DOCUMENT RESUME

ED 057 810 LI 003 299

AUTHOR	Watson, Peter G.; de Boer, Aeint
TITLE	Center for Information Services, Phase II: Detailed System Design and Programming, Part 5 - Summary of Phase IIA; Early Design and Experimentation Activities, Phase IIA Final Report.
INSTITUTION	California Univ., Los Angeles. Inst. of Library Research.
SPONS AGENCY	National Science Foundation, Washington, D.C.
PUB DATE	1 Jun 71
NOTE	109p.; (2 References)
EDRS PRICE	MF-$0.65 HC-$6.58
DESCRIPTORS	*Computer Programs; *Computer Science; *Design; Information Centers; *Information Services; *Library Science
IDENTIFIERS	*Computer Software; University of California (Los Angeles)

ABSTRACT

Early design activities at the Institute of Library Research (ILR) are described in this part of the center for Information Services (CIS) project. Both library aspects and computer aspects of information services from data bases are covered. The appendices present various "read routines" and an inventory of programs written or utilized by ILR staff during this phase. (For related documents see LI 003295-003298 and LI 003300 through LI 003301). (Author/NH)

CENTER FOR INFORMATION SERVICES

PHASE II: DETAILED SYSTEM DESIGN AND PROGRAMMING

GRANT GN-827

PHASE IIA FINAL REPORT

PART 5

SUMMARY OF PHASE IIA EARLY DESIGN AND EXPERIMENTATION ACTIVITIES

by

Peter G. Watson
and
Aeint de Boer

1 June 1971

Institute of Library Research
University of California
Los Angeles, California

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ABSTRACT

Part 5 of the Phase IIA Final Report on the Center for Information Services project at UCLA decribes early design activities at the Institute of Library Research. Both library aspects and computer aspects of information services from data bases are covered. The Appendices present various "read routines" and an inventory of programs written or utilized by ILR staff during this Phase.

NOTE ON REPORT FORMAT

The main body of this report was keyboarded using a video console, corrected and formatted for publication by computer using an ILR-developed program (FMS), and printed directly in the format shown.

I. INTRODUCTION AND BACKGROUND

Since 1966, the Institute of Library Research at UCLA has been engaged in a large-scale project centering upon computerized information stores ("data bases") and the prospects for their use in a university environment. The first phase of this investigation, a feasibility study, was completed and reported to the National Science Foundation in December 1967[1]. Since the present report is a direct continuation of that work, it is appropriate to discuss it briefly here.

The feasibility study of "Mechanized Information Services in the University Library" began with the postulate that computerized data bases were becoming widely available, that they represent a major new information resource that the university cannot ignore, and that, therefore, the university must begin to make plans to deal with data bases and their impact on teaching and research. A structure was needed, an organizational and technical framework within which computer processible data stores could be integrated into the existing information network of the university.

The feasibility study suggested a rational method for the university to accomplish the enormous and complex task of providing its community with information services from computerized data bases. This method entailed two innovative hypotheses.

On the technical side, there had to be found some way of avoiding the projected consequences of the programming situation as it was in 1966-7, in which each data base had its own individually tailored program or set of programs to gain access to the data. Extrapolated over only one decade or so, the idea of a university trying to service even a modest selection of 15-20 data bases, each with its own programming system, written in any one of many possible programming languages and designed to operate on any one of several possible types of computer, raised so many obstacles--technical, economic, service-oriented and administrative--as to make it clearly untenable as a long-term answer. Accordingly, a programming strategy was

---

[1] Hayes, R.M. Mechanized Information Services in the University Research Library. Phase I, Final Report (parts 1-13). University of California, Institute of Library Research, Los Angeles. December 1967.

proposed which would enable the host system to deal with any input data file on its own terms. For this, three subsystems were envisaged, corresponding to the three predominant types of data bases identified by the study--reference, full-text and numerical. The programs which comprised each of these subsystems would be task oriented rather than file oriented, permitting a given process to be executed upon any data base of the appropriate type. The university might thus sidestep the huge problem of acquiring and trying to operate many disparate program packages, and concentrate on acquisition of the data bases.

On the administrative side, an agency with the capability of processing whatever data bases it wished and of offering the needed range of services from them, did not exist. Since the proposed activities combined some of the functions of the library with some of the functions of the computing center there were three alternatives: locate the proposed agency in the library and make whatever arrangements were necessary to assure computer hardware and software support; locate the agency in the computing center and make the necessary arrangements for library and bibliographic support; or establish an entirely new organizational framework for computerized information services. Bearing in mind that such an agency would not be an Information Center, but rather a Center for Information Services, and that there was more evidence that libraries were beginning to harness computer technology than there was evidence that computing centers desired to take on the full panoply of library services, the suggested location was the library. (The very term "Center for Information Services" defines, in a fundamental sense, any library's enduring mission.) To create a new agency would incur large additional costs without any assurance of success and with the danger of needlessly duplicating part of the activities of both the library and the computing center.

During Phase IIA of the "Center for Information Services" project at UCLA (NSF Grant GN-827) the Institute of Library Research has continued to explore the areas defined in the Final Report on Phase I, namely the technical, bibliographic and administrative problems involved in creating a university-library-based capability for the provision of a broad range of information services from a broad range of data bases. In reporting the experimental activities of ILR during this period, this part of the Phase IIA Final Report deals with data base handling from two points of view--Library Aspects, and Computer Aspects. It should be recorded that the section on Computer Aspects relates only to the first half of the Phase IIA time period, after which several ILR technical staff members were transferred to the Campus Computing Network, where a programming team was being formed. Subsequent technical work by this team is presented in Part 3 of this Final Report.

II. LIBRARY ASPECTS

INTRODUCTION

The Inventory of Available Data Bases, part 3 of the Final Report on Phase I of this project, has turned out to be one of the most consistently demanded items to emerge from that study. It emphasized bibliographic reference files, but also contained examples of the other two types (numerical files and full-text files), and it concentrated mainly upon the well-publicized, nationally available material. Within this framework, it listed about 40 data bases, emanating from about 30 distributers. However, the prospect of having the university library acquire and provide service from a wide range of data bases gave rise to many questions beyond those of mere availability. In particular, what were the file characteristics of specific data bases, and the degree of compatibility between various files? What kind of effort did it require simply to read the contents of a file? What were the costs of performing certain standard processes upon them? What did the faculty, as the first group of would-be users, know about data bases, and what did they expect to see in terms of service from them? How might data bases be ordered, and how were they to be handled once they had arrived? Was the documentation accurate and/or adequate? In short, the paramount need was for project staff to gain experience in working with tapes, identifying and coping with problems, both those specific to a file and those general to some or all files. Only from hard, practical experience would it be possible to start designing programming strategies and library procedures to handle data bases in the generalized manner called for. Phase IIA thus offered both the programmers and librarians engaged in this project what most other institutions have not had, but which we find to be an almost indispensable prerequisite to any attempt at acquiring and processing data bases in an efficient and systematic manner: that is, time to experiment on a fairly modest scale with various solutions before plunging into the complexities of real-life commitment, for example through a binding subscription or through the purchase of some major data base like the U.S. Census of 1970.

The Institute of Library Research has met with considerable generosity from distributers who have provided ILR with a sample reel upon request even though they understood that we needed it solely for general research purposes and could make no commitment relative to the whole

file. But the operative phrase here is "upon request"--ILR, as a research organization, has many contacts in the world of computerized data bases, and can often arrange to have courtesy access to a file where a librarian working in a library could not. Naturally we have benefited from this situation-- it has enabled us to work with several sample files at no cost for the data. The Census Bureau in particular is to be commended for issuing a series of sample reels (based on the dress rehearsal census in Madison, Wisconsin) beginning well over a year in advance of the 1970 Census itself. A few other distributers have created test reels which they send to prospective clients. Most, however, do not appear to have grasped the sound business sense of doing this as a matter of course. We recommend that some appropriate organization of data base purchasers such as the American Library Association, or the recently formed Association of Scientific Information Dissemination Centers (ASIDIC), take the lead in defining the conditions which would make the acquisition of a sample tape, plus a certain specified level of technical documentation, a regular element in the decision-making concerning the acquisition of a file. For example, to safeguard the investment of distributers, (although this is a sellers' market and is likely to remain so for many years), a fee of about $100 might be charged for a test tape, to be applied toward the purchase price, or the first subscription that eventually resulted. There is no doubt of the benefit to the prospective purchaser in having a sample reel to "play with"; there ought to be no doubt in the mind of the distributer that it materially increases his chance of a sale.

The development of computerized information services, of course, presupposes the existence of available personnel to do the necessary work (which, as far as libraries are concerned, will usually include a re-examination of the basic library functions--acquisition, cataloging, and public service). For this, there is really no alternative except the establishment of a library systems office, however small it may be to begin with; but university libraries are aware of the need for a systems staff for many other reasons, and most now have one or have plans for one.

DATA COLLECTION ACTIVITIES

Data Base File

An ongoing task of ILR during Phase IIA has been to maintain the Data Base File, from which the Inventory was compiled, and to provide a broad base of bibliographic and published information about existing and anticipated data bases upon which selection and acquisition decisions will eventually be made. This collection is probably as complete

a body of information on generally available data bases as presently exists within a university library framework. No updated version of the Inventory was scheduled for Phase IIA, though this is certainly a desideratum later in the project. In the meantime, a somewhat augmented version of the original inventory has been published in Hayes, R. M. and J. Becker: Handbook of Data Processing for Libraries, Wiley-Becker-Hayes, (1970).

Like the Inventory which was drawn from it, the Data Base File deals mainly with the bibliographic and nationally available items--the known and marketed data bases--but there are many others. Bibliographic reference data was emphasized from the start because it was one of the first types to become generally available and, since it represents a form of information which is already known to librarians via the printed versions, is likely to be the primary concern of any library planning to acquire data bases.

In the area of numerical files, with the 1970 Census leading the way in terms both of scope and probable utilization, increasing numbers of socio-economic data bases are becoming available: statistical information on aspects of urban living such as transportation, sanitation, employment, school districting, land ownership and use, economic levels and voting patterns. In addition, there are vast quantities of raw numerical data pouring out from artificial satellites, and many large hospitals now utilize computers for a variety or tasks, among them the gathering and analysis of patient-monitoring data which later can assume added value as part of a statistical file of biomedical data.

As examples of full-text data we may point to the statutes and constitutions of all 50 states, and the growing number and range of literary texts upon which computer-based experiments are being conducted. A good indication of the extent of the work now going on in the humanities (also covering much of the social sciences) can be found in the "Directory of Scholars Active" published periodically in the journal Computers and the Humanities. The main body of this part of the Final Report is an example of full-text data. The text has been keyboarded via a video console, formatted for publication by computer using the Format Manipulation System (FMS) described in Appendix G, and can be further manipulated by computer in either the original (keyboarded) form or in the formatted form.

