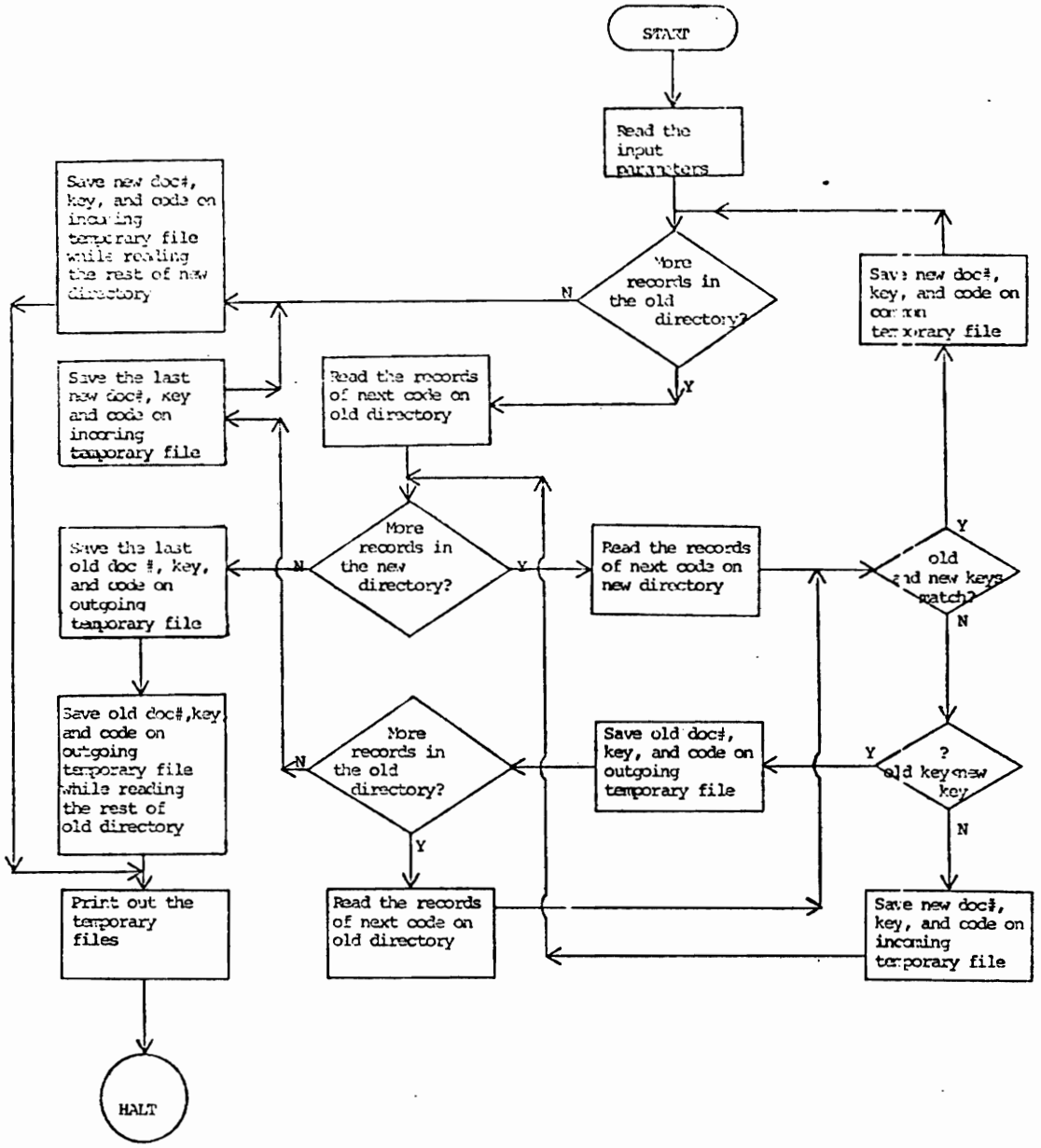


Figure C.14

General Flowchart of Program COMDIR

program's operation will be described here only for the latter case since similar description holds also for the first case.

The program starts by reading the input file and accumulating the codes that correspond to the in-document index term types assigned to each document. When the document number in the current record does not match the previous document's number, then the codes accumulated up to that point are the surrogates for the index terms assigned to the previous document. The accumulated codes are written to file I along with the previous document's number. The current document's surrogates are then accumulated starting with the code from the current record.

Before a code is added to the document's surrogate record, the total number of codes already accumulated is compared to the maximum number of terms allowed per document. If the document already has the maximum allowable number of terms, then the current term and all the remaining terms in the document, are printed out but not included in the surrogate record for the document. In this manner a surrogate for each document is created. The number of surrogate records created is compared to the number of document representatives in the collection specified as an input parameter. If these values do not match, a warning message along with both values is printed out.

The flowchart of program DOCSUR is given in Figure C.15.

C.13 Program UPDATE:

Purpose of Program: To decompose phrases and to delete, change or add terms on an interactive basis.

Program Description:

The operation of program UPDATE is best described through a flowchart which depicts the user-system interaction. Figures C.16 through C.19 contain such flowcharts. However, there still remain some points that should be further discussed.

File 12 (13) may be considered as a communication media between the user and file 11(4) as well as a directory used by the program to access the respective term (phrase) tokens. As illustrated in Figure C.20, another directory represented by the box with broken lines, is also used for the realization of the interaction. The interaction through path (2) is achieved by means of token codes alone, while the interaction through path (1) is made possible by means of type codes, file 12 (13) and the ISAM Key Directory. Detailed flow diagrams illustrating the access methods used by the program for the realization of interaction were already given in Figures C.8 and C.9.