The important point is that if and when libraries really take control of this new information medium, numerical and full-text data are going to be at least as important as the bibliographic data, and appreciably more difficult to handle.

A comprehensive inventory of all of these data bases is a basic national requirement; one whose execution is likely to be large enough to demand a project of its own.

Tape Documentation File

In Phase IIA, a Tape Documentation File was created. This file has the technical documentation about a data base, as opposed to the bibliographic information, and contains such items as:

a. A copy of the original documentation accompanying the tape.

b. A report by the staff member responsible for initial work such as "opening" the tape and identifying the elements of the file structure.

c. Any relevant printout associated with (b), e.g. the dump, the read program (plus the card deck if possible) and a specimen of formatted printout.

d. Any technical correspondence, e.g. to the distributer or to another UC campus, itemizing gaps in the original documentation, discussing file characteristics, etc.

e. Later work as it occurs, e.g. a record of what was being attempted, plus some sample printout.

The establishment of such a file by the purchaser or processer of data bases will help to guarantee the order and continuity of his work with tapes, for example if--as is envisaged with CIS--the user will ultimately be able to create his own programs to manipulate the data. At ILR, the Tape Documentation File was assembled with the additional intention of providing technical continuity when the UCLA library eventually assumes responsibility for the CIS system.

Program Inventory

An inventory of programs and subroutines was prepared, which is reproduced as Appendix G. It contains various subroutines and complete programs written or utilized by the ILR staff for experimentation with data bases and supporting activities.

WORKING WITH DATA BASES

Sample Files

Throughout Phase IIA it was imperative that project staff familiarize themselves with the structure and scope of as many different data bases as was reasonably possible. In this way both groups of staff, librarians and programmers, would attain a sense of what was typical in data bases and what was unique to a particular file, and gain insights into the whole range of problems surrounding generalized file management and information services; problems that range from the smallest technical question to the broadest administrative issue. During the reporting period, ILR project staff had access to a representative cross-section of files and sample files as follows:

American Institute of Physics--SPIN/0 bibliographic file (sample)

American Petroleum Institute--bibliographic file (sample)

Aspen Corporation--California constitution and statutes (sample)

Atomic Energy Commission--Nuclear Science Abstracts (sample)

Bureau of the Census
- --Census of Households (1960), 1/1000 sample
- --Census of Housing and Population (1970), first and second counts, test reels

Chemical Abstracts Service--CA Condensates, January 1969-

Communications of Behavioral Biology (sample)

Engineering Index, Inc.--COMPENDEX (sample)

ERIC (USOE Education Resources Information Center)
- --Current Index to Journals in Education, July-September 1970
- --Research in Educational, complete to August 1969, and July-September 1970
- --Thesaurus of Descriptors

Library of Congress
- --MARC distribution service, 1969-
- --Subject headings, 7th edition

National Library of Medicine--MEDLARS

Standard and Poor's Corporation--COMPUSTAT

Webster's New Collegiate Dictionary--7th edition

Obviously, the level of involvement varied from file to file, depending upon the type, quantity and availability of data and the nature of the work being undertaken, but the end result - a fund of experience - is more significant than how one specific file was treated.

Read Routines

A number of read routines were prepared as an exercise in understanding the immediate, practical problems which occur when a new file of data arrives. Six of these, for the American Petroleum Institute file; the U.S. Census of 1960 1/1000 sample; Communications of Behavioral Biology; COMPENDEX; Nuclear Science Abstracts; and CA Condensates are presented as Appendices A to F to this part of the Final Report.

It should perhaps be clarified that a "read routine" in this sense is not merely a matter of mounting the tape and asking the computer to print out exactly what is recorded thereon. This latter process is customarily known (for obvious reasons) as getting a "dump" of the contents, and in most computer systems there are standard utility programs to do it. However, what emerges from a "dump" is a machine representation of the information, encoded and perhaps doubly encoded, forming an oddly-arranged stream of characters unintelligible to anyone but a programmer who has read the documentation. A "read routine" is the program which instructs the computer to read what it finds recorded and to print out the information, formatted and readable by humans. This in turn involves telling the computer the file structure--how the various fields of data comprising a record are arranged, where and how to locate each field and assign its correct name, what is the maximum number of characters allowed for each field, how to recognize print-control characters and suppress them in the printout, what to do when a blank field is encountered, etc. etc. This can become a complex and time-consuming operation; it can take a skilled programmer a man-week or more of work, or anywhere from 10 to 30 runs on the computer, to effectively "open" the tape, i.e. bring matters to the point where the organized retrieval of information can begin to be attempted.

Documentation

The technical documentation purports to be the detailed and complete description of the file structure and how to access it, and is therefore an indispensable operating manual in any data base system. If programs are being provided, there will customarily be descriptions of them as well as of the file, but since our investigations are directed toward the establishment of a CIS which will not be purchasing or utilizing distributers' programs, this discussion essentially relates to file descriptions. Project staff have identified six closely linked criteria by which prospective purchasers, in particular library staff, can evaluate the utility of technical documentation:

Promptness. Documentation can precede, accompany or follow the arrival of the file to which it refers--there are practically no guides to this at present, because the "marketing" (in the broadest sense) of data bases is only just becoming a recognized, systematic publishing activity. Obviously, if the documentation comes early--especially if it was unsolicited and is being used as a (supposed) advertisement for the product--it has little significance, having no immediate referent; if it is late there is virtually no progress to be made with the tape until it arrives. The too-early provision of documentation may signal that it is so readily available because it is no longer fully up-to-date, and conversely, the distributer's natural desire to supply the latest documentation may mean one has to wait for it after the arrival of the file, or the first portion thereof. The next issue to examine is, therefore, that of currency.

Currency. Assuming that it arrives on time, the documentation must be up-to-date. The fact that the state-of-the-art of data base creation is still very volatile has led, in our experience, to a tendency to supply recently superseded documentation, usually with some informal updating such as hand-written alterations in the text, brief notes in a personal letter or, more rarely, mimeographed insert pages. This can be attributed to the natural logistical inertia of the printed medium; when much concentrated effort has been expended in getting out a document, it is both economically and psychologically impossible to avoid treating it as final and authoritative, dealing with any amendments as minor errata and corrigenda, best handled in the context of the overall correctness of the main document. It seems clear that machine methods of Technical Document storage, manipulation and reproduction could well constitute part of the solution to this inertia.

Accuracy. If what is supplied is current information as far as the distributer is concerned, the next question is "Is it also accurate?"--and it is self-evident that it should be. A comment on typography may be made here: while a simple clerical error within a word (e.g. letter transposition) usually causes no trouble, the same type of error within a number (e.g. digit transposition) is almost invariably undetectable until an inordinate amount of checking has been done, by which time a great deal of time, money and effort has already been lost. If the blocksize is given as 5600 and it should have read 6500, the computer may print out the information (assuming all else in the read program is working) minus the last 900 characters, or it may not be able to begin.

Completeness. Even if the documentation is on time, is current and is accurate, one must always try to determine whether what is supplied is complete. In fact, experience so far shows that simple failure to itemize or mention certain of the details that the programmer needs to know is a more prevalent fault that the outright mis-stating of them, and is therefore potentially a larger problem for the library. Up to this point, creating data base file structures and using the computer to retrieve from them has been a sufficiently exclusive activity for communication to be built on unspoken assumptions, and the natural tendency not to bother documenting something you assume everyone in a relatively small field takes for granted can lead to horrendous problems at the point of acquisition. For example, when an ILR staff member comments:

> The file has a tape mark at the beginning where one normally expects to find a label or nothing at all. This was not mentioned in the accompanying documentation and led to some waste of time and effort... (Emphasis ours)

The point is not so much whether she was correct in her assumption (perhaps the writer of the documentation would have disagreed that "one normally expects to find a label or nothing at all") but that she had to make the assumption. That particular file caused certain additional problems, and the tape was "opened" in 29 runs at a machine cost of $46.08. Add to that the full-time equivalent of about one week of work by an experienced programmer (for whom a reasonable average salary estimate is $1100 per month) and the true cost is revealed to be in the region of $300.00 which quite clearly cannot be contemplated for each of the 15-20 data bases that it is postulated a CIS would acquire in its first few years of operation.

Stylistic Clarity. By this we mean such things as terminological precision, systematic presentation, avoidance of ambiguities, etc. This too can cause disproportionate trouble. Such statements as "the file consists principally of..." without specifying what else may be present, are needlessly vague; and the following from some early ERIC documentation shows a confusion over the use of the words "record" and "block" that can cause trouble for the programmer:

> ERIC Report Resumes are stored on magnetic tape (9 channel, 800 bpi) in accession number sequence in the form of one or more IBM System/360 Operating System variable length records. Each ERIC Report Resume equals one record on magnetic tape. These records are grouped in variable length blocks which have a maximum length of 32,000 bytes. (Emphasis ours)

The point of this illustration is by no means to castigate the ERIC system (which, taken all round, is actually one of the better ones to work with) but to draw attention to one perennial source of confusion in the effort to describe for someone else exactly what is on a reel of magnetic tape, and that is the widespread uncertainty about a group of concepts relating to the units of mechanized information storage (especially on magnetic tape), such as physical records, logical records, blocks, blocksize, blocking factors, etc. Many people--including programmers and system analysts--do not seem to be quite sure that you understand these items to mean what they understand them to mean. The prevalence of the IBM System/360 computers has fostered some standardization of concepts, but it is a circumstantial, rather than a theoretical, phenomenon, and could just as easily evaporate with a change in the market.

Listing a data element called "Access Words", and neither specifying that none occurs in the entire test tape nor explaining how "Access Words" (were they to appear) would differ from subject terms; repeating the "Subject Heading and Subheading" field, the first time linking the subject term to its subdivision by a dash and adding a document number into the field, the second time linking the elements with a comma and omitting the document number (which also appears, unannounced, at the end of every abstract); referring in the documentation to a "Card Service Category Code" whereas it appears to be designated "Sales Code" on the machine record itself -- these are typical examples of the imprecise and indeed sometimes careless nature of much documentation as presently found. Any librarian who must deal with it will be disappointed at the general laxity--especially since relatively large sums of money are invariably at stake--and, incidentally, vis a vis reference files, will be particularly amazed at the variety

of ad hoc methods of giving a bibliographic citation in the record, representing decisions already taken which will be practically impossible to correct retroactively in time to come, when library networks have made standardization of formats, of catalogs or holdings lists, and of documentation an economic necessity.

Level of Formality. As indicated above, significant and useful material relating to the technical description of a file often appears in an informal fashion; as comments embedded in a personal letter, as hand-written addenda, or as loose-leaf photocopies of what are obviously internal work-sheets, flow-charts, translation tables, etc. It appears that the best compromise between formality (implying here delays and also rigidity) and informality (implying here scattered, inaccessible, and sometimes incoherent information) would be to have a basic printed document in loose-leaf form, highly modular in arrangement, and an agreement to keep it updated. As noted above, use of the computer to produce and update this document should be tried where feasible. A trend toward loose-leaf manuals is apparent, but since most never get updated, that special virtue is often lost.

Field Names

A brief study was done at ILR to determine whether a common set of data elements could be defined for bibliographic files; five serial bibliographic data bases and their documentation (CA Condensates, Communications of Behavioral Biology, COMPENDEX, NSA Entry file, ERIC Report Resumes) were examined. From these alone, it was possible to list 45 different bibliographic elements (see Figure 1), and if the MARC record for either serials or monographs had been added, the total would have been about 115 (MARC for monographs has over 50 fields currently in use, and another 20 defined). Four of the files contained at least one bibliographic element that appeared to be unique, although this is often difficult to ascertain precisely, because frequently the "same" element, under a variety of names (e.g. Subject terms, Keywords, Selector terms, Descriptors, Access words, Index terms, etc.) has a slightly different structure, giving it a slightly different potential for retrieval.

Machine-readable bibliographic files are notable for a propensity to give several numbers to the citation: one or more accession numbers, an abstract number, citation number, report number, etc. The designers of MARC found it profitable (even for monographs) to divide the many possible numbers into Control Numbers and Knowledge Numbers. The use numbers is complemented by a series of codes, usually one

Figure 1

Bibliographic Data Elements

Data Elements	CA	CBB	COM	NSA	ERIC
Abstract		x	x		x
Abstracting journal; vol, issue nos.	x				x
Address of principal author	x				
Affiliation				x	x
Assignee (corporate owner of patent)				x	
Author(s), corporate	x	x		x	
Author(s), personal	x	x	x	x	x
Availability (distribution)				x	
Citation number		x			
Coden	x			x	
Conference preprints (society, date, pages, and price)				x	
Contract number				x	x
Corporate code				x	x
Date of publication, month		x	x	x	x
Date of publication, year	x	x	x	x	x
Date(s) (patent)				x	
Descriptor terms	x	x	x	x	x
Drop note				x	
Field group (codes)				x	
Field group (words)				x	
ID number (accession number)			x		x
Journal title (full)			x		x
Journal title (abbr)	x	x		x	
Journal volume number	x		x	x	x
Journal issue number	x		x	x	x
Language	x			x	
Microfilm code			x		
Pagination (page numbers, e.g. pp. 12-220)	x		x		x
Pagination (number of pages, e.g. 12p.)		x		x	x
Patent number				x	
Place of publication				x	
Price				x	x
Publisher				x	
Report number				x	x
Sales codes			x		
Secondary number				x	
Security code			x		
Series title				x	
Short title				x	
Source article				x	
Source of information				x	
Subject heading			x		
Title (in English)	x	x	x	x	x
Translation note				x	
nacular title		x		x	

digit or character in length, specifying the existence or otherwise of various conditions. For example, a "Security code" might allow the binary choice of "Yes" or "No" to the implied question "Is this document for general distribution?" or it might allow a graduated series of responses to reflect several different levels of availability. The numbers, tags and codes spread throughout a machine-readable bibliographic record (which frequently serve to make the machine record a significantly different entity from its printed counterpart) are the means to take advantage of the computer's ability to do very precise retrieval and high-speed sorting. Any field which is a computer-identifiable unit, a "data element", can be retrieved from each record, and similarly any field can be designated a sort key and used to generate subject indexes, author lists, title lists, chronological lists, etc.

It was rapidly decided that attempting to establish a canonical set of names for data elements having different names in different files was not a feasible approach to take. Subsequent work (at CCN) was therefore oriented toward creating individual read routines for each file.

Test Profiles

Chemical Abstracts. The CA Condensates Search Service at UC Riverside (see part 2 of this Report) was utilized in order to give ILR and Library staff some practical experience with constructing profiles, analyzing the results of searches, refining the profiles, and with gathering statistics on computerized information retrieval. In the CIS Seminar of April-June 1970 (see part 6 of this Report) participants had submitted searches as a class exercise, and these were first analyzed be a Research Assistant, who did the corresponding manual search for each question. Since this was a sample of only eleven profiles, most of them written by non-chemists and therefore of a general nature, it was not the purpose to do anything more than use these profiles as a broad indication of how the system functioned, and as a guide to the keeping of useful statistics on computerized searching. The following is a summary of the analysis (it should be remembered that this was Chemical Abstracts Service's old format, prior to the introduction of the Standard Distribution Format).

Phase I: Analysis of Eleven Profiles and Hits.

The profiles were prepared by librarians participating in a seminar on computerized information services in libraries. The seminar was held during the Spring Quarter, 1970, at the UCLA School of Library Service, and it was conducted by the Institute of Library Research.

The profiles were run against tape 72:15 of CA Condensates at the Riverside campus of the University of California under the direcion of Mrs. K. Forrest.

Examination of each profile proceeded by:

a. Reading the question,

b. Searching the keyword subject index,

c. Reading the abstract and deciding if it answered the question,

d. Listing those abstracts deemed relevant to the profile, and

e. Reading the abstract (hard copy) of all hits made by the computer and recording what possible term-matches were satisfying the profile.

There were 4,916 entries on the tape used (74577 to 79493). Based upon information found in Preparation of Search Profiles (a Chemical Abstracts Service publication), an assumption was made that hits result from term-matches of the profile and terms in the title, and/or the keywords stored on the tape. Keywords were understood to be information containing words which appear also in the abstract. Many were not identifiable in the hard copy of the abstract. This confirmed a report[1] from the Computer Center at the University of California at Santa Barbara in September, 1969:

> Only the title and a selected number of keywords appearing in the abstract are reproduced on tape, but not the entire abstract.... This relationship between keywords and the abstract is not always apparent. It was found repeatedly that while a search term was not found in the abstract, it nevertheless appeared on the tape as evidenced when part of the tape was printed out. Since the entire retrieval system is based on a term-match basis, this situation certainly presents a problem.

---

[1]Bellomy, Fred and Wong, R.S. Evaluation of the Chemical Abstracts SDI System. University of California, Santa Barbara, September 3, 1969. Part I, p. 3.

Keywords were not necessarily taken from the abstract, but were sometimes the result of an abstractor's or editor's intellectual interpretation of the content of the abstract. The problem, as viewed from the standpoint of the user or the profile analyst, was: "Is there a significant difference between the retrieval from the tape (title and selected keywords) and retrieval based on term-matches with the title and abstract found in the hard copy? How does machine retrieval compare with manual retrieval using the subject index?" For an example of the various possibilities of retrieval, it might be interesting to consider the profile for question no. 306, "Information on nutritious diets for pigs". Seven hits were made for which no term-matches were found in the hard copy of the abstract; term-matches <u>were</u> found with the keyword subject index. This indicates that the keywords included for this series of abstracts were not taken entirely from the abstract. Three abstracts found by manual search were not retrieved by computer even though term-matches were found in the title or body of the abstract. Seven hits were made by the computer which might be judged relevant, which were not found in the manual search. To generalize, 84% of the possible hits were made by the machine; 33% of the hits made were not traceable to term-matches in the abstract; 1.2% of the possible hits were not made by machine, but judged retrievable (containing parameters to satisfy term-matches with the abstract).

Profile 316 provided an interesting study in interpretation and profile construction: "Biochemistry and pharmacology of aggressive behavior". The Research Assistant first interpreted this question to mean agressive behavior in man, while the individual who constructed the profile did not exclude other forms of life. When assisting someone in construction of a profile, it is necessary to ascertain precisely what is actually desired. Profile 314 asked for "Information on the accumulation and effects of Strontium 90 in mammals". The descriptors (in two parameters) related only to mammals as a unit; an interpretation of parts of mammals affected by Strontium 90 (e.g. livers, bones, etc.) would have resulted in a larger group of retrieved abstracts.

The keyword subject index, found on the pink pages of every issue of <u>Chemical Abstracts</u>, is not always easy to use. For example: an abstract pertaining to profile 302 ("How do widely used synthetic chemicals released into lakes, streams, oceans, affect marine animals used as food?") was missed (77865a) because the keyword subject index access was only by "mercurals grain toxicity" and "grain toxicity mercurals". There was no access by index terms such as "pollution", "toxicity", or "marine animals" even though the abstract did deal with these concepts. Note that the abstract was retrieved in the machine search.

There is only one entry for abstract number 77944e, under "DDT fish birds". Abstract number 76327u cannot be found under "pigs", but only under "dietary pig yeast". Manual searching of the keyword subject index, therefore, does not guarantee retrieval of all relevant abstracts relating to a profile. How do most people conduct a search of current issues of Chemical Abstracts? For those whose interests are confined to a narrowly definable field (within the CA coverage) scanning of the titles of a section is the most rewarding. For those whose interests are of interdisciplinary scope, the keyword index must be consulted.

Hits per descriptor combination: about 50% of the descriptor combinations in profiles 304 and 306 did not retrieve any hits, while 20% retrieved 65% of the hits. For profiles containing many descriptor combinations (over 20), the above ratios do not hold. Of the 330 combinations for profile 317, about 313 did not retrieve any hits (95%); together, two retrieved 7 hits of a total of 17 (41%); each of ten descriptor combinations retrieved one hit (5.9% each). The above estimates were based upon examination of the descriptors and the hard copy.

Phase II: Construction of a Profile; Prediction of Hits.

Chemical Abstracts 72:15 was used to select a subject--Gas Chromatography--which would yield a large number of hits. Some obvious terms relating to the subject were omitted so that the number would not exceed 100. The question was chosen for its interdisciplinary application within the broad field of Chemistry. Prior to the machine search, those abstracts were listed which contained term-matches between the hard copy and the profile. A list was also made of abstracts which did not lead to a match with the terms of the profile, but which (because of the abstract's content) would generate proper term-matches with the keywords in the machine record.

Of a possible total of 97 hits, the machine search retrieved 74 (76%). In Figure 2, list B contains the number of abstracts predicted to be hits and of hits actually made; list C shows the number of abstracts with no precise term-matches but which were predicted to be hits on the basis of keywords on the tape; and list D has the number of abstracts which were retrieved by machine and not manually.

Of the 74 hits made by term-matches in the machine search, 7 could not be related to the hard copy. Of the 67 remaining, 19 were retrieved by one logical combination of descriptors, 14 by another, 11 by a third. There were 18 such combinations possible, 8 of which produced no hits.