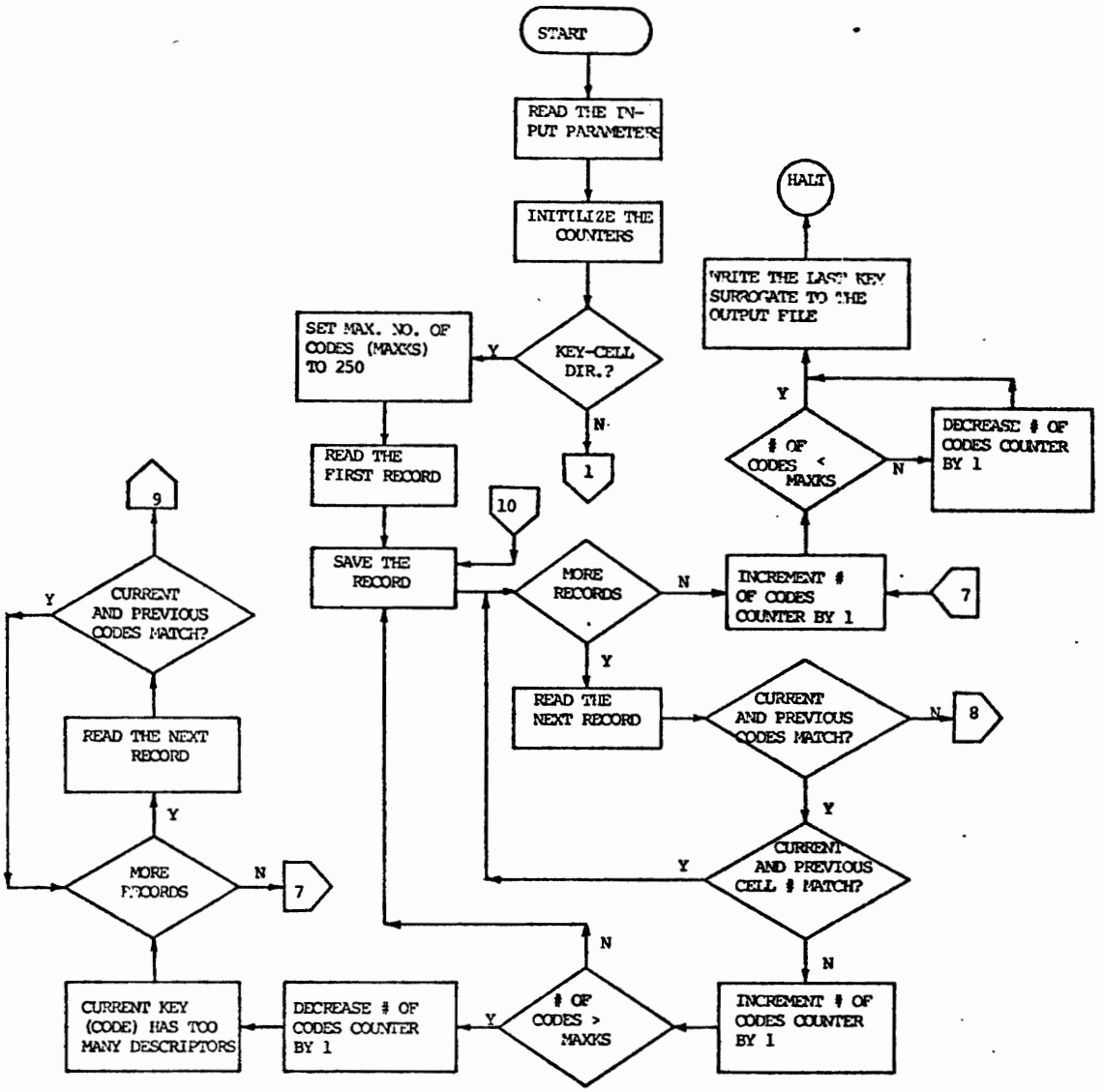


Figure C.15

General Flowchart of Program DOCSUR

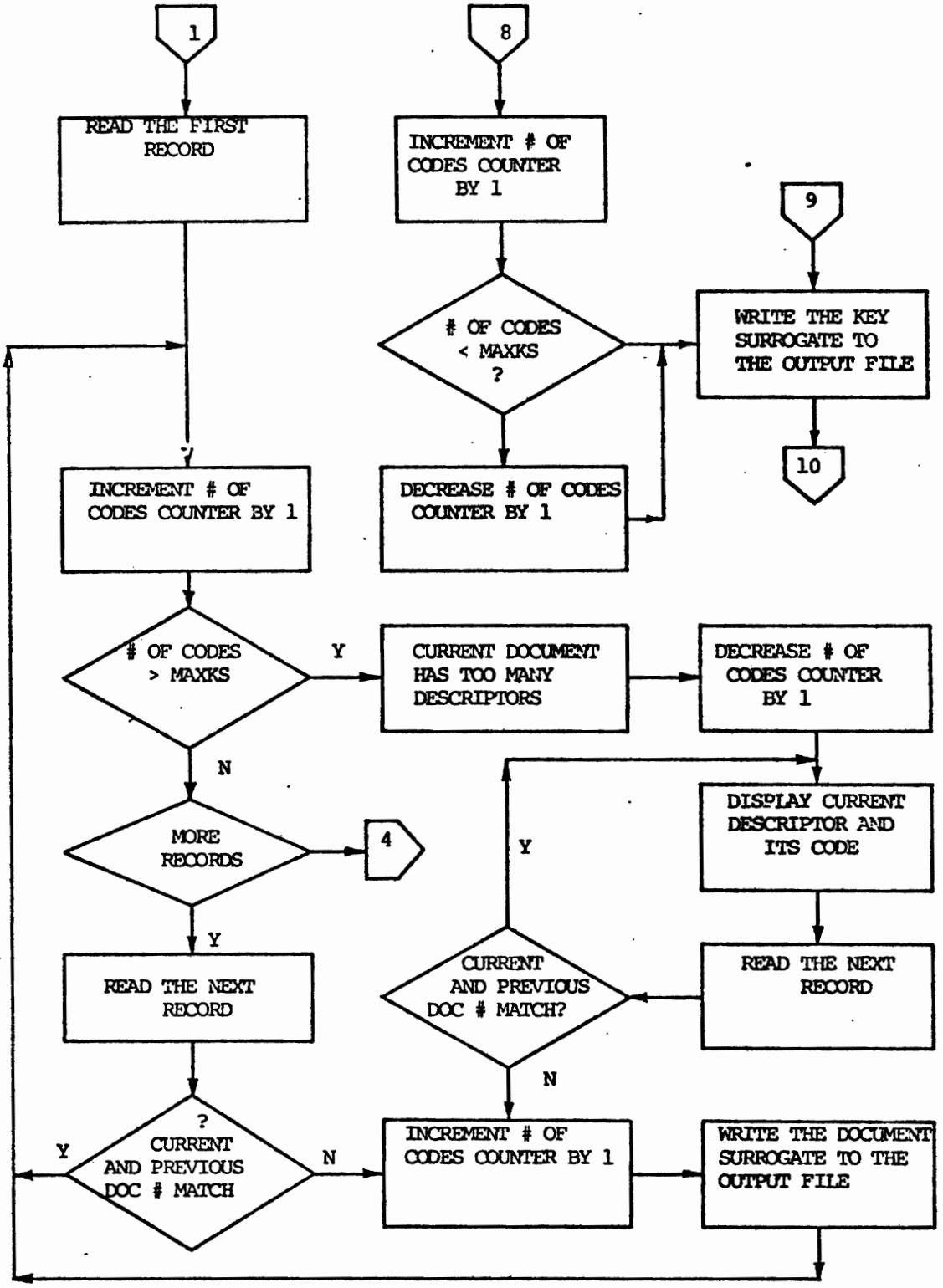


Figure C.15
-CONTINUED-

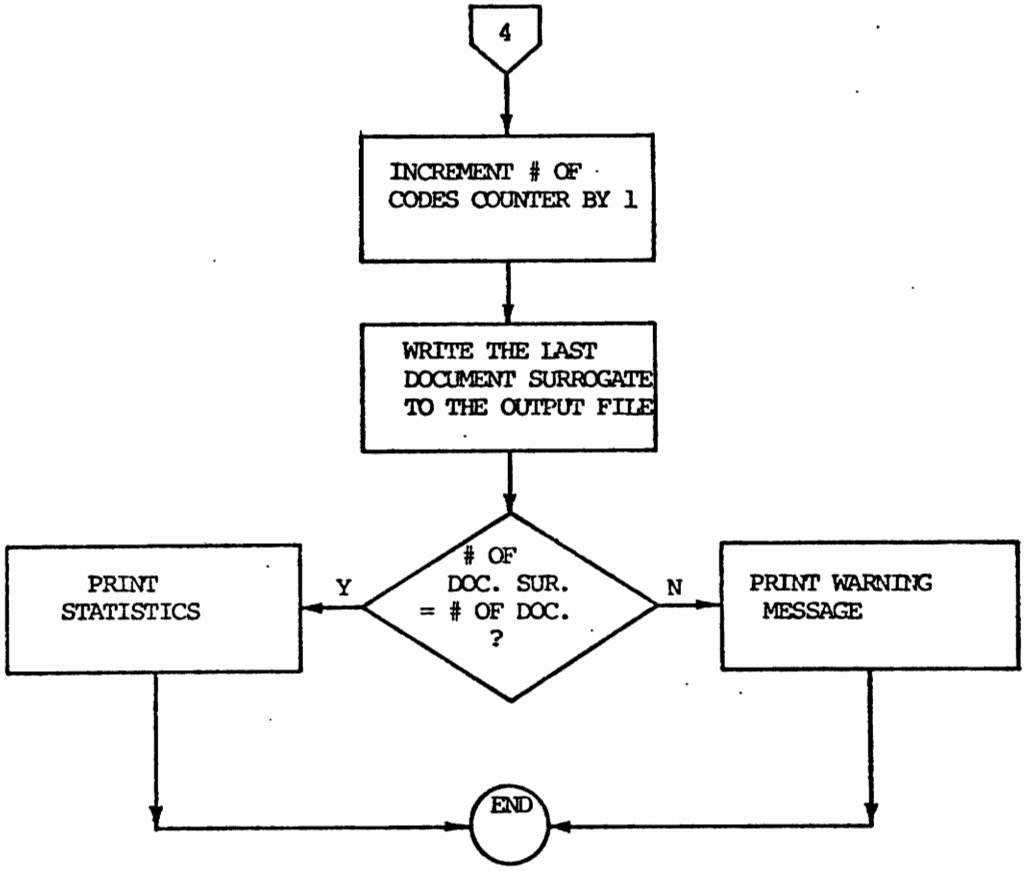


Figure C.15
-CONTINUED-

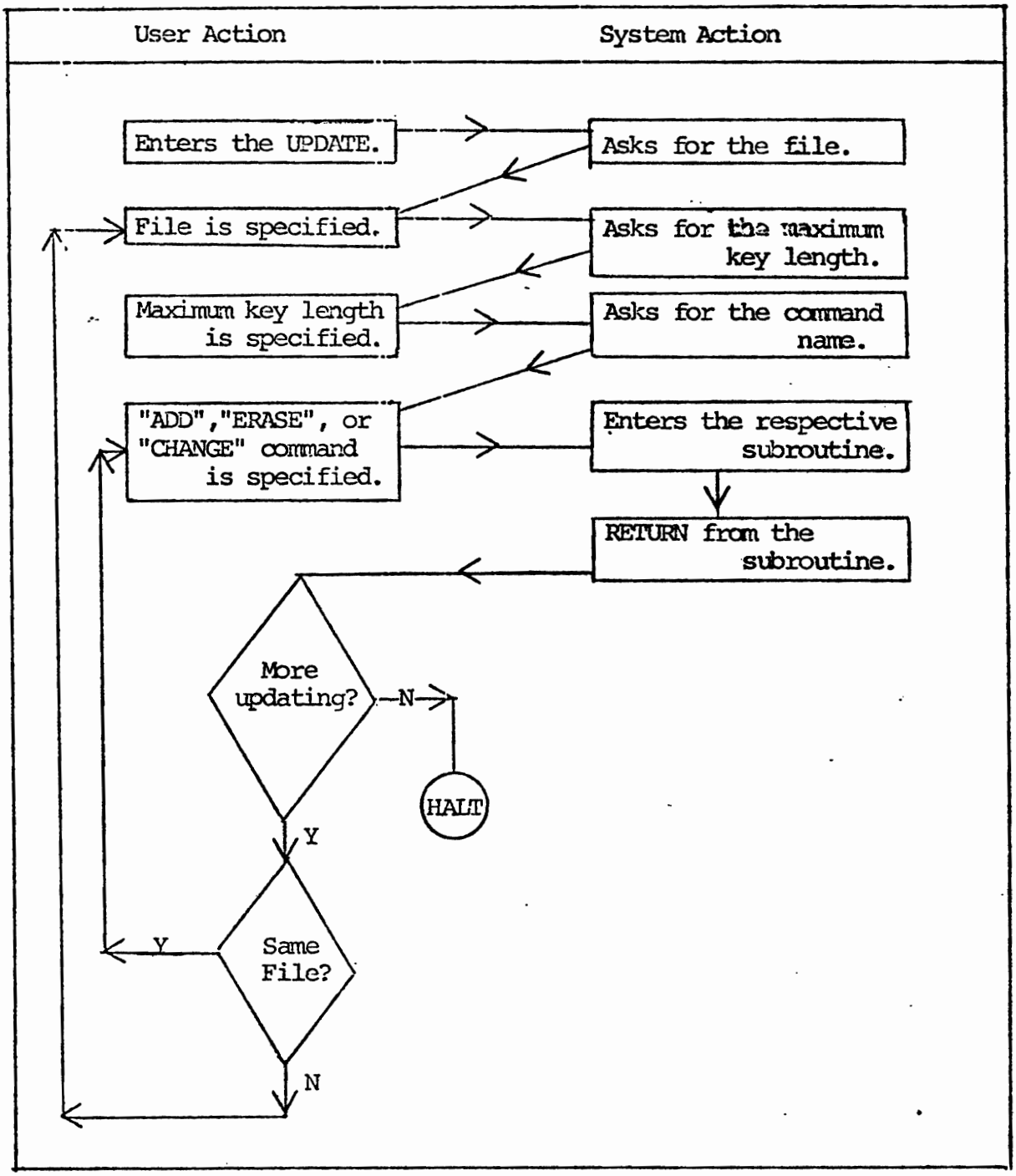


Figure C.16

Main Program UPDATE

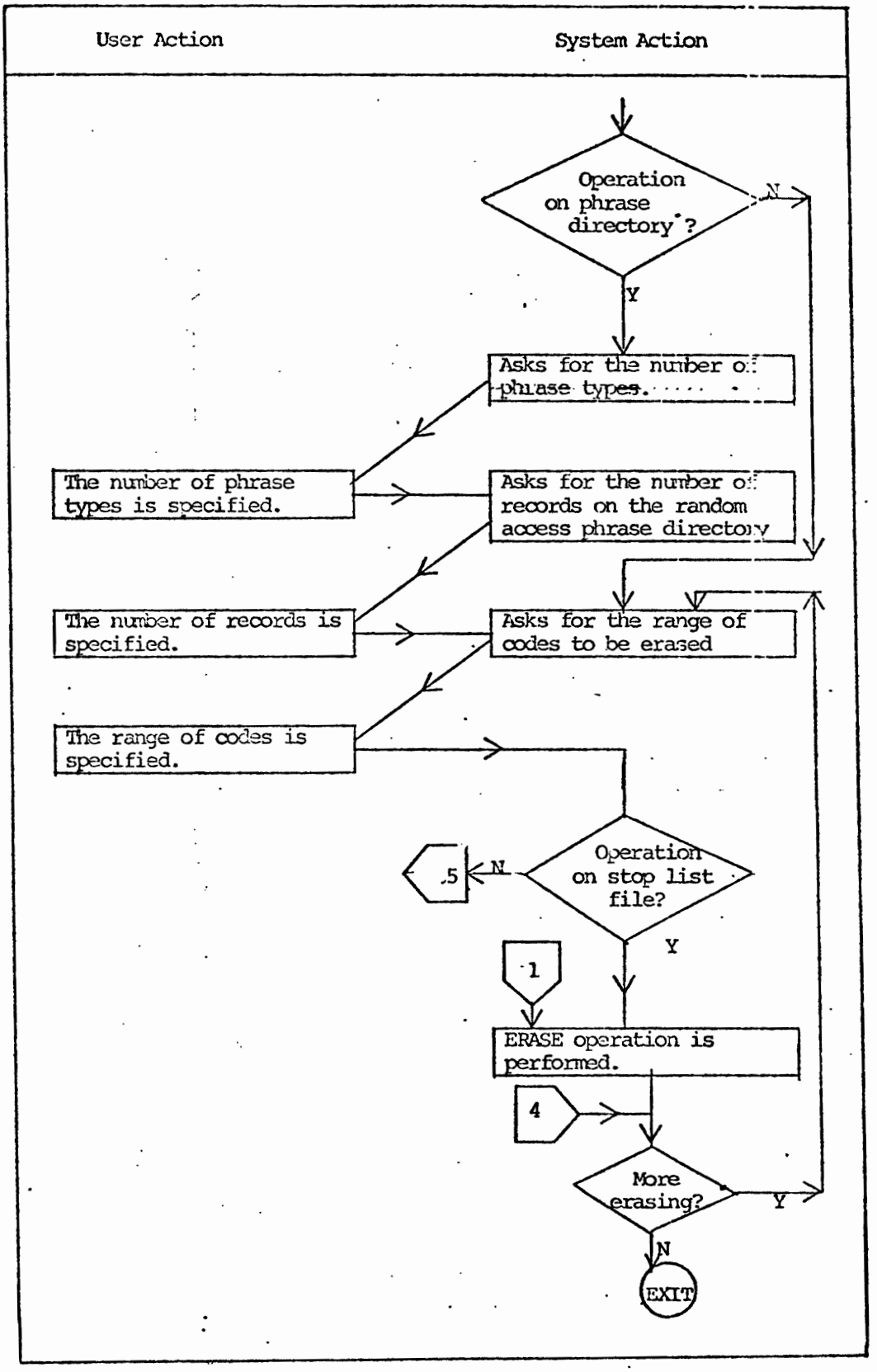


Figure C.17
Subroutine ERASE

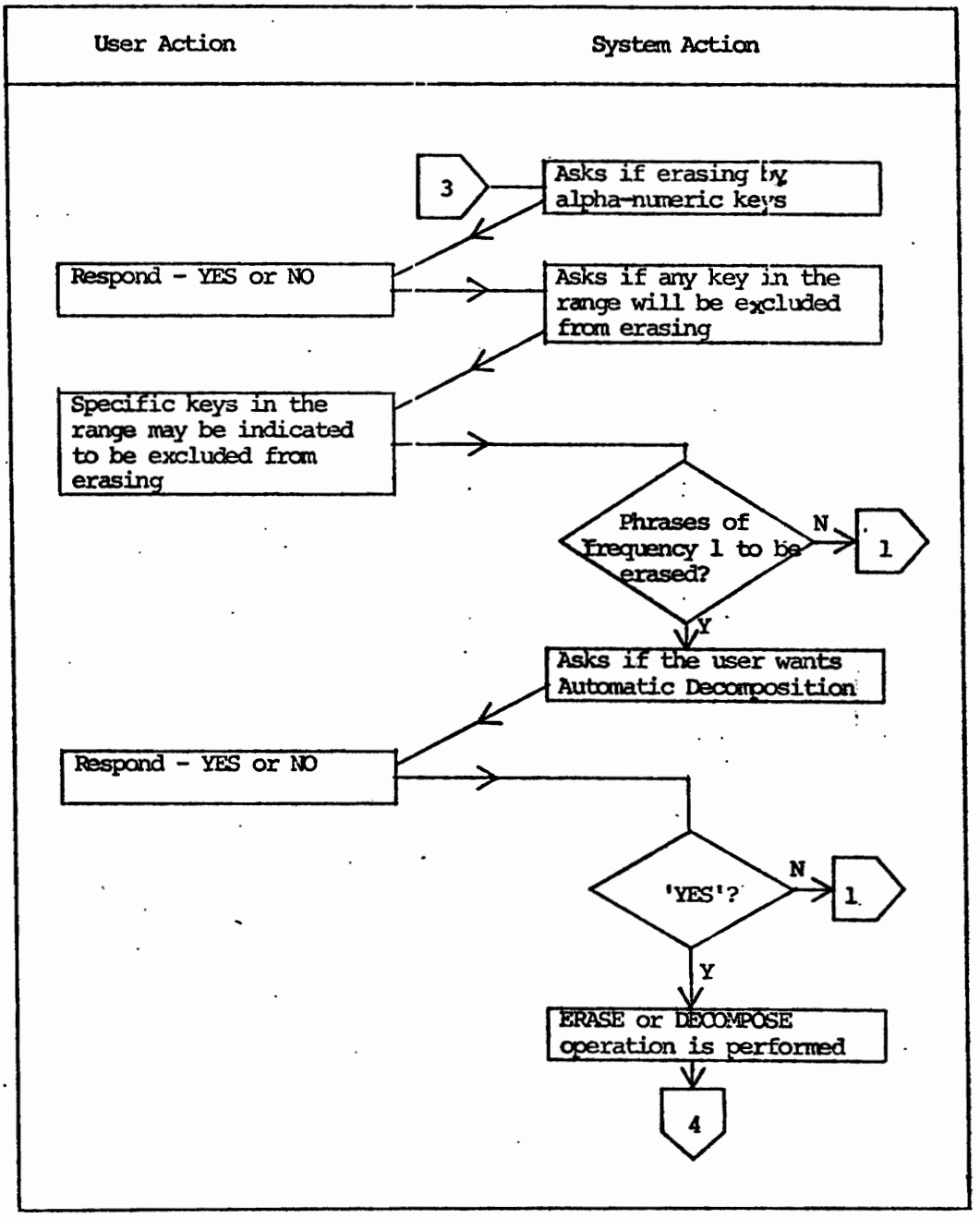


Figure C.17

-CONTINUED-

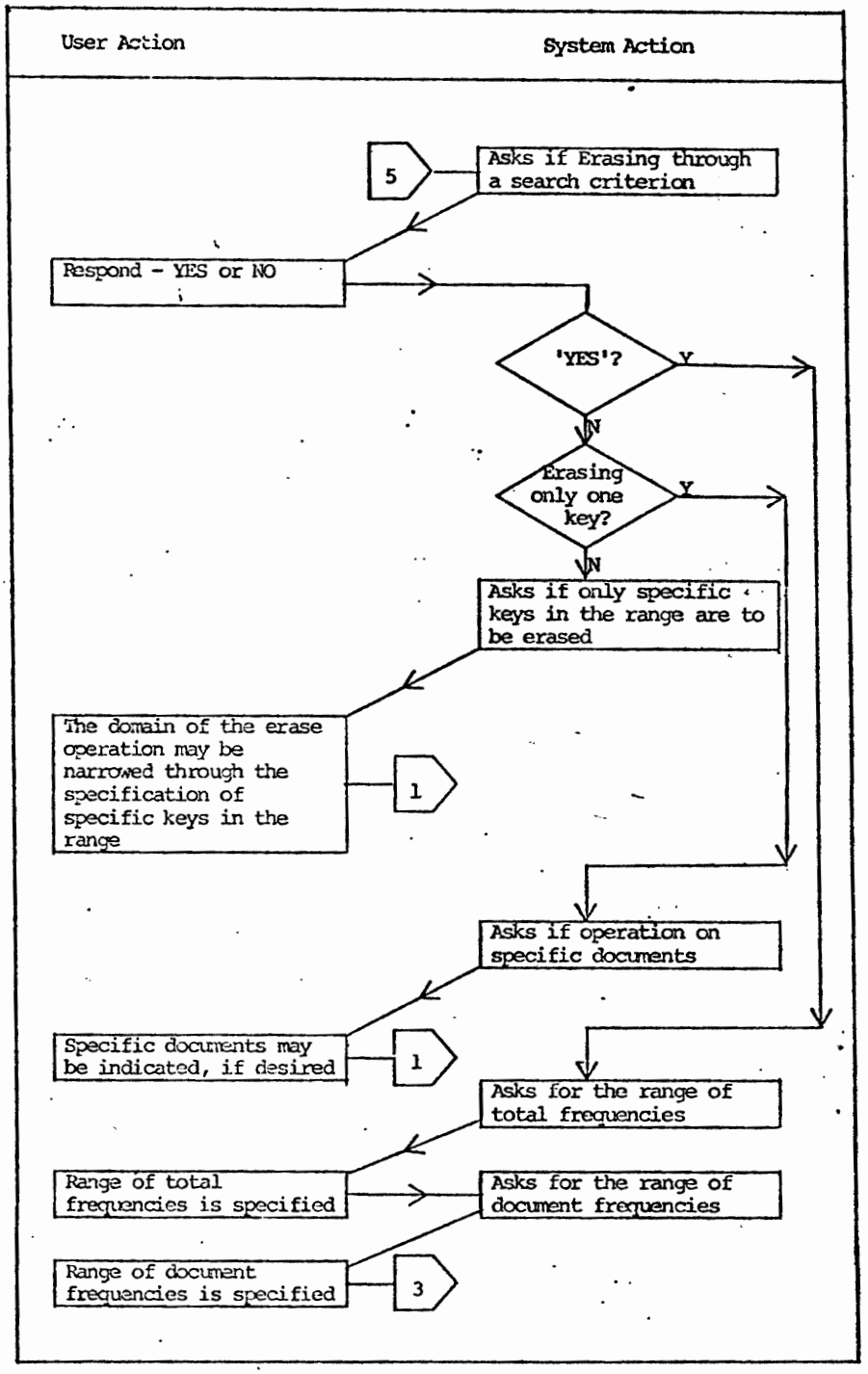


Figure C.17
-CONTINUED-

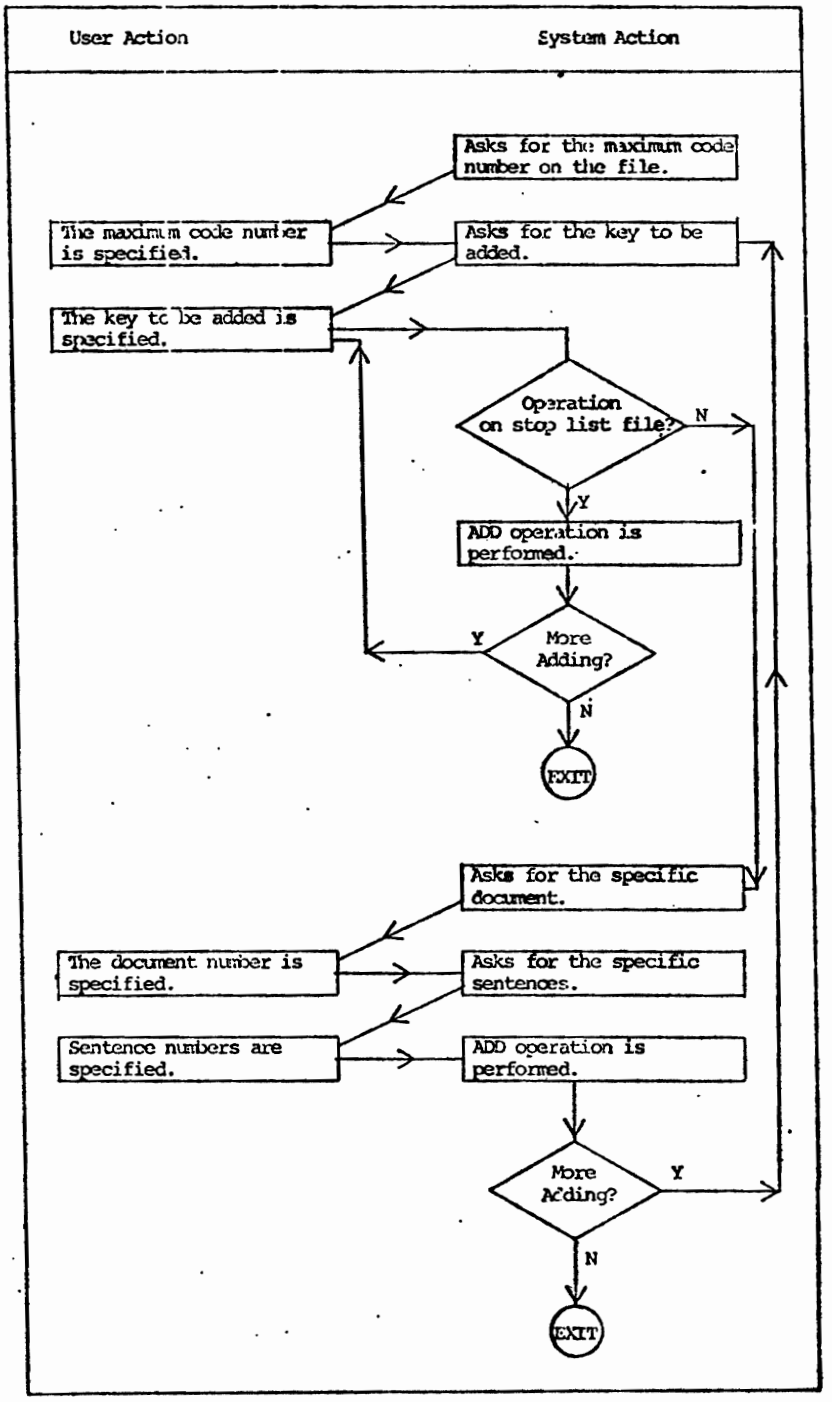


Figure C.18

Subroutine ADD

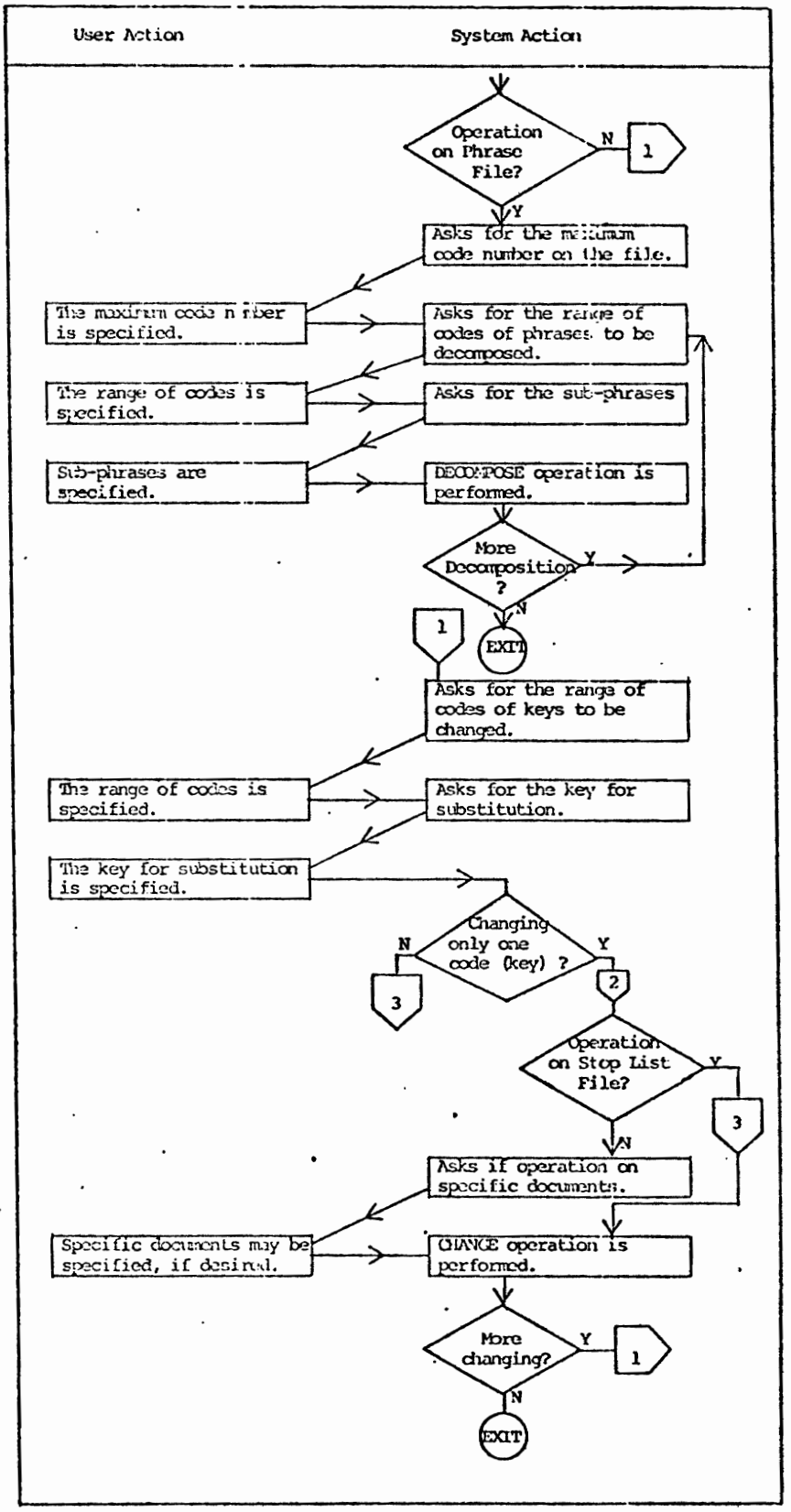


Figure C.19
Subroutine CHANGE

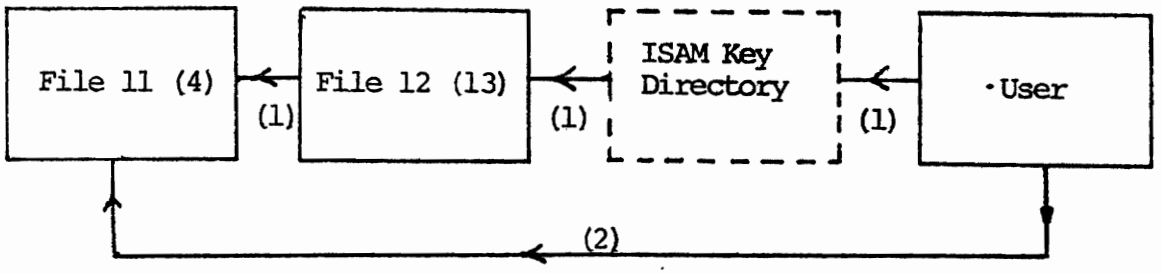


Figure C.20

Illustration of Referencing File 11 (4)

Deletion of a term is achieved by replacing its corresponding codes by a "delete mark" which is defined to be a negative number. If deletion is due to the failure of automatic decomposition of a phrase, then the code fields of the corresponding records are replaced by "-2", indicating that the phrase may come to existence if it is matched to a sub-phrase while some other phrase is being decomposed. In all other cases, the program uses "-1" as a delete mark that makes access to the correspoinding records impossible. The records of a deleted term cannot be accessed by the user, even though they physically exist in file.

If deletion by frequencies and/or alpha-numeric keys has been demanded, the program first checks if the term to be deleted satisfies the stipulations in the search criterion. This is accomplished simply by testing the required fields

of the corresponding code-record in file 12 (13). If the stipulations are satisfied, then the deletion process is performed; otherwise it is aborted.

Adding a new term to the file is accomplished by writing the term to the end of the file. The address of the first available record to be written is determined by the fact that the "maximum code number" on file 11 (4) happens to be the ISAM key to the last record of the file. The "maximum code number" counter is always incremented by one, whenever a new term is to be added.