Figure 2

CA Condensates Test

A.	Possible hits	97
	Hits retrieved by machine	74 (76%)
B.	Predicted hits.	82
	Hits made (from predicted group).	59 (72%)
	Hits not made (from predicted group).	23
C.	Predicted hits with no term-match in abstract .	3
	Hits made with no term-match in abstract. . . .	3
D.	Hits not made manually - no term-matches. . . .	4
	Hits not made manually - with term-matches. . .	8
	Total hits not predicted.	12

Education-Psychology Data. In August and September, 1970, preliminary planning and committee work were undertaken towards the establishment of a test SDI project to be run by the Education and Psychology Library working with CIS project staff. Arrangements were made for a controlled group of about 10 to 15 faculty from the departments of Education and Psychology, and negotiations began with the American Psychological Association to obtain a part of the Psychological Abstracts data base. At the same time, the UCLA Library provided funds to purchase 3-month update tapes (July-September 1970) of the two ERIC files, Research in Education (RIE) and Current Index to Journals in Education (CIJE) ($50 each). As a trial procedure, they were ordered using the library's regular acquisitions routines.

Since the CIS project has already developed a versatile set of reference retrieval programs for use with the ERIC (RIE) data base, an SDI experiment in this field was seen as an appropriate vehicle by which to gather information and experience on several points:

a. How will the faculty respond to the availability of this type of service from their disciplinary library?

b. Can the disciplinary library at this point begin to assume a primary role in selective dissemination activities (i.e., did the first CIS Seminar lead to positive, tangible results)?

c. How will the Library respond to the need to order files of data on tape under carefully observed conditions?

d. How successfully can these data bases (three files from two sources) be processed and searched for the benefit of one fairly homogenous intellectual group: what problems can be brought to the surface which are likely to be of a general nature in the handling of multiple data bases--problems of file manipulation, profile construction, user acceptance, etc.?

No adequate response was obtained from the American Psychological Association concerning Psychological Abstracts, and so the experiment proceeded with the two ERIC files. The programs for searching RIE, mentioned above, had been designed as modular search programs, and they were used to process CIJE after only minimal changes.

This is an ongoing project, and a more detailed report will form part of a later CIS Report (probably the Final Report on Phase IIB) but by the end of Phase IIA, several important conclusions could be drawn. The Library at UCLA will probably be able to handle tape acquisitions with only small procedural changes, one of the most pressing of which, the checking of incoming material so that funds can be disbursed, has been worked out satisfactorily in conjunction with the Systems department. Discussions on acquisition questions are continuing, as are discussions on the cataloging of ERIC tapes. Some trial cataloging of the ERIC material was circulated internally for discussion, and used as an excellent "real-life" example in the sessions in cataloging in the second CIS Seminar (see part 6 of this Report). The quarterly availability of ERIC tapes, although it has not been a problem in this initial phase since there were all kinds of other "set-up tasks" to be done, threatens to be too infrequent for a regular well-organized SDI service. The user can, after all, see the printed copy of the journals every month. The common subject terminology is a great asset to the system as a whole, although there are certain difficulties with the use of the thesaurus. Also, it was not clear at the outset whether the tape version of RIE contained the entire contents of the printed version (ED- and EP-numbered documents) or only the report resumes (ED): in fact it was the latter. Owing to the necessarily small number of staff engaged upon this trial service and the quarterly availability of the tapes, feedback has been somewhat slow in accumulating, but faculty response to the search results was encouraging--in most cases, a high rate of relevance was obtained (about 70%) after one or two modifications of the profile.

III. COMPUTER ASPECTS

INTRODUCTION

Although Phase IIA has been concerned primarily with the design of the overall CIS system - administration, procedures, software and hardware - a considerable amount of experimental programming was performed to support this effort.

Initially, this programming was performed by the ILR staff primarily to gain a familiarity with various aspects of the software and hardware problems. Key areas of concern were:

a. Hardware
b. Operating systems
c. Programming languages
d. Software system programming strategies

In mid-1970 several members of the ILR staff were transferred to the Campus Computing Network (CCN) and the emphasis shifted toward experimental programming to clarify issues in the actual design of prototype CIS software and production programming to support experimental operation with the CA Condensates tapes.

HARDWARE

Phase IIA began with the premise that the CIS software system would be centered around an IBM 360/30 located in the University Research Library and would have available the services of an IBM 360/91 at the computing center for tasks beyond the economical capabilities of the library's computer.

However, with the advent of a university-wide budget squeeze, the library has scaled down its planned computer to an IBM 360/25. This in turn has shifted the emphasis of the CIS software system to the IBM 360/91 with the library's computer performing as a small stand-alone computer for those tasks which it can perform, or as a remote job entry and input/output station to the IBM 360/91 for those tasks requiring the larger computer.

OPERATING SYSTEMS

The CIS software system is being designed to operate on the IBM 360/91 at the Campus Computing Network at UCLA. This computer uses the IBM Operating System/360 (OS/360) with the MVT (Multiprogramming with a Variable number of Tasks) option.

While the CIS software system was still thought of as being centered around an IBM 360/30, however, it was recognized that it would be highly desirable to have compatible operating systems on the two computers. This would allow the computers to share common programming languages, programs, data set labels, etc. For this reason, a short study was made to determine if it was feasible to use the IBM Operating System/360 with the PCP (Primary Control Program) on a 65K IBM 360/30.

This study concluded that it was indeed feasible, and went so far as to design a possible operating system configuration for the combination. The impossibility of acquiring an IBM 360/30, however, has reduced any interest in this study to a purely theoretical level.

The currently envisaged IBM 360/25 will certainly use a software system supplied by CCN when operating as a remote job entry and input/output station to the IBM 360/91. This software will probably be very similar to that currently available to support the two local IBM 360/20's presently operating in similar capacities. Operating as a stand-alone computer the IBM 360/25 will probably use the standard IBM Disk Operating System (DOS). While this is an adequate operating system in its own right, its incompatibilities with OS/360 on the IBM 360/91 will cause many problems. Amongst these are:

a. Incompatible tape and disk data set labels. A data set written by one system cannot be conveniently read by the other.

b. Incompatible programming languages. Programs written for one system cannot be used directly on the other.

c. CCN personnel, while highly competent with OS/360, have little or no experience with DOS.

These factors combine to push the IBM 360/25 into the background and assign to it the primary status of remote job entry and input/output station at this stage of the CIS project. This current status does not preclude greater utilization of the IBM 360/25 at a later stage, particularly during actual operation, when library personnel will become responsible for the project.

PROGRAMMING LANGUAGES

The selection of a programming language for a given computer application has been, historically, largely a matter of intuition. ILR felt that language selection for the CIS project should be placed on a more rational basis. To this end, a set of criteria for language selection was established and a number of available languages were rated subjectively by these criteria. Then a selection was made on the basis of the ratings. Happily, both intuition and language ratings agree on PL/I and Assembler as the programming languages to be used for the CIS project.

Criteria

The criteria for language selection fall into three categories:

a. Language - based on the definition of the language.

b. Implementation - based on an implementation of the language.

c. Environment - based on the (local) environment.

These categories and the criteria within them are not entirely independent. It is hoped, however, that any interactions have not significantly biased the conclusions.

In addition, the nature of the application acts as a constraint in rating a given language by a given criterion.

Language Criteria. The first six of these criteria are loosely grouped under the heading "Programming Time" in that a high rating will tend to reduce the amount of elapsed time required for programming.

1) Problem oriented notation. Is the notation (syntax) of the language "natural" for the problem at hand?

2) Range of operations. Does the language include a sufficient range of operations to handle the problem at hand, or is it necessary to resort to programming "tricks" or other measures?

3) Volume of coding. Is the language sufficiently concise to minimize the probability of errors due to sheer volume of coding?

4) Ease of learning. Is the language, or a subset of the language, easy to learn and apply to real life problems?

5) Ease of coding. Is the language easy to code, or is it prone to difficulties due to complexity, coding restrictions or other reasons?

6) Ease of maintenance. Is a completed program easy to document, understand and modify?

7) Knowledge of machine. To what extent is a knowledge of the machine and/or machine language required to program in this language?

8) Interaction with OS. To what extent are the services of the operating system available in the language?

9) Ability to segment code. Does the language allow the segmentation of programs into smaller units, subroutines or functions?

Implementation Criteria. The first three of these criteria are grouped under the heading "Efficiency" and include generally recognized measures of the efficiency of a language implementation.

1) Compilation time. Does the program compile rapidly? This criterion declines in importance as the ratio of execution time to compilation time increases. In a production environment its importance is minimal.

2) Execution time. Is the object code efficient in terms of execution speed? This criterion increases in importance with the maximum execution time for a program. A 100% increase in time due to inefficiency may be tolerable for a one minute program and intolerable for a one hour program.

3) Object program size. Is the object code efficient in terms of the amount of storage used? This criterion decreases in importance as the amount of available storage increases.

The next four criteria are grouped under the heading "Debugging" and serve to measure the degree to which the implementation assists in the debugging of programs.

4) Compilation diagnostics. Are the compile time diagnostics complete, accurate, understandable and indicative of the actual problem?

5) Execution diagnostics. Are the execution time diagnostics complete, accurate, understandable and indicative of the actual problem?

6) Compilation listings. Are the listings provided by the compiler complete and informative? Do they provide adequate documentation for a finished program?

7) Knowledge of machine. To what extent is a knowledge of the machine and/or machine language necessary for successful debugging?

8) Interface with languages. How easily can routines written in this language use or be used by routines written in other languages?

Environment Criteria.

1) Evaluation of prior use. What is the overall opinion by users of this language as to its suitability for the problem at hand?

2) Support by ILR. To what extent are ILR staff members familiar with and able to use or offer advice in the use of this language?

3) Support by CCN. To what extent are CCN staff members able to use and offer advice in the use of this language?

4) Support by vendor. To what extent is the vendor able to supply documentation, compilers, and advice in the use of this language? Is support available for the isolation and correction of errors or bugs in the implementation of the language?

5) Programmer availability. Are programmers proficient in this language available for hiring?

6) Convertibility. To what extent are programs in this language convertible for use on other computers of the same series or to other dissimilar computers?

Candidates

In making these ratings six languages were selected as candidates:

a. ALGOL (Level F)
b. Assembler (Level G)
c. COBOL (Level F)
d. FORTRAN (Level H)
e. PL/I (Level F)
f. RPG (Level E)

Several of these languages are available in more than one implementation (Level) on the IBM 360/91. For these ratings, the most powerful implementation available has been chosen.

Ratings

Each of the six candidates was rated on a scale of 1 to 5 (poor, below average, average, above average, excellent) for each of the twenty-three criteria. The result of this process is given in Figure 3.

Analysis

A superficial examination of the ratings would rank the languages in the order: PL/I, FORTRAN, COBOL, Assembler, ALGOL, and RPG.

A more critical examination of the ratings, effectively applying weights to some of the criteria, reveals additional information. Perhaps the most important criteria are "Range of operations" and "Evaluation of prior use", both indicative of the appropriateness of the language to the problem. By these criteria ALGOL, FORTRAN, and RPG are judged to be inappropriate.

The ratings of PL/I, in general, dominate those for COBOL, particularly in the "Range of operations", "Debugging", and "Environment" criteria. The only areas where COBOL dominates PL/I are "Execution time", "Object program size", and "Convertibility". These observations indicate that PL/I is more appropriate for program development, and COBOL is more appropriate for operation.

The only areas of superiority for Assembler are in "Range of operation", "Execution time", and "Object program size". These indicate that Assembler would be most appropriate for operation and in this area dominates both COBOL and PL/I.

Figure 3

Programming Language Ratings*

	ALGOL	Assem	COBOL	FORT	PL/I	RPG
Language						
Programming time						
Problem oriented notation	3	1	4	3	4	2
Range of operations	1	5	3	1	4	1
Volume of coding	3	1	3	3	3	4
Ease of learning	3	1	3	4	3	1
Ease of coding	3	1	3	4	3	2
Ease of maintenance	3	1	3	3	3	3
Knowledge of machine	4	1	4	4	3	4
Interaction with OS	1	5	2	2	3	1
Ability to segment code	3	4	3	4	3	1
Implementation						
Efficiency						
Compilation time	3	2	3	3	3	3
Execution time	3	5	3	4	2	3
Object program size	3	5	3	4	2	4
Debugging						
Compilation diagnostics	2	4	3	3	4	2
Execution diagnostics	2	1	3	3	4	2
Compilation listings	2	4	3	3	4	2
Knowledge of machine	3	1	4	3	4	3
Interface with languages	1	4	3	4	3	1
Environment						
Evaluation of prior use	1	3	2	1	4	1
Support by ILR	1	2	2	3	4	2
Support by CCN	1	4	2	4	4	1
Support by vendor	1	3	4	4	4	2
Programmer availability	1	2	2	4	3	1
Convertibility	3	1	4	3	2	3
	51	61	69	74	76	48

* Rating Scale: 5 - Excellent
4 - Above average
3 - Average
2 - Below average
1 - Poor

Selections

The development of the software system for CIS will take place at CCN on the IBM 360/91. Due to the appropriateness of the language to the problem area, the relative ease of debugging and the available support for the language, PL/I was selected to be the primary language during system development. Assembler language was selected for use (sparingly) when needed to supplement the "Range of operations" and "Interaction with OS" facilities of PL/I.

During later phases of the project, it will probably prove desirable to rewrite the software system in Assembler language to take full advantage of its excellent ratings for operational efficiency.

SOFTWARE SYSTEM PROGRAMMING STRATEGIES

There are a number of overall programming strategies possible for the CIS software system. Duing Phase IIA, ILR staff examined several of these in varying depth.

Custom Programming

This involves acquiring or writing a program for each operation to be performed. Each select step, report step, etc. would be a separate program written for the purpose at hand. The read routines in Appendices A to F are examples of this approach. Custom programming is unquestionably a slow and expensive process and fails to recognize common elements of programs to perform similar functions. It would also lead to a proliferation of programs and procedures for their use.

Data Base Conversion

This involves defining a canonical form for each type of data base (bibliographic, numerical and full-text) and converting data bases to canonical form. Custom programs may be used for conversion. This approach is not unreasonable for bibliographic data bases (e.g. Nuclear Science Abstracts and CA Condensates) which are inherently similar in nature, but it becomes impractical when applied to numerical data bases (e.g. the 1970 Census and COMPUSTAT) or full-text data bases (e.g. the California Constitution and Webster's Dictionary) due to gross differences in the data bases. IBM's TEXT-PAC system is based on converting bibliographic data bases into a form that can be used by the system for Selective Dissemination and Retrospective searching and other functions.

Modular Programming

This involves identification of commonly used procedures (e.g. file read) which are dependent upon a particular data base. These procedures would be written for each data base, as necessary, and combined with program skeletons to form complete programs. The reference retrieval program presented in Part 1 of this Final Report is an example of this approach.

Language Development

This involves the development of a programing language (perhaps based on PL/I) to handle files and file descriptions in a straightforward manner. The use of such a language would not reduce the number of programs to be written (as opposed to Custom Programming) but would simplify the development of individual programs. This approach is used by the General Electric Company in their Integrated Data Store (IDS) system. IDS uses a non-procedural language to define file structures and a procedural language imbedded in a host language (e.g. COBOL or FORTRAN) for file processing.

Program Generator Development

This involves identification of commonly used procedures (e.g. sort, report) and the delevopment of program generators to generate programs for these procedures from specifications of files and operations desired. This approach is exemplified by IBM's Sort/Merge and Report Program Generator. These programs, however, do not have sufficient capabilities for the the purposes of this project.

Data Base Management System Acquisition

This involves identification and acquisition of an existing Data Base Management System of sufficient capability to serve the purposes of this project. Dozens of these systems are available from universities, computer manufacturers, independent software firms and others. To date, none has been proved suitable for CIS. This approach is quite reasonable, however, if a suitable system could be found.

Data Base Management System Development

This involves the development of a Data Base Management System with the capabilities needed. Such a system would be basically similar to existing systems of this nature, but would incorporate features necessary for the CIS software and would be oriented toward efficient use in the CIS environment. This is the approach that has been selected for the CIS project.

APPENDIX A

API FILE:

READ ROUTINE AND OTHER INFORMATION

(K. D. Reilly. December 1969)

BACKGROUND AND RECORD FORMAT

The API file, a sample of approximately 5000 records of which is available at UCLA, is produced by the American Petroleum Institute. The file is a strict "bibliographic" file consisting of the following type of information:

document number
author(s)
title
descriptors (major and minor)
bibliographic reference

The initial portion of each record consists of record-control information which will allow the programmer to compute the relative location of any field in the record. (Variable-length fields in API are a variable number of fixed-length segments.) Size is specified in terms of the numbers of these segments. The segments sizes used are 36 characters (for authors and descriptors) and 66 characters (for titles and the bibliographic reference). In addition to this the total record size is given in a special form: a number which must be multiplied by six (6) in order to get the total number of characters in the record.

Additional description of the record format can be found in API Information Retrieval System: Computer Manual prepared by Allen J. Humphrey, (New York, API, n.d.)

READ ROUTINE FOR API TAPE

The purpose of "read" routine for a tape is to "expose" some or all of the fields of each record so that such (field) data can be manipulated (searched, printed out, etc.) by programs (subroutines) used in conjunction with the "read" routine.

An example of a "read" routine in PL/1 is listed below. This particular routine was used in conjunction with a sub-routine, called OUTPUT (for printing out the fields of the record in a particular format, 3 x 5 cards) and thus many of the fields are DECLARED with the EXTERNAL attribute for inter-routine communication purposes. The important features to note are:

1. The (elementary) structure that can be used with API records. Identifiable in this structure are:

   RECLEN — which, when converted to numerical from character form and multiplied by 6, gives the record size. See the field RECL below.

   DOC_# — the document number assigned by the API staff. (The document number is a code that tells something about the nature of the data represented in the record (e.g., a literature item, etc.) and the sequence number for the particular year in which it was indexed. The programmer who wishes to make use of this data must refer to the API information in the Computer Manual.

   TTLL — a character which when converted to numeric form and multiplied by 66 gives the number of characters in the TITLE field. See TTL below.

   MAUT — a pair of characters which when converted to numeric form give the number of authors. See the NAUT field below.

   BIBREFLEN — a single character, which when converted to numeric form and multiplied by 66, gives the length of the bibliographic field (BIBREF). See BIBREFL field below.

Special consideration must be given to the two fields REC1 and REC2. If these character strings are concatenated and then converted to numeric form they give the number of descriptors in the record. See the MDES and NDES fields below.

2. This particular routine goes through the record from the front and extracts the fields in their natural order in the record. NOTE: it is not necessary to go through the record in this front-to-rear fashion. As we have remarked above, fields can be picked out in arbitrary order; to do this, the programmer must do some arithmetic over and above that displayed in the routine listing.

3. The routine stores individual authors in a PL/1 array called AUT. It puts the individual descriptors into the array DES and then puts the Primary Descriptor, which has a "*" in position 34, into SUB(1) and all other descriptors which are access points to the document--these have a "P" for Permuted in position 34--into subsequent positions in SUB. Descriptors which are not access points, but merely supply the user with further pertinent information about the document have no "P" in position 34. (SUB contains the major descriptors then, and DES contains all of the descriptors.) The Auxiliary variable, NSUB, is formed, providing the OUTPUT subroutine with the number of elements in SUB.

ACCESS TO THE API TAPE (UCLA USERS).

The API records that ILR possesses are a sample from the total API file, they are contained on a single reel of tape (APIØØ1) at the Campus Computing Network in Bin #21. A standard request form will make the tape available to the user as he needs it. See CCN personnel for details.

In order to access the tape, the following DD cards can be used:*

```
//GO.TAPEIN DD UNIT=24ØØ,
//             DISP=OLD,
//             VOLUME=SER=APIØØ1,
//             LABEL=(,BLP,,IN),
//             DCB=(RECFM=U,BLKSIZE=45ØØ)
```

The tape is 9-track and has a density of 800 BPI.

REMARKS

The "read" routine was developed while CCN had its IBM 360/75, but should run without difficulty on the present machine (IBM 360/91). The development effort took less than usual (3-4 runs) due to the fact that Alan Humphrey of ILR-Berkeley, formerly with the American Petroleum Institute, was able to provide us with helpful information, the essential (written) parts of which can be found filed with the ILR copy of the Computer Manual.

The API file was used, along with MARC I, in some of the early experimenting with what might be called the RR approach

*The TAPEIN used in GO.TAPEIN must be defined in the PL/1 program, also, with the file attribute.

("read" routine) to programming an MM (multiple-master-file) "library" of data bases. The RR approach makes use of common sub-routines for search and output along with a "read" routine which is specific to each file, in the attempt to handle a variety of different master file systems (as is planned for a Center for Information Services.) The few programs run against the API file, available in the tape documentation file of the ILR, center around the outputting of "hit" records in a 3 x 5 card format. (This output routine presented as Appendix A, has been put to use not only with the API file but also with the MARC I, COMPENDEX, and ERIC files and is currently under further development.) At the time when efforts on API ceased, a sample of records had been placed on direct access for further experimenting. Needless to say, much more work can still be done profitably with the sample API records (e.g., investigation of "roles" and "links", etc.). Since the API file is a likely candidate for the eventual Center for Information Services data library, experience with it takes on added importance.