Change operation consists of replacing the occurrences of the term (to be changed) in file 12 (13) and in file 11 (4) by the new term specified. However, if the domain of the change operation is file 4, then the phrase types in file 13 are not changed. By doing so, the program is always able to perform binary search against the list of phrase types in file 13, which becomes necessary in the process of automatic decomposition. It should be noted that change operation (decomposition) of file 4 forces the system to add sub-phrases to file 4, if more than one sub-phrase is specified.

Automatic decomposition of a phrase starts with parsing the phrase in left to right order, subjecting to pull out a sub-phrase which already exists in file as a phrase or a deleted-phrase with delete mark "-2" i.e. deleted due to

the failure of automatic decomposition. Therefore, for each extracted sub-phrase, a binary search is done to determine if there is a "code-record" in file 13, which contains the sub-phrase as a phrase type. In the process of binary searching, the program does not compare the sub-phrase against deleted phrases with delete mark "-1". If such a phrase can be found, it is in turn added to file 11. If the matched phrase in file 13 is already a deleted-phrase with delete mark "-2", then the code fields (which contain the delete mark) are replaced by the actual codes. If the matched phrase is not a deleted-phrase, its total frequency is tested, before further decomposition. If this total frequency is equal to 1, then the matched phrase is flagged off by the special mark "0". The marking of a phrase in this way indicates to the system that the phrase is not of total frequency 1 any more. If the total frequency is different than 1, no marking is performed. Phrases marked in this way are not subjected to automatic decomposition, since they are not of frequency 1 any longer.

After a sub-phrase is tested and added to the file, if matched, automatic decomposition restarts where it left off. At the completion of the parsing the main phrase is deleted from both files, if decomposition has not been successful. However if decomposition were successful, it is replaced in file 11, while it is deleted from file 13. In the deletion

operation, the delete mark "-1" is used, if the decomposition has been successful, and "-2" is used otherwise.