LISTING OF READ-ROUTINE*

```
LIST2:   PROC OPTIONS(MAIN);
         DCL AUT(10) CHAR(36) EXTERNAL,
             OUTPUT ENTRY,
             DES(100) CHAR(36),
             MDES CHAR(3),
             NDES BINARY,
             NAUT BINARY FIXED EXTERNAL,
             RECL BINARY FIXED EXTERNAL,
             NSUB BINARY FIXED EXTERNAL,
             TTL BINARY FIXED EXTERNAL,
             BIBREFL BINARY FIXED EXTERNAL,
             BGN BINARY,
             SUB(10) CHAR(36) EXTERNAL,
             BIBREF CHAR(198) VAR EXTERNAL,
             TITLE CHAR(131) VAR EXTERNAL,
             1 RECORD,
               2 REC1 CHAR(1),
               2 RECLEN CHAR(3),
               2 DOC_# CHAR(8),
               2 TTLL CHAR(1),
               2 MAUT CHAR(2),
               2 BIBREFLEN CHAR(1),
               2 REC2 CHAR(2),
               2 REC CHAR(4500),
            TAPEIN FILE RECORD;
ON ENDFILE(TAPEIN) GO TO END:
PUT PAGE;

                    /* API TAPE */
         OPEN FILE(SYSPRINT) PAGESIZE(58);
READ_IN:
         REC = (4500) ' ';
         L = 2;
         AUT = ' ';
         NAUT = 0;
         RECL = 0;
         TTL = 0;
         BIBREFL = 0;
         SUB = ' ';
         BIBREF = ' ';
         TITLE = ' ';
         READ FILE(TAPEIN) INTO (RECORD);
         MDES = REC1 || REC2;
         NDES = MDES;
         RECL = RECLEN;
```

LISTING OF READ-ROUTINE* (Continued)

```
            TTL = TTLL;
            TTL = 66*TTL
            NAUT = MAUT;
            BIBREFL = BIBREFLEN;
            BIBREFL = 66*BIBREFL;
            TITLE = SUBSTR(REC,1,TTL);
            BGN = TTL + 1;
            DO I = 1 TO NAUT;
            AUT(I) = SUBSTR(REC,BGN,36);
            BGN = BGN +36;
            END;
            BIBREF = SUBSTR(REC,BGN,BIBREFL);
            BGN = BGN + BIBREFL;
            DO J = 1 TO NDES;
            DES(J) = SUBSTR(REC,BGN,36);
            BGN = BGN + 36;
            IF SUBSTR(DES(J),34,1) = '*' THEN DO;
            SUB(1) = SUBSTR(DES(J),1,33);
            NSUB = NSUB + 1;
            END;
            IF SUBSTR(DES(J),34,1) = 'P' THEN DO;
            SUB(L) = SUBSTR(DES(J),1,33);
            L = L + 1;
            NSUB = NSUB + 1;
            END;
            END;
            PUT EDIT(AUT) (SKIP,A);
            PUT EDIT(SUB) (SKIP,A);
            PUT EDIT(TITLE) (SKIP,A);
            PUT EDIT(BIBREF) (SKIP,A);
          PUT PAGE;
          CALL OUTPUT:
          GO TO READ_IN;
      END; END LIST2;
```

* The listing of OUTPUT is not given here. (It was a pre-compiled (object) routine in the present run.) Its output also is not listed.

OUTPUT (FROM MAIN PROGRAM)*
(one record -- unnumbered)

GREAT CANADIAN OIL SANDS LTD
BRECKENRIDGE R M
} Authors

TAR SAND
ATHABASCA AREA
} Major Subjects

PROGRESS REPORT...ATHABASCA TAR SANDS } Title
WORLD PETROL V37 N.13.51,54-56 %DEC 1966< } Bibliographic Reference

*The descriptors in the array DES were (unfortunately) not asked for in the program and accordingly do not appear.

APPENDIX B

U. S. CENSUS OF HOUSEHOLDS (1960): 1/1000 SAMPLE. "READ" ROUTINE AND OTHER INFORMATION

(K. D. Reilly. December 1969)

BACKGROUND

The 1960 Census of Households (1/1000 Sample of Households) file is available in its entirety at UCLA. It is at the Campus Computing Network and is available to any user; it is obtained by asking for tape CCN110. The Census of Households is only a very small part of the overall outpouring of data from the U. S. Department of Commerce, Bureau of the Census. Essentially, what the Census takers do every ten years is to ask every person that they can find certain basic questions; in order to comply with the laws of confidentiality, it is summary tabulations of these data that form the bulk of the "Census" data. In addition to the basic data collection there is the sample effort. In 1960, some 20% of households were asked one set of questions; a different 5% were asked a somewhat (though not extensively) different set of questions. These households samples, for obvious reasons, are referred to as the 20% and 5% samples. For those questions which are asked of both 5% and 20% of households there is a 25% sample. (Note that the percentages changed in 1970; some new questions were added and old ones deleted, but the file remains substantially the same in content for the two periods.) The 1/1000 sample is a stratified sample extracted from the 25% sample; therefore, it contains both 5% and 20% records. The contents of the file can be ascertained in Figure 1, which provides the data item names, and the status of each item (whether asked of everyone, a 5% sample, etc.) for both 1960 and 1970. Besides what is exhibited below, each record has a serial number, a region indicator, etc., as can be seen in the F /1 structure of Figure 2.*

*From this point on all references are to the 1960 file and may not be accurate for the 1970 file.

FIGURE 1

DATA ITEMS OF U. S. CENSUS

Population Items	Complete-Count or Sample Percentage	
	1960	1970
Relationship to head of household	100	100
Color or race	100	100
Age (month and year of birth)	100	100
Sex	100	100
Marital status	100	100
State or country of birth	25	20
Years of school completed	25	20
Number of children ever born	25	20
Activity 5 years ago	-	20
Employment status	25	20
Hours worked last week	25	20
Weeks worked last year	25	20
Last year in which worked	25	20
Occupation, industry, and class of worker	25	20
Income last year:		
Wage and salary income	25	20
Self-employment income	25	20
Other income	25	20
Country of birth of parents	25	15
Mother tongue	25	15
Year moved into this house	25	15
Place of residence 5 years ago	25	15
School or college enrollment (public or private)	25	15
Veteran status	25	15
Place of work	25	15
Means of transportation to work	25	15
Occupation-Industry 5 years ago	-	5
Citizenship	-	5
Year of immigration	-	5
Marital history	25	5
Vocational training complete	-	5
Presence and duration of disability	-	5

FIGURE 1 (Continued)

Housing Items	Complete-Count or Sample Percentage	
	1960	1970
Number of units at this address	-	100
Telephone	25	100
Access to unit	100	100
Kitchen or cooking facilities	100	100
Complete kitchen facilities	-	100
Condition of housing unit	100	-
Rooms	100	100
Water supply	100	100
Flush toilet	100	100
Bathtub or shower	100	100
Basement	20	100
Tenure	100	100
Commercial establishment on property	100	100
Value	100	100
Contract rent	100	100
Vacancy status	100	100
Months vacant	25	100
Components of gross rent	25	20
Heating equipment	25	20
Year structure build	25	20
Number of units in structure	20	20
Whether a trailer	25	20
Farm residence (acreage and sales of farm products)	25	20
Land used for farming	25	-
Source of water	20	15
Sewage disposal	20	15
Bathrooms	20	15
Air conditioning	5	15
Automobiles	20	15
Stories, elevator in structure	20	5
Fuel--heating, cooking, water heating	5	5
Bedrooms	5	5
Second home	-	5
Clothes washing machine	5	5
Clothes dryer	5	5
Dishwasher	-	5
Home food freezer	5	5
Television	5	5
Radio	5	5

Besides the basic documentation provided by Census Bureau the tape documentation file contains a Master's thesis by K. Pearson, PROVIDING FOR MACHINE-READABLE STATISTICAL DATA SETS IN UNIVERSITY RESEARCH LIBRARIES, published as Report SP-3155/000/00 by Systems Development Corporation. This work covers a number of issues of much broader scope than what is discussed in this report; its main interest here is that it contains an example program using the U. S. Census of Households (1/1000 Sample) file. A basic data quality control routine** checks "common universe" totals.

RECORD FORMAT

Each item or field in the U. S. Census of Households (1/1000 Sample) file is fixed in length. Most of the fields are single-character (0-9; V or X) codes, although a few, e.g., wage and salary information, are multiple character. The record size is a total of 120 characters. The distinction between the 5% and 20% record is done by simple check of the content of a single field located at the 113th position in the record. This field is blank for the 20% sample.

READ ROUTINE

The purpose of a "read" routine for a file is to "expose" some or all of the fields of each record so that such (field) data can be manipulated (scanned, compared to other data, printed out, etc.) in subroutines used in conjunction with the "read" routine. The concept of "read" routine for a fixed-field record, in general, consists merely of devising the structure(s) necessary to provide 'names' to the field data, and thus is little more than a section of code that can be placed away on a disk and called (say by the PL/1 preprocessor command, INCLUDE) in any particular program for which the structure(s) is (are) needed. This is the approach discussed in the cited work of K. Pearson. Pearson's "included" text also has a means of separating the 5% and 20% samples and of providing for conversion of certain character-string fields to numerical form. No provision comparable to Pearson's conversion subroutine is presented in this report. The approach taken to separate the two samples here makes use of PL/1 pointer variables to "overlay" a portion of core (in the buffer region) with two different structures, one for the 5% records and the other for the 20% record. These structures at the top level (see Figure 2) are:

**A listing of this routine, written by A. deBoer of the CIS staff, is in the ILR tape documentation file. The results obtained from running this program and a complete paper listing of the entire file are in-house at CCN.

FIGURE 2

STRUCTURES FOR 5% AND 20% SAMPLE RECORDS OF

U. S. CENSUS OF HOUSEHOLDS FILE

(1/1000 Sample), 1960

```
DCL  1  FIVE                          BASED (PT),
          2 SERIALNR                        CHAR (5),
          2 REGION                          CHAR (1),
          2 PLACESIZE                       CHAR (1),
          2 SMSA                            CHAR (1),
          2 URBANSIZE                       CHAR (1),
          2 AGE                             CHAR (2),   /*  10 - 11  */
          2 BIRTHQRTR                       CHAR (1),
          2 SEX_MRG_QRTR                    CHAR (1),
          2 YR_1ST_MGR                      CHAR (2),
          2 MARITALSTATUS                   CHAR (1),
          2 RELATION                        CHAR (1),
          2 FAMILYNR                        CHAR (1),
          2 FAMILYTYPE                      CHAR (1),
          2 RACE                            CHAR (1),   /*  20  */
          2 NATIVITY                        CHAR (1),
          2 PARENTORIGIN                    CHAR (2),
          2 RECODE_PAR_ORIGIN               CHAR (1),
          2 POB_FOREIGN                     CHAR (2),
          2 POB_RECODE                      CHAR (1),
          2 MOTHERTONGUE                    CHAR (2),
          2 MOMTONGUERECODE                 CHAR (1),   /*  30  */
          2 YEARMOVED                       CHAR (1),
          2 SAME_SMSA_1955                  CHAR (1),
          2 METRO1955                       CHAR (1),
          2 HIGHESTGRADE                    CHAR (1),
          2 SCHOOL                          CHAR (1),
          2 EMPLOYSTATUS                    CHAR (1),
          2 HOURS                           CHAR (1),
          2 OCCUPATION                      CHAR (3),   /*  41 - 43  */
          2 INDUSTRY                        CHAR (3),
          2 WORKERCLASS                     CHAR (1),
          2 WORKPLACE                       CHAR (1),
          2 TRANSPORT                       CHAR (1),
          2 WEEKSWORKED59                   CHAR (1),
          2 CHILDRENBORN                    CHAR (1),
          2 TOTAL59EARN                     CHAR (1),
          2 WAGES59                         CHAR (3),   /*  50 - 52  */
          2 SELF59                          CHAR (3),
          2 OTHER59                         CHAR (3),
          2 TOTAL59INCOME                   CHAR (3),   /*  59 - 61  */
```

FIGURE 2 (Continued)

```
2 SOCIOSTATUS                CHAR (1),
2 SOCIOCONSTANT              CHAR (1),
2 HOUSEHOLDTYPE              CHAR (1),
2 HOUSEHOLDTOTAL             CHAR (1),
2 NR_NONRELATIVE             CHAR (1),
2 NR_IN_FAMILY               CHAR (1),
2 NR_REL_UNDER18             CHAR (1),
2 AGE_UNDER18                CHAR (1),
2 NR_65                      CHAR (1),   /*  70  */
2 NR_LABOR                   CHAR (1),
2 NR_EMPLOYED                CHAR (1),
2 NR_UNEMPLOYED              CHAR (1),
2 FAM20LABORSTAT             CHAR (1),
2 RELATEDEARNERS             CHAR (1),
2 FAMILY59TOTALING           CHAR (3),
2 FAM59INC                   CHAR (1),
2 OWNCHILD_UNDER3            CHAR (1),   /*  80  */
2 OWNCHILD_3TO4              CHAR (1),
2 OWNCHILD_5                 CHAR (1),
2 OWNCHILD_6                 CHAR (1),
2 OWNCHILD_6TO11             CHAR (1),
2 OWNCHILD_12TO17            CHAR (1),
2 OWNSINGLE_UNDER18          CHAR (1),
2 OWNSINGLE_18PLUS           CHAR (1),
2 OWN_GRADE_1TO8             CHAR (1),
2 AGE_YOUNGEST               CHAR (1),
2 AP_AGE                     CHAR (2),   /*  90 - 91  */
2 AP_SEX_MRGSTATUS           CHAR (1),
2 AP_RACE                    CHAR (1),
2 AP_HIGHESTGRADE            CHAR (1),
2 AP_EMPLOYSTATUS            CHAR (1),
2 AP_HRSLASTWK               CHAR (1),
2 AP_OCCUPATION              CHAR (1),
2 AP_CHILDBORN               CHAR (1),
2 AP_TOTAL59INC              CHAR (1),
2 MOM_HIGHESTGRADE           CHAR (1),   /*  100  */
2 MOM_LABORSTATUS            CHAR (1),
2 TENURE                     CHAR (1),
2 KITCHENPHONE               CHAR (1),
2 ROOMS                      CHAR (1),
2 PERSONS_PER_ROOM           CHAR (1),
2 BATH                       CHAR (1),
2 WATER                      CHAR (1),
2 TOILET                     CHAR (1),
2 BATHING                    CHAR (1),
2 BUILT                      CHAR (1),   /*  110  */
2 HEATING                    CHAR (1),
2 AUTOS                      CHAR (1),
```

FIGURE 2 (Continued)

```
          2 CONTRACT_RENT                          CHAR (1),
          2 GROSS_RENT                             CHAR (1),
          2 VALUE_RATIO                            CHAR (1),
          2 BEDROOMS                               CHAR (1),
          2 FUEL                                   CHAR (1),
          2 WASHERDRYER                            CHAR (1),
          2 RADIOTV                                CHAR (1),
          2 AIRCON_FREEZER                         CHAR (1);
DCL   1   TWENTY                        BASED (PT),
          2 FILLER1                                CHAR (26),
          2 POB_US                                 CHAR (1),
          2 FILLER2                                CHAR (9),
          2 YEARLASTWORKED                         CHAR (1),
          2 FILLER3                                CHAR (10),
          2 VETERAN                                CHAR (1),
          2 FILLER4                                CHAR (15),
          2 QUARTERSTYPE                           CHAR (1),
          2 FILLER5                                CHAR (14),
          2 TOTAL59INC_NOTFAM                      CHAR (1),
          2 FILLER6                                CHAR (16),
          2 AP_YRLASTWORKED                        CHAR (1),
          2 FILLER7                                CHAR (1),
          2 AP_VETERAN                             CHAR (1),
          2 FILLER8                                CHAR (15),
          2 PROPERTYVALUE                          CHAR (1),
          2 RENT_RATIO                             CHAR (1),
          2 BATHROOMS                              CHAR (1),
          2 BLANKFOR20PC                           CHAR (1),
          2 STRUCTURETYPE                          CHAR (1),
/*        2 STORIES                                CHAR (1),
          2 SEWAGE                                 CHAR (1),
          2 BASEMENT                               CHAR (1);
```

Note: The structure, TWENTY, has not been utilized other than for simple listings and there may be some inaccurate and/or misleading name usages in the latter portions, e.g., around STORIES where some sharing of fields may occur. Refer to the original documentation for clarification on this matter.

```
FIVE   BASED (PT),
TWENTY BASED (PT).
```

Note that all the field names of both structures are available to the programmer to manipulate. Thus POB_US and POB_RECODE are one and the same piece of data, but represent different concepts depending on whether the particular record is from the 5% or the 20% sample. One PL/1 READ command suffices to "fill" both structures at the same time; the statement appears as:

```
READ FILE(DSINPUT) SET (PT);
```

SPECIAL COMMENTS

There are possible further developments (currently under investigation) of the "read" routine concept as it might be applied to the Household Sample data file. In further development of the idea it must be taken into account that individual records in the file often cannot be used alone. This is because the records of all the members of a single household follow one another in sequence. Since the serial number changes when a new household arises in the file, a "read" routine can be constructed so that for certain studies the second, third, etc., records for a given family can be "thrown" away (e.g., a study in which the extent of association between the education level of the head of the household and the number of automobiles is sought). Of course, this is but one possible "logical" division of family records and on other occasions, it may be that the wife or the children or members of the "sub-family" are under investigation.

Still another vagary in these data that the ideal "read" routine must cope with is that of "shared" field locations. The field HOURS gives the number of hours worked in the previous week (previous to the Census taker's call upon the household) for working individuals, whereas it gives the last year worked for retired and unemployed individuals. The "tip-off" for which meaning this field takes on in a particular context is the value of the field EMPLOYSTATUS, the field just in front of HOURS in the structure. Note in passing that there are still other peculiarities that must be taken into account, e.g., the special treatment needed for handling the automobile ownership field. It would appear that incorporation of these complexities into a "read" routine is not impossible by any means but is not necessarily a straightforward matter.

ACCESS TO THE CENSUS TAPE (UCLA User's)

The 1960 Census of Households (1/1000 Sample) tape is available in its entirety on a single reel of 800 BPI, 9-track

tape at CCN. It is not in the so-called User's Bin; it is on the CCN's own tape rack and may be retrieved from there by using the customary request sheet and merely asking for tape #CCN110. See CCN personnel for details.

The Job Control Language for the tape is as follows:*

```
//GO.DSINPUT DD VOL=SER=CCN11Ø,
//              DISP=(OLD,KEEP),
//              DSN=CENSUS6Ø,
//              LABEL=(1,SL,,IN),
//              DCB=(RECFM=FB,LRECL=12Ø,BLKSIZE=12ØØØ),
//              UNIT=24ØØ
```

REMARKS ON THE USE OF THE FILE

Census data are very popular. They form the basis for a multitude of research projects in their own right. A paragraph of review on some of these projects is found on page 47 of the report by K. Reilly, "Nature of Typical Data Bases", Part 5 of the final report on "Mechanized Information Services in the University Library - Phase I: Planning", published by ILR.

Census data formed the basis for a large-scale study in New Haven, Connecticut. A number (so far eleven) of documents published by the U. S. Bureau of Census describe this study; collectively these documents bear the name CENSUS USE STUDY and individual reports include: General Description, Computer Mapping, Data Tabulation Activities, the DIME Geocoding System, Data Interests of Local Agencies, Family Health Survey, Health Information System, Data Uses in Health Planning, Data Uses in Urban Planning, Data Uses in School Administration, Area Travel Study. In addition to these activities, the New Haven study led to the development of "computer programming packages" for computer graphing and for merging of local data (differently organized data, i.e., different keys or serial numbers, etc.) with Census data to form a more comprehensive data base. These are, by name: DIME, a program package for creating the file that relates local data file keys or serial numbers to geographical keys or codes (e.g., Census tracts); GRIDS, a program package for use on small-scale computers for mapping of data, with "shading" capabilities, etc.; ADMATCH, a program package for assigning geographical codes to local records. Continuation of exploratory studies on the use of Census data is underway in Los Angeles. Part of this work is

*The PL/1 program must have a DECLARE command in it for a tape (Input) with the name DSINPUT.

being done here at UCLA. The "partners" are ILR (informally), the Engineering Department, and SCRIS (Southern California Regional Information Study).

The work to date that has been performed here at ILR has been primarily exploratory. As mentioned above, the file was reviewed in the study of Reilly and was involved in the exercise portion of the Pearson master's thesis. Pearson searched the file for librarians and librarian aids and printed out in tabular form some information (e.g., age breakdown, salary, portion of the country, etc.) about them. In addition to this, a portion of the records were placed on direct access, in part for purposes of random sampling from the file. There was a beginning of a study utilizing the IBM SSP (Scientific Subroutine Package) statistical programs in relation to the file within the scope of the "Center for Information Services" project. The first program in this effort was a simple Chi-square "association" test for education vs. number of automobiles. Information on these studies is availalbe in the ILR tape documentation file.

APPENDIX C

READ ROUTINE FOR THE CBB FILE

(G. Silva. August 1969)

I. Identification

Title	READMT	
User Group	Institute of Library Research	
Category	CIS Text Processing	
Source Language	PL/1	
Machine Configuration	IBM 360/91	
Space Requirements	Step Region:	Core Used:
	PL/1 158K LKED 154K GO 150K	72K
Author	G. Silva	
Date	August 1969	

READ ROUTINE FOR THE CBB FILE

II. Purpose

1. To read magnetic tape records of the Communications of Behavioral Biology file. These records contain bibliographical information described under "Record Format" (see section IV).
2. To identify each field within a record.