Figure C.21 contains the flowchart describing the process of automatic decomposition.

In addition to the automatic decomposition process, the program is also able to decompose phrases nonautomatically. Nonautomatic decomposition of a phrase, with any total frequency, is achieved by pulling out the sub-phrases that are specified by the user and substituting the first sub-phrase for the main phrase, and adding the rest of sub-phrases, if any, to file 4.

C.14 Program FRQWCN

Purpose of Program: To generate a summary report.

Program Description:

If zero is specified for the input parameter <order>, the program reads the input records and accumulates the number of records read as long as the total frequencies of the previous and current keys match. It also accumulates the number of records read as long as the document frequencies of the previous and current keys match. Whenever document frequencies do not match, it stores the previous document frequency (breakdown document frequency) and the number of keys with this document frequency in an

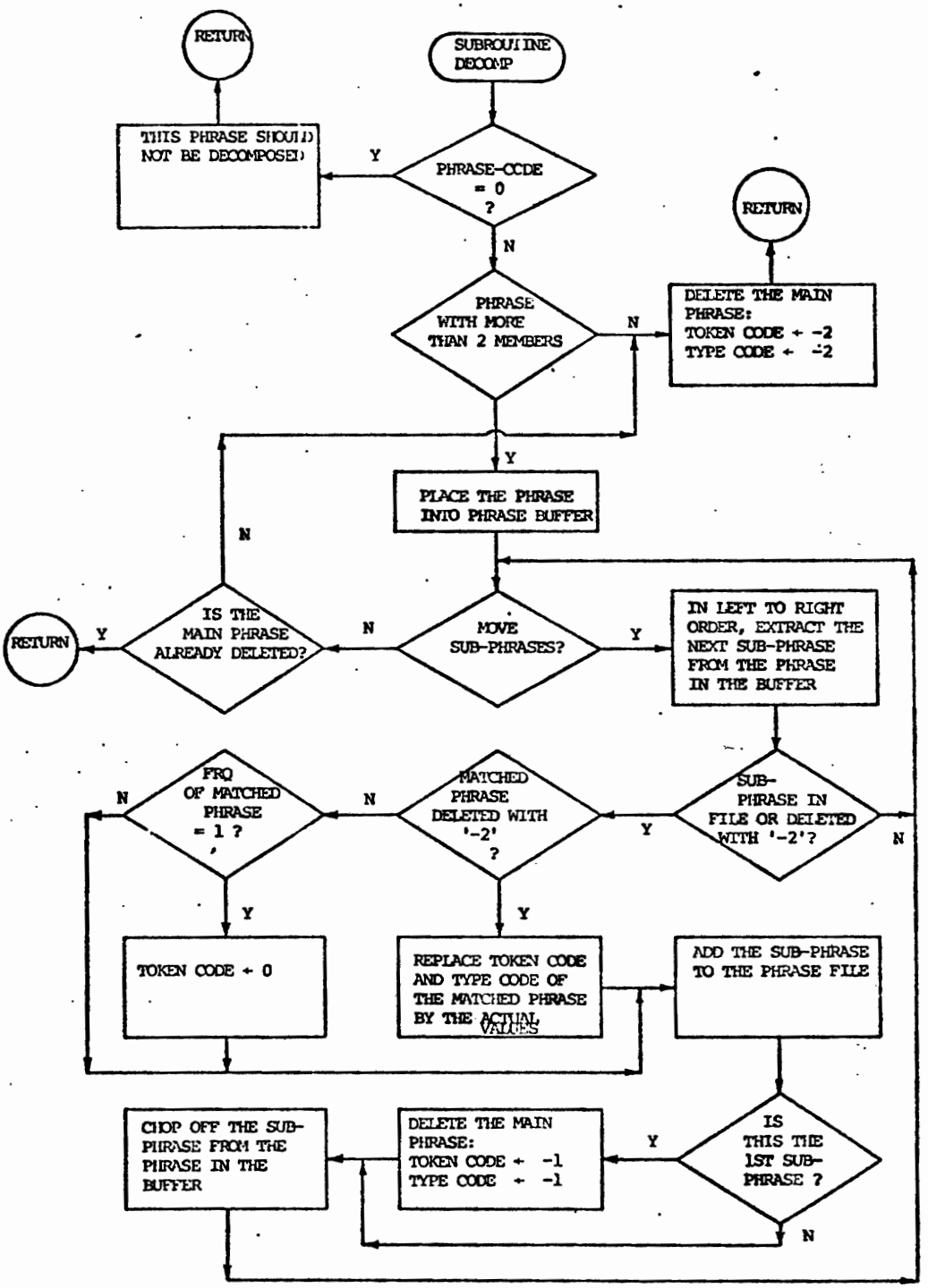


Figure C.21

General Flowchart of Subroutine DECOMP

array. As soon as the total frequencies do not match, the program saves on a temporary file the previous total frequency along with the number of keys with this total frequency and the information stored in the array. When the processing of the input file is finished, the program starts reading back the temporary file and calculating the statistics formulated in section 3.2.4.5. The statistics are in turn printed out in accordance with the formats shown in figure 3.14. Similar description holds for the operation of the program for <order>=1.

Figure C.22 contains the flowchart of program FRQWCN.

C.15 Program PRTON

Purpose of Program: On-line display of information.

Program Description.

The operation of program PRTON is best described through a flowchart which depicts the user-system interaction. Figure C.23 contains such a flowchart for the program.

Except for file 12 (13), file 11 (4) and Standard Formatted Text File, the display of a record in any file is performed through the specification of its address relative to the beginning of the file. In this context, the relative address of the Nth record in file is defined to be N, itself.

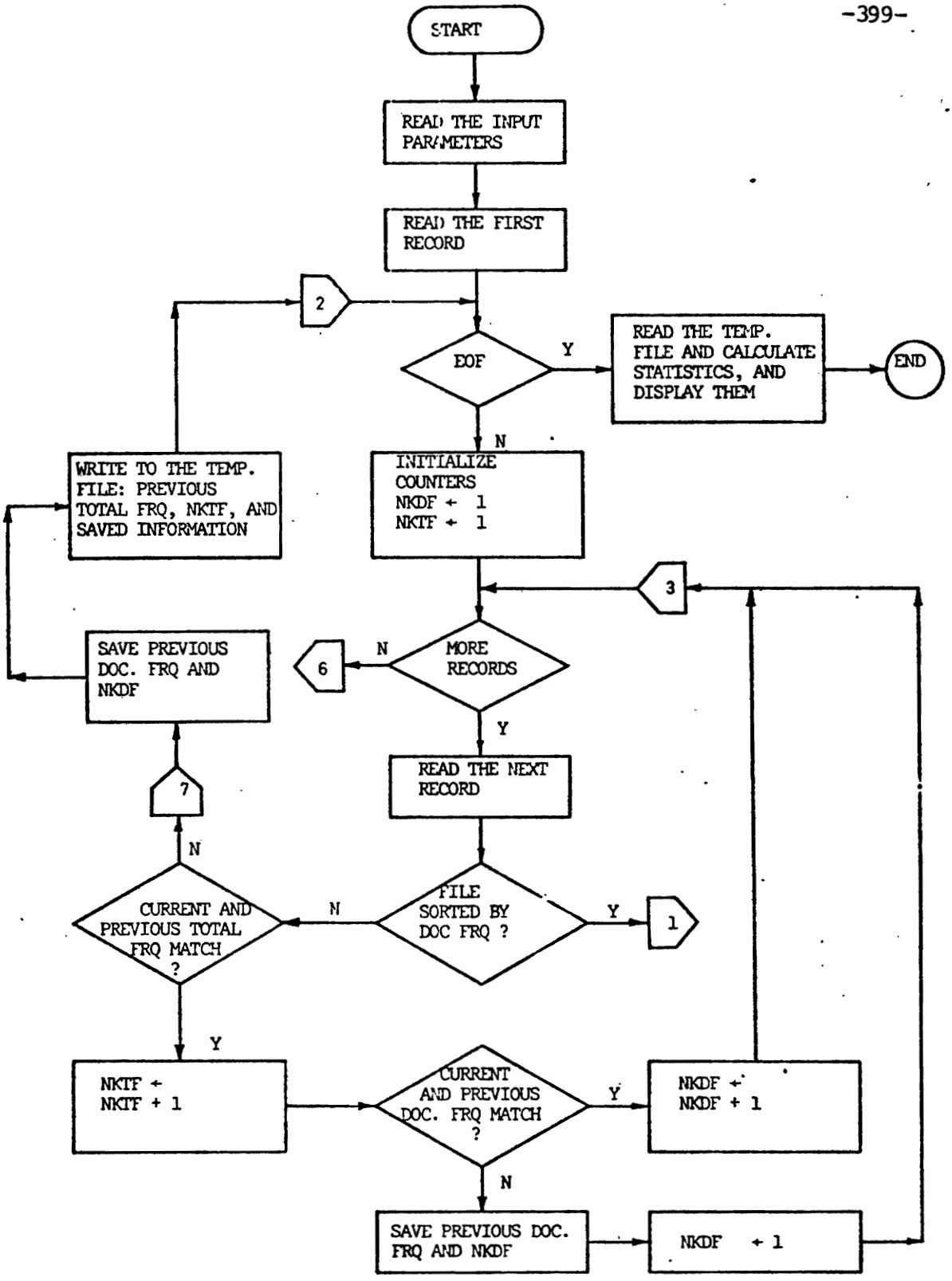


Figure C.22
General Flowchart of Program FRQNCN

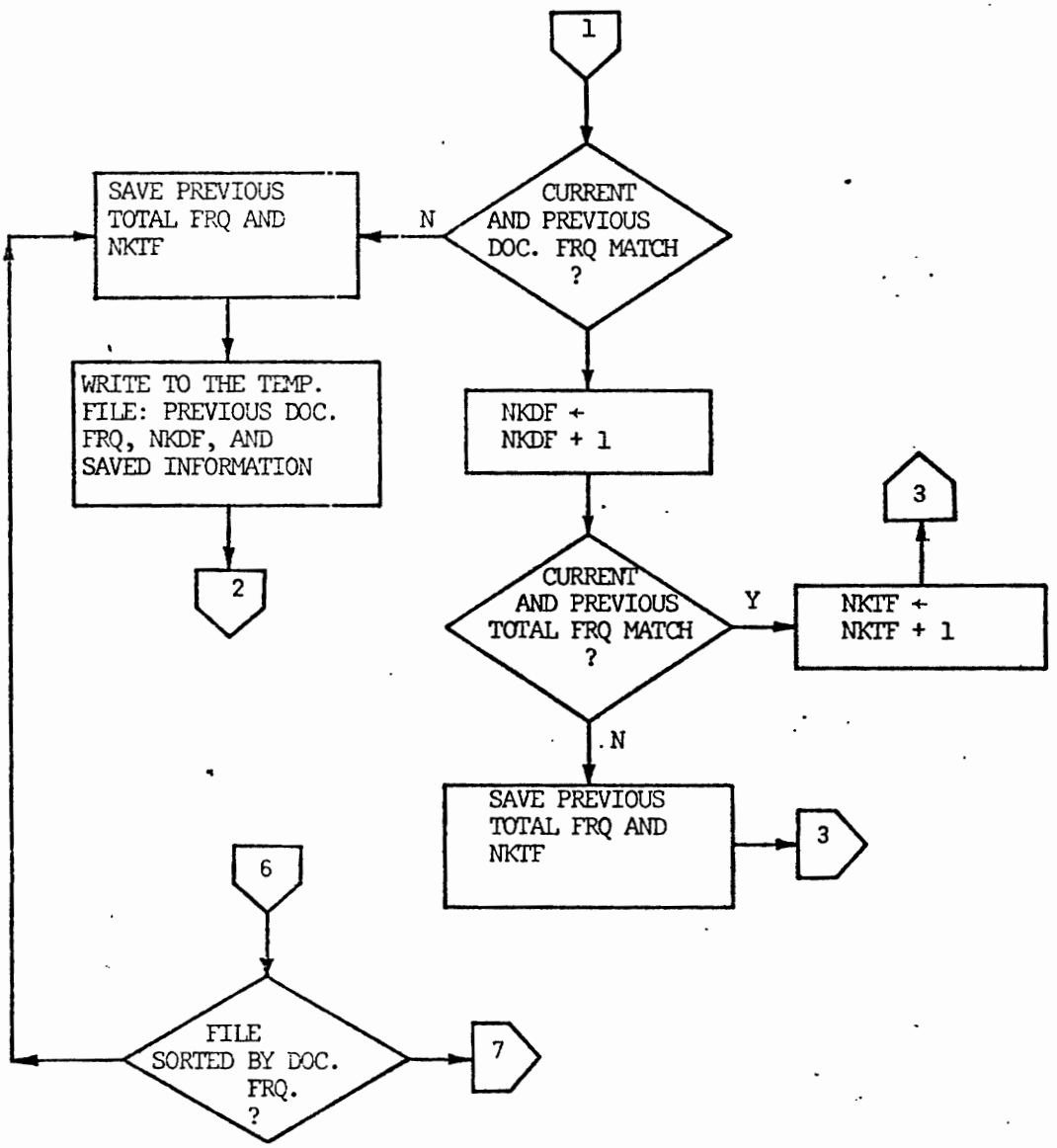


Figure C.22
-CONTINUED-

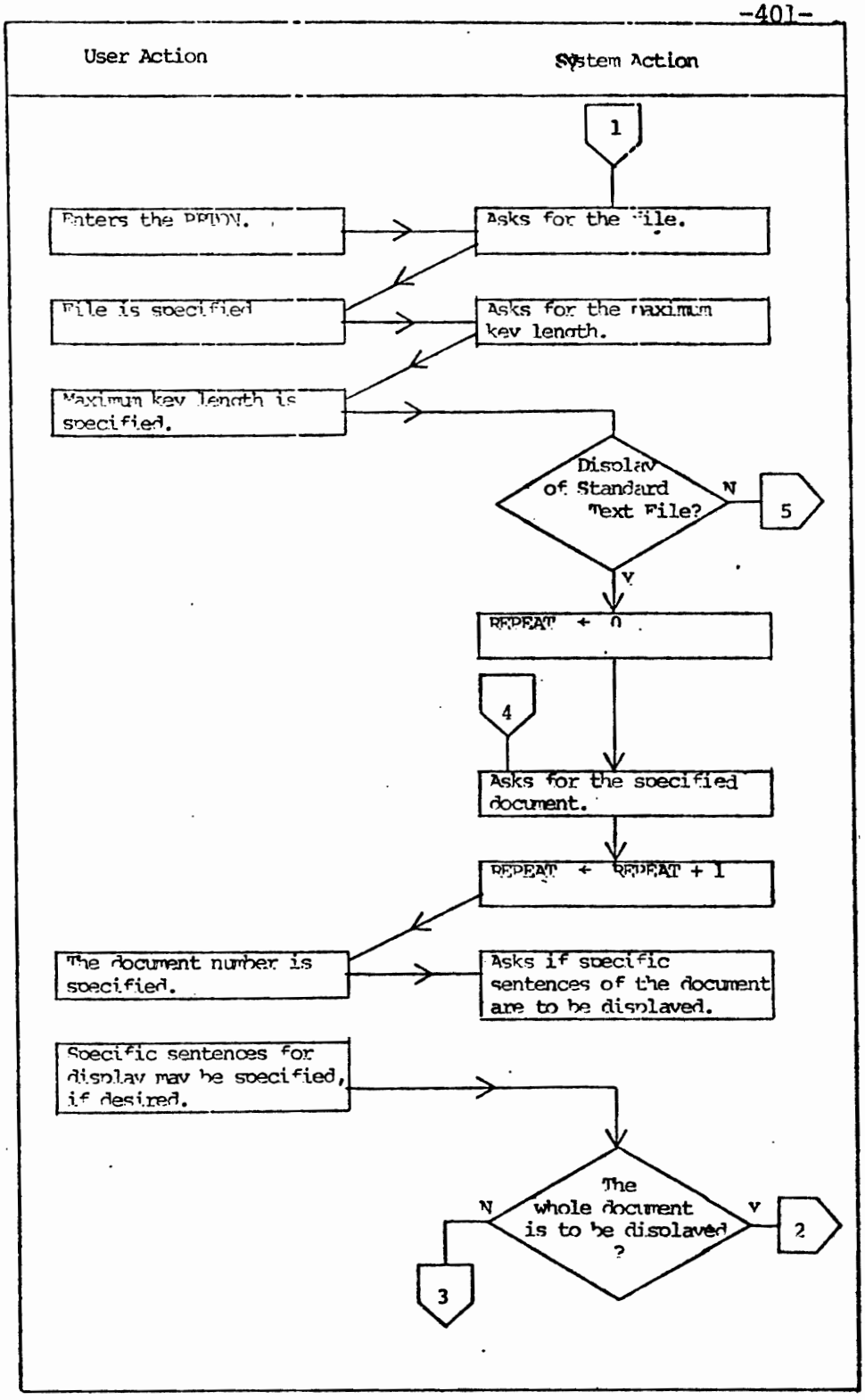


Figure C.23

Program PRTON

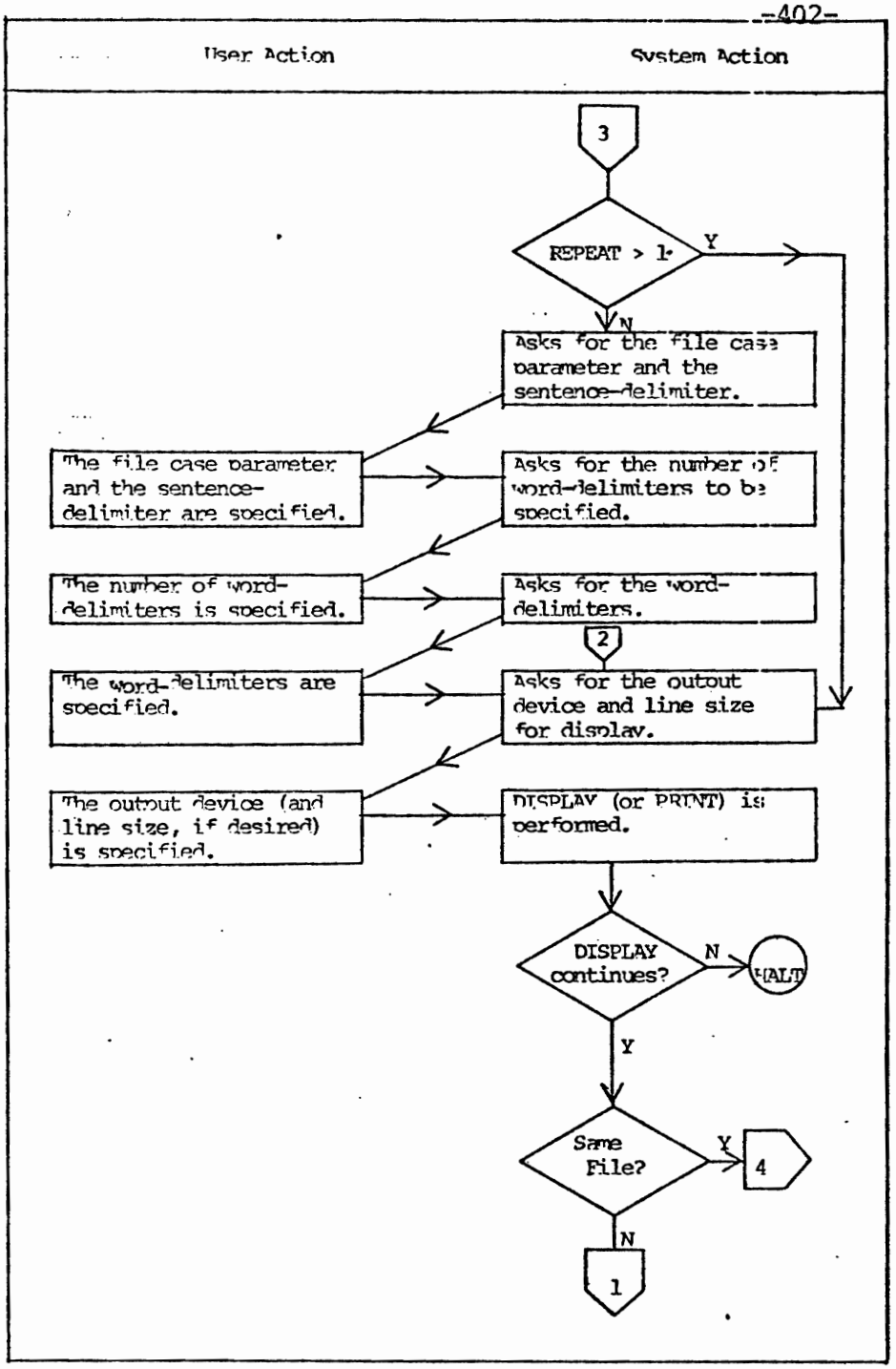


Figure C.23
-CONTINUED-

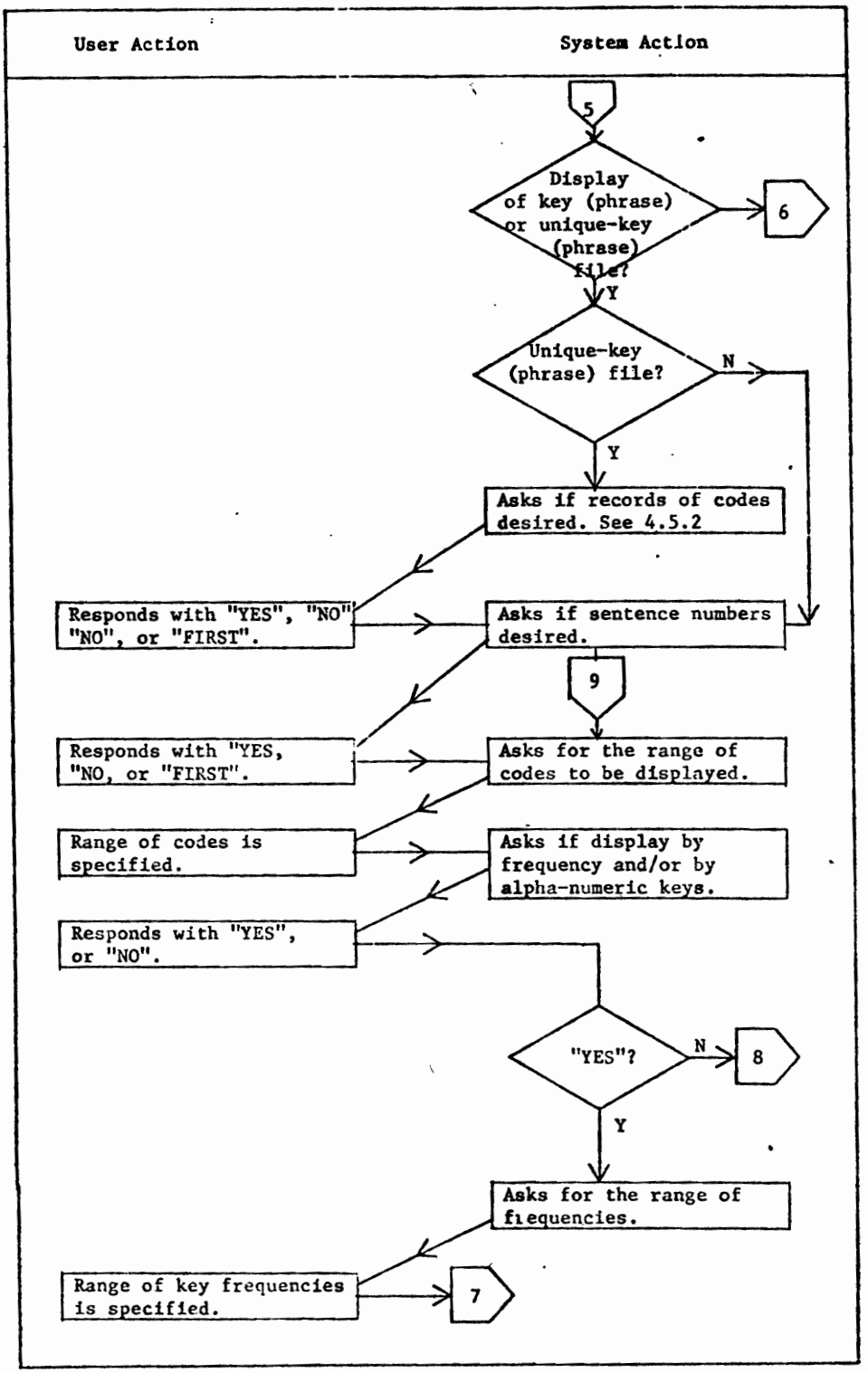


Figure C.23
-CONTINUED-

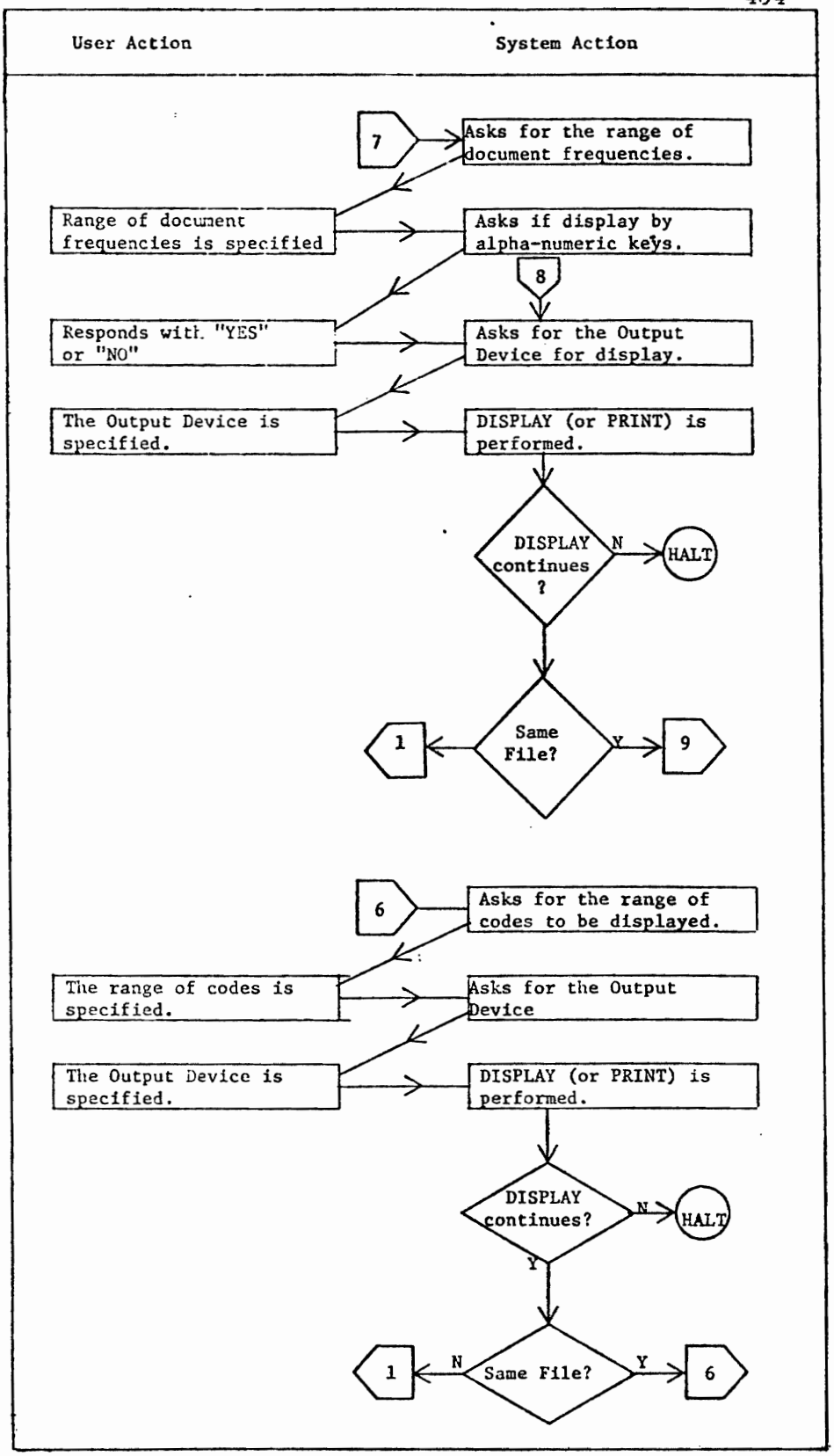


Figure C.23
-CONTINUED-

The display of records in file 12 (13) is achieved through the specification of respective type codes. Similarly, the display of file 11 (4) is achieved by the specification of respective type or token codes. Figure C.8 depicts the interaction for the display of these files. As the figure indicates, interactive display of file 11 (4) can be achieved by means of type codes, file 12 (13), and ISAM Key Directory, or by token codes alone. In the display of these files, if display by frequencies and/or alpha-numeric keys has been demanded, that is, through a search criterion, the program first checks if the key to be displayed satisfies the stipulations put forward by the search criterion. This is accomplished simply by testing the required fields of the corresponding code-record in file 12 (13). If the stipulations are satisfied, then display operation is performed; otherwise it is aborted. Figure C.8 and C.9 describe the access relationship between the respective files.

The program always examines if the record to be displayed contains a delete mark. By no means, a record containing a delete mark is displayed although it physically exists in file.

Display of the Standard Formatted Text File is achieved through the specification of document identification numbers. Furthermore, specific sentences of a document can

also be displayed, if their identification numbers are specified. Figure C.2 illustrates the access method used to read the Standard Formatted Text records corresponding to a document. The program reads the text of a specified document into computer memory and keeps it there until display of another document is demanded. A modified version of the subroutine used by program INDEX to determine the sentence boundaries is also used here for the display of specific sentences of a document. The subroutine first sortes the specified sentence identification numbers in the order of their magnitudes and then starts parsing the text of the document for the respective sentences.

The program always gives the user the option to specify an output device before print or display operation takes place. Large amount of information can be sent to the line-printer, while small amount of information can be displayed on a terminal for the user's examination.

C.16 Program PRTOFF.

Purpose of Program: Off-line display of information.

Program Description:

The program reads the input file sequentially and produces reports of a specified number of records or document representatives or the whole file on a print-out through the line-printer.

C.17 Program CLASIFY.

Purpose of Program: To perform document or key classification.

Program Description:

The classification algorithm used here follows algorithms originally suggested by Dr. David Lefkovitz (Le69) and subsequently adopted by Dr. Litofsky(Li69). The algorithm and its implementation are described in section 3.3.1.2.

C.18 Program GNTRCD.

Purpose of Program: To assign the in-document key types of each document to its respective cell.

Program Description:

Each cell (i.e. terminal node) is assigned the in-document key types which result from the union of key types of the documents in that cell.

The program starts by reading a record from file 20 and performing a binary search on file III for the document number contained in the record. After the record is matched, the program retrieves the respective cell number. Subsequently, it writes the cell number and respective type code to file IV. The program continues reading file 20 and writing each type code read and the respective cell number

to file IV, until a record with a different document number is read. When a record with a different document number is read, the program replaced the third field in the corresponding record of file III by the number of in-document types codes in the document.

Figure C.24 contains the flowchart of program GNTRCD.

C.19 Program GNFKFL.

Purpose of Program: To generate complete set of keys within a cell.

Program Description:

First, the program eliminates multi-occurrences of codes in each terminal node, and also produces for each key type in the terminal node the document frequency of the key within the terminal node. This task is achieved by reading a code from the input file and comparing it with the previous one. If the current code matches the previous code in the terminal node, a frequency counter is incremented and another code is read. If no match occurs; the previous code, its terminal node number and its in-cell frequency are written to a temporary file. If the current terminal node number does not match the previous terminal node number; the same information is saved on the temporary file.

Second, the program reads back the temporary file sequentially in order to produce file VII. If the current

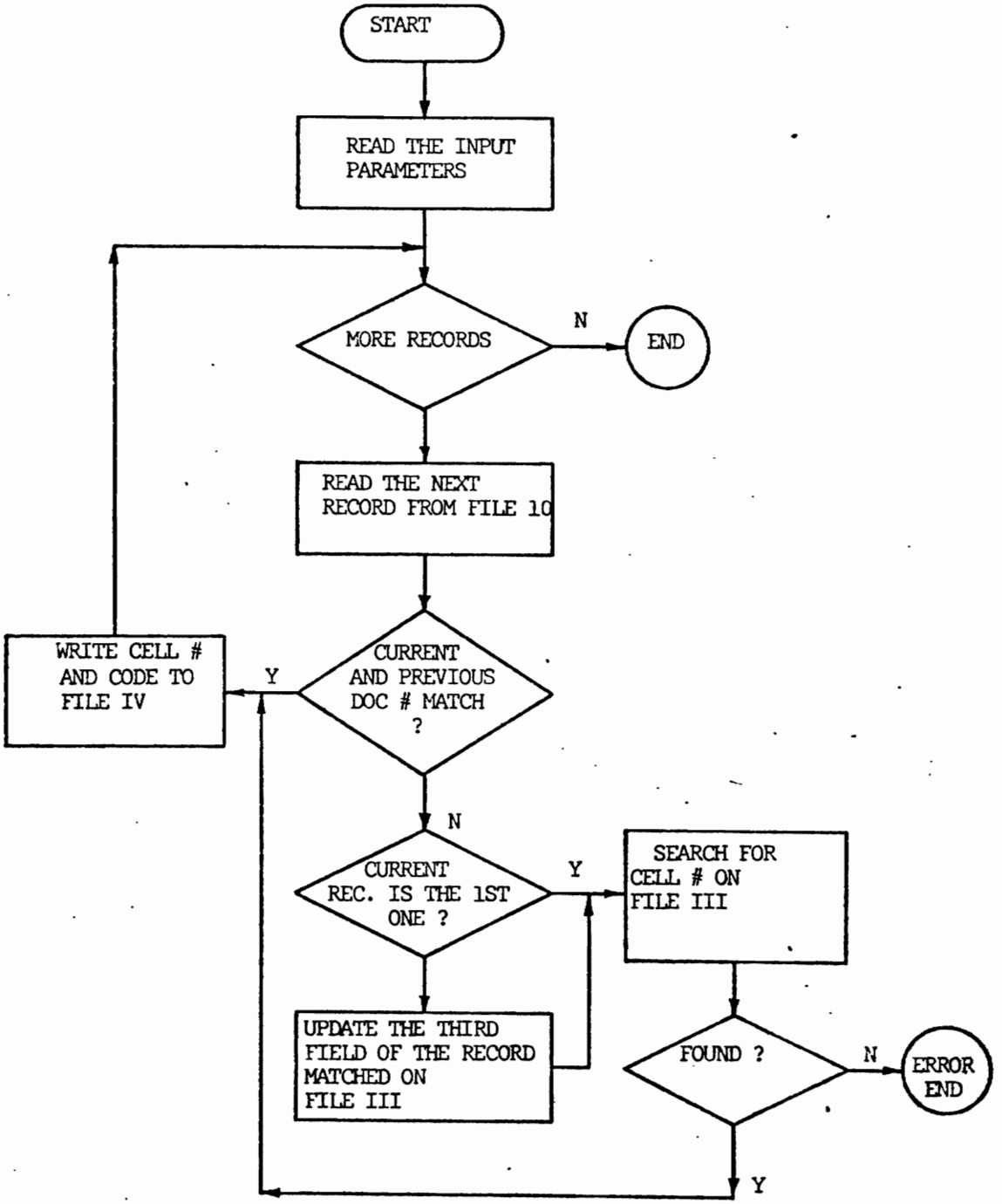


Figure C.24

General Flowchart of Program GNTRCD.

terminal node number matches the previous terminal node number, a counter is incremented and the previous code and in-cell frequency are saved, and another terminal node number is read. If the current terminal node number does not match the previous one, first the previous terminal node number, and the total in the counter are written to the output file. Then, it splits the saved information into blocks and writes each block of information to the output file.

The flowchart of program GNFKFL is given in Figure C.25.

C.20 Program TREE.

Purpose of Program: To generate hierarchical classification tree.

Program Description:

The program intersects all the cells created by the classification process, to create a hierarchy of keys.

The hierarchy generation process starts with the last N (stratification number) cells created. The keys (codes) of the N cells under a parent node (next level up the tree) are intersected and those resulting keys, along with corresponding lists of in-cell frequencies, are assigned to the parent node. However, all the keys that form the parent node, along with their corresponding information are saved

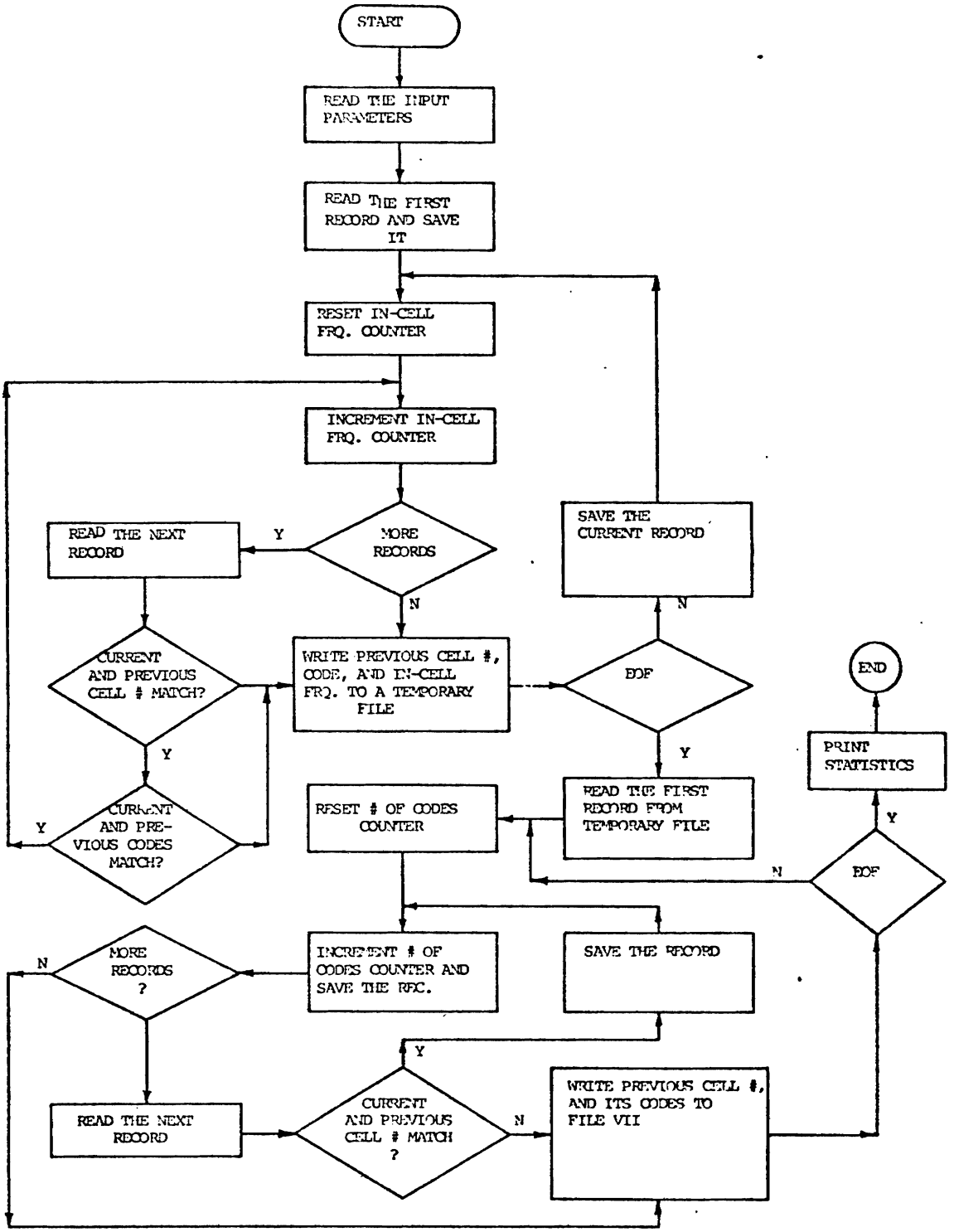


Figure C.25 General Flowchart of Program GNFKFL

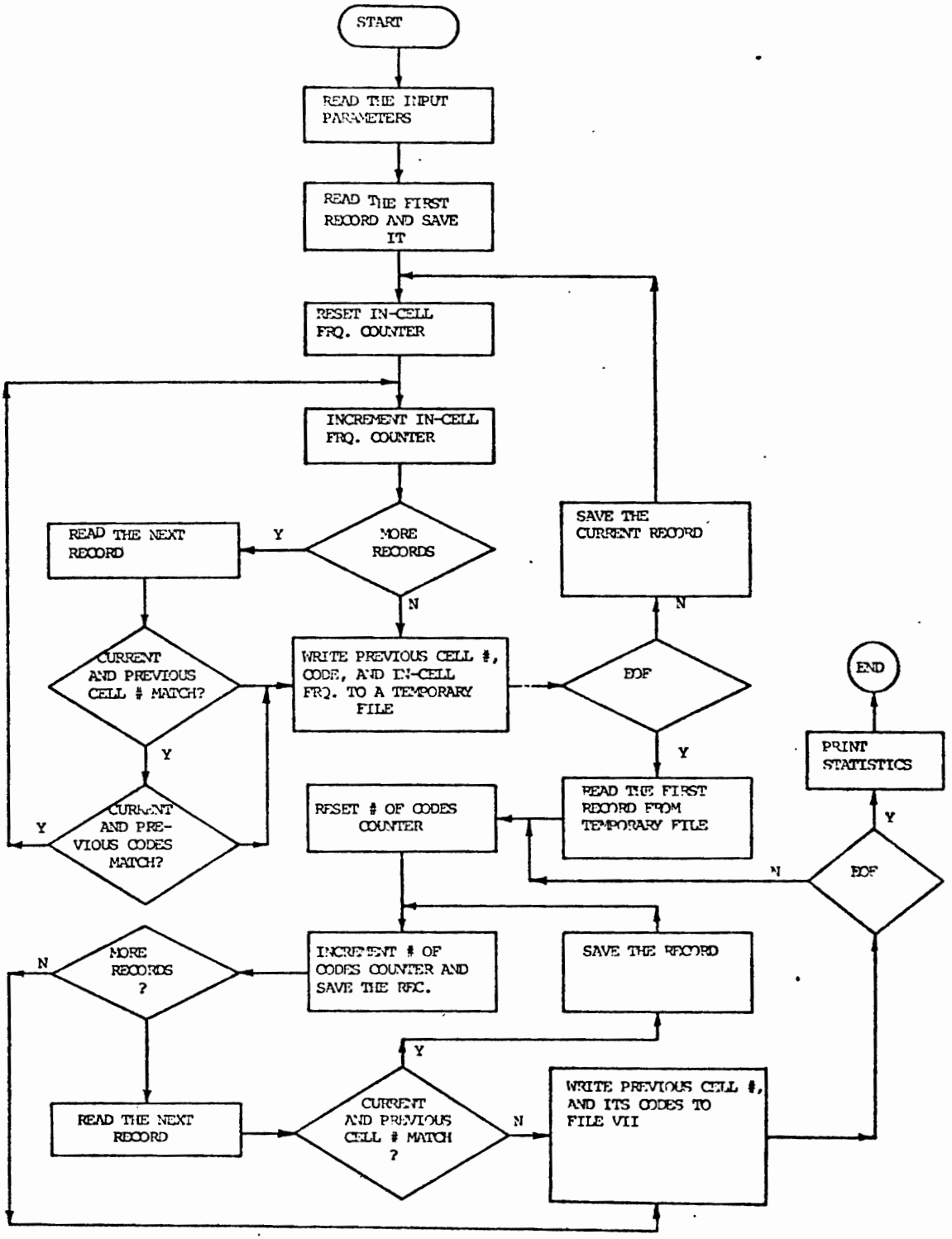


Figure C.25 General Flowchart of Program GNFKFL

on a temporary file, since as the intersection progresses up the tree, the parent node will be intersected with other "N-1" nodes to get another parent node. Also, key sets of the original N nodes are deleted of the keys assigned to the parent node. This process is continued until the root node of the tree is reached.

The balanced tree numbering comes into effect here. Because the tree numbering is balanced, the following formula is used to calculate the parent node number P for any given node number K.

$$P = \left\lfloor \frac{K-2+N}{N} \right\rfloor$$

C.21 Program NDTOKY.

Purpose of Program: To generate Node-to-key directory.

Program Description:

The program reads records associated with each node in file X and looks up each type code to obtain the corresponding key type and its total frequency. The lookup is done on file 8 or 9. It then writes to the Node-to-key directory a header-record indicating the integer node number, and the number of keys found to be in this particular node, and as many key-records as it takes to hold all the keys, along with their frequencies. At the same

time, it converts the integer node number into its canonical form, and writes the canonical node number and the corresponding key to the Node-to-key table. However, if demanded, the program creates a second version of the Node-to-key table by writing the canonical node number and its corresponding keys ordered by their total frequencies to this second table.

The program generates also another file which contains a record for each key in the tree. In this file, each record contains the integer number of the node containing the key and the list of in-cell frequencies associated with the key.

C.22 Program KYTOND.

Purpose of Program: To generate Key-to-node directory.

Program Description:

The program compares adjacent keys on the input file. A frequency counter is incremented and the information in the previous record is saved, as long as the previous and current keys match. Whenever, the adjacent keys do not match, it writes to the Key-to-node directory a header-record indicating the previous key type and the node frequency of the key, and as many node-records as it takes to hold all the integer node numbers, along with corresponding lists of in-cell frequencies, associated with this key. Also, it converts the integer node numbers into

their canonical forms and writes each key type along with respective canonical node numbers to the Key-to-node table.

C.23 Program MRGCLY.

Purpose of Program: To produce a report of the classified data-base.

Program Description:

The program produces the classified data-base directory by merging the three input files. The merging process is accomplished by first reading the Standard Formatted Text file, then reading file 20 or 6 to obtain all keys for the document and finally reading file III and using it to look up the cell number assigned to this particular document by the classification process. Then all the pertinent information is written to the Classified Data-base directory.

C.24 Program PRTCLF.

Purpose of Program: To produce a report of the Classified Data-base or to generate an Affinity Dictionary.

Program Description:

The operation of the program is controlled by the input parameter <option>=IOPT. If 1 is specified, the program reads file XV sequentially and writes onto a file each text item along with the respective cell number and index terms.

If 0 is specified, the program reads file XXII sequentially and prints out each cell number followed by the respective key and node numbers.

C.25 Program KTCLND.

Purpose of Program: To generate an Affinity Dictionary for the Retrieval System.

Program Description:

The program starts by reading a header-record from the Key-to-node Directory and a record from file XIX. It then overwrites the header record by a new record containing also the cell number, and continues reading the next node-records on the Node-to-key Directory. Subsequently, it converts the integer cell and node numbers into the corresponding canonical forms, and prints out the cell number and node numbers for each key type. However, it also saves the printed information on file XXII, which will be used to generate an Affinity Dictionary for the user.

APPENDIX D

GLOSSARY

The following terms occur with some significance in this dissertation and are defined here in order to eliminate possible misunderstandings.

Affinity Dictionary: A report listing all the index terms with their respective item classification and key classification numbers. This dictionary is ordered in ascending key classification numbers so that the affinitive index terms are placed next to each other.

Affinity of index terms: is determinable and measurable by the use of respective index terms in common text items.

A Posteriori: used adjectively, of knowledge or cognition originating entirely based on experience (examination of text items), opposed to a priori.

A Posteriori Classification: classification, where none of information needed for the classification is available separately or independently of the objects (text items or keys) being classified.

A Posteriori Indexing: indexing, where none of information needed for the indexing is available separately or independently of the objects (text items) being indexed.

A Priori: used adjectively, of knowledge or cognition originating partly or completely prior to experience (examination of text items.)

A Priori Classification: classification, where some of information needed for the classification is available separately or independently of the objects (text items or keys) being classified.

A Priori Indexing: indexing, where some of the information needed for the indexing is available separately or independently of the objects (text items) being indexed.

Automatic Classification: An algorithm and computer process to generate a classification system.

Automatic Indexing: An algorithm and computer process, for performing indexing automatically.

Candidate index term: A (word or phrase) term which has been extracted from a text item and which requires term discrimination analysis to decide whether or not it should be retained or rejected.

Cell: is a group of limited numbers of "alike" items, or keys grouped together by the automatic classification process. The cell constitutes also a terminal node of the classification tree.

Classification: An arrangement of objects in classes or groups. This is required to produce a classification system.

Classification number: A Prefix Language canonical number assigned to each node of a classification tree, based on the position of the node in the tree. Keys can be assigned to nodes of a classification tree but text items can be assigned only to terminal nodes.

Classification Schedule: A set of hierarchically structured classification rules, such as progressively more specialized subject areas, used in the classification of text items or keys.

Classification System: A scheme of organization of objects defined in a classification schedule, by which objects can be classified, namely progressively assigned to more specialized groups of objects. The classified objects may be text items or keys.

Classification Tree: See hierarchical classification tree.

Classified Data-base of text items: The rearranged data-base in which the text items are ordered by their respective classification numbers.

Common words: words whose frequency of occurrence are relatively high in all fields of interest (e.g. "the," "and," etc.)

Decomposition of phrase: decomposition of a phrase of n words into sub-phrases, consisting of $m (m<n)$ sequential words in the phrase.

Document: See text item.

Document classification: see item classification.

Document frequency of term type: number of documents to which the term type has been assigned as an index term.

Document surrogate: see item surrogate.

Document representative: is constituted by the identification number of the document and the (candidate) index terms assigned to the document.

(Hierarchical) Classification Tree: is a representation of a hierarchical classification schedule in tree form, where the more generic classification rules are at a parent node and the more specialized classification rules are at the sibling nodes.

Identification codes of term types: integer numbers assigned to the term types, to facilitate references to term types.

Identification number of document (item): a sequentially generated number assigned to each document.

Identification number of sentence: a sequentially generated number assigned to each sentence of a document.

In-cell frequency of key: number of documents within the cell to which the key has been assigned.

In-document frequency of term type: the number of occurrences of the term type within a document.

In-document term type: a unique term type within a document.

Indexing: the assignment of one or more several index terms to an object (a text item or a key).

Index term: name, concept, descriptor, affiliation, word, etc., that may be assigned to an object (text item, key, etc.) for use ultimately for classification or retrieval of the objects.

Index term (key) vocabulary size reduction:the process of discrimination and rejection of terms considered to be ineffective in classification or retrieval and thus reducing the average number of index terms per item and the number of index term types.

Item: see text item.

Item frequency of term type: see document frequency of term type.

Item classification: the generation of classification system for classifying text items and the assignment of classification numbers to text items. The classified text items, when ordered by the respective classication numbers are also referred to as the classified data-base of text items.

Item surrogate: An item identification number and the codes of index term types assigned to the item.

Key: is a candidate index term that has been evaluated in the term discrimination analysis and determined to be effective in classifying items and in retrieval. It is to be retained as a basis for classification and subsequent retrieval. Other candidate index terms are rejected in the term discrimination analysis.

Key Surrogate: A key and the item classification numbers (of cells) assigned to the key.

Key-to-node Directory: A classification schedule, in which keys are ordered alphabetically and with each key is associated the classification numbers of the nodes of the hierarchical classification tree which contain the key.

Likeness of text items: is dterminable and measurable by ths use of common index terms in the respective text items.

Node frequency of key: number of nodes at the hierarchical classification tree, which contain the key.

Node-to-key Directory: A classification schedule, in which with each classification number is associated the keys assigned to the respective node of the hierarchical classification tree.

Overindexed document: a document which is assigned more candidate index terms than a user-indicated maximum.

Phrase: a sequence of words.

Phrase Dictionary: a dictionary containing the user specified phrases.

Precision: the ratio of the number of relevant documents retrieved to the total number of documents retrieved.

Recall: the ratio of the number of relevant documents retrieved to the total number of relevant documents in a collection.

Retrieval: The process of searching and identifying of specific documents which contain the desired data being sought.

Sentence delimiter: any character which signals the beginning or ending of a sentence in text.

Standard formatted text: the text in a format acceptable by the indexing system.

Standard phrase: a composition of words with certain syntactic and structural dependency.

Stop list: a list containing common or high-usage words.

Summary Report of the indexing process: a report which indicates the necessary reduction in the index term vocabulary size and the potential impact of user elected decisions to delete terms on a mass basis.

Summary Report of the classification process: a report which indicates the progress in the classification.

Term: a word or phrase.

Term discrimination: the process of automatically discriminating index terms which do not contribute to a satisfactory classification.

Term surrogate: see key surrogate.

Term token: an occurrence of term type.

Term type: a unique value or representation that defines a term.

Terminal node: a node of the classification tree, at which no further partitioning takes place.

Text item: any aggregate of information in English Language.

Total frequency of term: the number of occurrences of the term in the database.

Underindexed document: a document which is assigned less candidate index terms than a user-indicated minimum.

User specified phrase: a sequence of specific words in a presribed order. The component words and their relative orderings form a phrase dictionary.

Word: any string of characters set off by word delimiters on either side.

Word delimiter: any character which signals the beginning or ending of a word in text.

Captain Grace M. Hopper
Office of Chief of Naval Operations
NAICOM/MIS Planning Branch
NOP-916D Pentagon
Washington, D.C. 20350

U.S. Naval Research Laboratory
Technical Information Division
Washington, D. C. 20390 — 6 copies

Commandant of the Marine Corps (Code AX)
Dr. A. L. Slafkosky
Scientific Advisor
Washington, D. C. 20380 — 1 copy

Office of Naval Research
Code 455
Arlington, Virginia 22217 — 1 copy

Office of Naval Research
Code 458
Arlington, Virginia 22217 — 1 copy

Naval Electronics Laboratory Center
Computer Science Department
San Diego, California 92152 — 1 copy

Naval Ship Research & Development Ctr.
Dr. G. H. Gleissner
Computation & Mathematics Department
Bethesda, Maryland 20034 — 1 copy

Office of Naval Research
New York Area Office
Dr. J. Laderman
207 West 24th Street
New York, New York 10011 — 1 copy

Director
National Security Agency
Attn: Mr. Glick
Fort George G. Meade, MD 20755

DISTRIBUTION LIST

Defense Documentation Center Cameron Station Alexandria, Virginia 22314	12 copies
Office of Naval Research Department of the Navy Information Systems Program Code 437 Arlington, Virginia 22217	2 copies
Office of Naval Research Branch Office/Boston 495 Summer Street Boston, Massachusetts 02210	1 copy
Office of Naval Research Branch Office/Chicago 536 South Clark Street Chicago, Illinois 60605	1 copy
Office of Naval Research Branch Office/Pasadena 1030 East Green Street Pasadena, California 91101	1 copy
Director, Naval Research Laboratory ATTN: Library, Code 2029 (ONRL) Washington, D. C. 20390	6 copies