READ ROUTINE FOR THE CBB FILE

III. Tape Specifications

1. 9 Track Tape
2. 800 BPI
3. Standard Label
4. One File
5. Volume Serial Number is ILR 005
6. The Data Set name is CBBCCF
7. Record Format is Variable (V)
8. The Block Size is 7200
9. The Code is BCDIC

The DD Card used to read the tape is:

```
//GO.CBBR   DD DSNAME=CBBCCF,UNIT=TAPE9,VOLUME=SER=ILR005,
//    DCB=(RECFM=V,BLKSIZE=7200),DISP=OLD,LABEL=(3,SL,,IN)
```

Note:

The subparameter IN (fourth subparameter of the LABEL parameter), tells the system that this tape is to be used as input only and eliminates the operator/computer dialogue on write rings.

READ ROUTINE FOR THE CBB FILE

IV. File Description

The file contains variable length records.

Record Format

Records consist of fixed header information followed by a variable length field.

Header Information:	Field Length	PL/1 Attributes
Citation Number	5 bytes	FIXED(9)
Something	1 byte	BIT(8), or CHAR(1)
Number of Journal Title Words	2 bytes	BIT(16)
Year Published	2 bytes	FIXED(3)
Type of Publication	2 bytes	BIT(16)
Date Merged into Master File	6 bytes	CHAR(6)
Number of Descriptors	2 bytes	BIT(16)
Number of Authors	2 bytes	BIT(16)
Language Code	2 bytes	FIXED(3)
Number of Title Words	2 bytes	BIT(16)
Number of Page Words	2 bytes	BIT(16)
Number of Date-Publ Words	2 bytes	BIT(16)
Number of Vernacular Words	2 bytes	BIT(16)
Something	4 bytes	CHAR(4)
Number of Abstracts	2 bytes	BIT(16)
File Source Code	1 byte	FIXED(1)
Author Type Code	1 byte	FIXED(1)
Combined Citation Search Code	1 byte	FIXED(1)
Journal Title Code	4 bytes	CHAR(4)
Starting Page	6 bytes	CHAR(6)
Something	1 byte	CHAR(1)
Number of Address Words	2 bytes	BIT(16)
Number of Abstract Words	2 bytes	BIT(16)
Reserved	2 bytes	CHAR(2)

READ ROUTINE FOR THE CBB FILE

IV. File Description (Continued)

Variable Portion	Field Length	PL/1 Attributes
Authors:		
Reserved	4 bytes	CHAR(4)
No. of Print Words	1 byte	FIXED(1)
No. of Sort Words	1 byte	FIXED(1)
Sort Author	36 bytes	CHAR(36)
Print Author	36 bytes	CHAR(36)
	78 bytes	
Title	6 x No. of title words	CHAR(600) VAR
Journal Title Abbreviation	6 x No. of jour. title wds	CHAR(60) VAR
Pages	6 x No. of page words	CHAR(30) VAR
Publication Date	6 x No. of date pub. wds	CHAR(18) VAR
Vernacular Title	6 x No. of vern. words	CHAR(480) VAR
Address	6 x No. of addr. words	CHAR(120) VAR
Abstract	6 x (No. of abstr) x (No. of abstr wds)	CHAR(3000) VAR
Descriptor Terms	48 x No. of descript.	CHAR(1000) VAR

READ ROUTINE FOR THE CBB FILE

V. Checkout

1. Timing

The program processed 100 logical records with the following results:

Step Name	Step CPU Seconds
PL1L	2.82s
LKED	.55s
GO	2.24s
Average processing time:	22.4 millisecs/log.rec

Note that if printing of data elements is bypassed, the processing time is .26s/100 log. recs, or 2.6 millisecs/log.rec. This is of the order of 2 mins/50,000records.

2. Error Checking

None

3. Cost

Opening the CBB file was closely interwoven with the development and testing of a much larger text processing system. I am therefore unable to separate the cost of opening the file from the rest of the processing.

READ ROUTINE FOR THE CBB FILE

VI. Program Description

1. Preprocessor variables:

REC structure name for header information	CBB_RECORD
REC_SIZE length of variable part of record	7000 Bytes
LABSTR length of abstract field	3000 Bytes
MAX_REC_NO the maximum number of records to be processed in any run	100

2. Processor variables (in order of occurrence in deck)

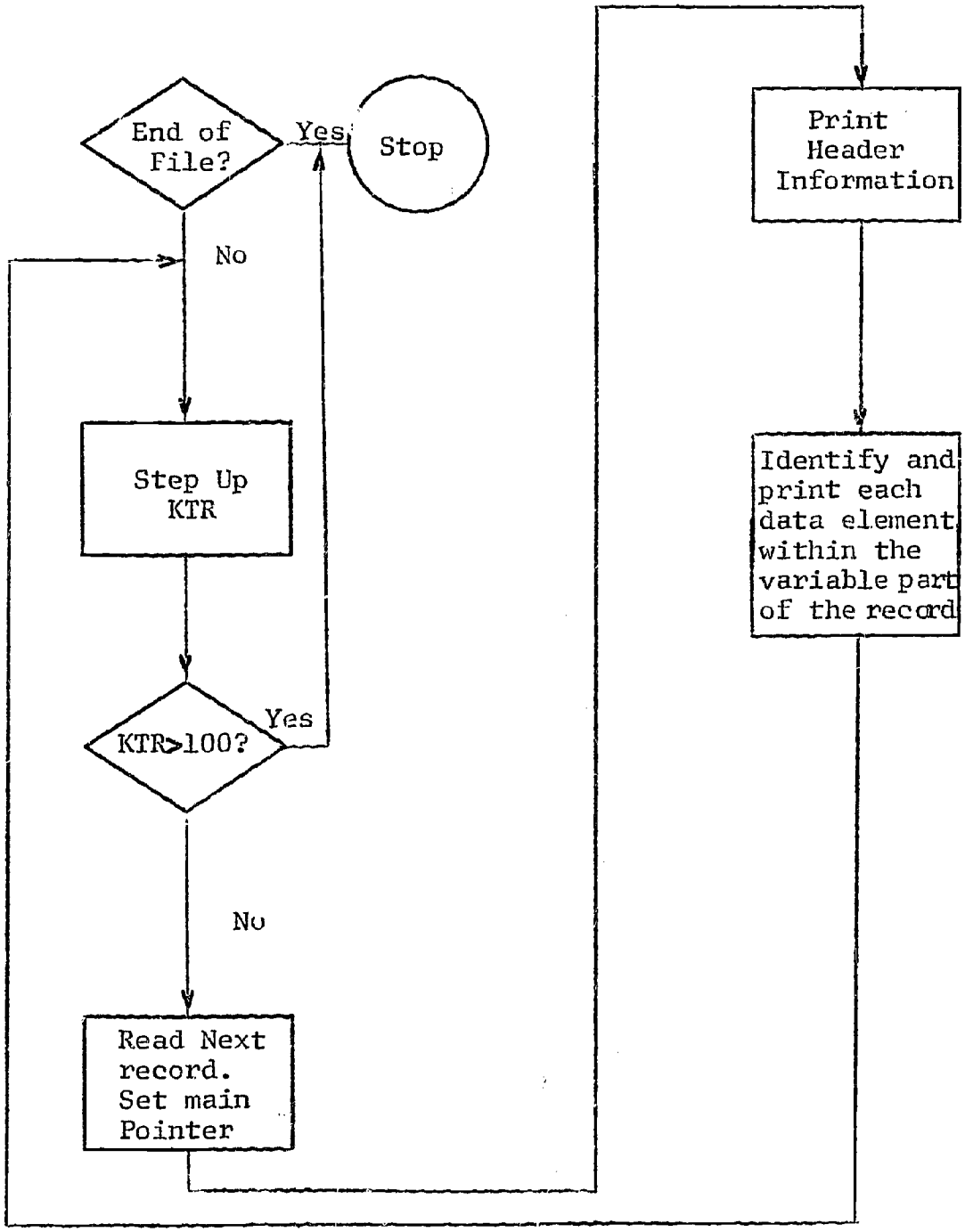
CBB_RECORD structure name for entire record

The variable names in the structure CBB_RECORD are self-explanatory.

KTR counts the number of records processed.

READ ROUTINE FOR THE CBB FILE

3. Flowchart



```
//NRB409MT  JOB  '20,2500',SILVA
//  EXEC PL1LFCLG,PARM='M,SM=(2,72,1),ST',PG=200,RG=150K
//PL1L.SYSIN DD *
  TESTCBB:
            PROCEDURE OPTIONS(MAIN);
        % DCL ITEST FIXED;
        % ITEST=1;
        % ITEST=0;
        % DCL REC CHAR;
        % REC= 'CBB_RECORD';
        % DCL REC_SIZE FIXED;
        % REC_SIZE = 7000;
        % DCL LABSTR FIXED;
        % LABSTR =3000;
        % DCL  MAX_REC_NO FIXED;
        % MAX_REC_NO =100;
    DCL 1 REC BASED(MAIN_P),
          2 HEADER,
            3 CITATION_NO FIXED(9),
            3 DISREGARD2 BIT(8),
            3 NO_J_TITL_WDS  BIT(16),
            3 YR_PUBL FIXED(3),
            3 TYPE_PUBL BIT(16),
            3 MERGE_DATE CHAR(6),
            3 NO_DESCRIPT BIT(16),
            3 NO_AUTHORS BIT(16),
            3 LANG_CODE FIXED(3),
            3 NO_TITL_WDS BIT(16),
            3 NO_PG_WDS BIT(16),
            3 NO_DATE_PUBL_WDS BIT(16),
            3 NO_VERN_WDS BIT(16),
            3 DISREGARD3 CHAR(4),
            3 NO_ABSTR BIT(16),
            3 FILE_SOURCE_CODE FIXED(1),
            3 AUTHOR_TYPE_CODE FIXED(1),
            3 COMB_CIT_SEARCH_CODE FIXED(1),
            3 J_TITL_CODE CHAR(4),
            3 STARTING_PG CHAR(6),
            3 DISREGARD4 CHAR(1),
            3 NO_ADDR_WDS BIT(16),
            3 NO_ABSTR_WDS BIT(16),
            3 DISREGARD5 CHAR(2),
          2 REMAINDER CHAR(REC_SIZE);
    DCL 1 AUTHORS BASED(P),
          2 RESVD CHAR (4),
          2 NO_PRINT_WDS FIXED(1),
          2 NO_SORT_WDS FIXED(1),
          2 SORT_AUTHOR CHAR(36),
          2 PRINT_AUTHOR CHAR(36);
    DCL TITLE CHAR(600) VAR;
    DCL J_TITL_ABBR CHAR(60) VAR;
    DCL PAGES CHAR(30) VAR;
    DCL PUBL_DATE CHAR(18) VAR;
    DCL VERNACULAR_TITL CHAR(480) VAR;
    DCL ADDRESS CHAR(120) VAR;
    DCL ABSTRACT CHAR(LABSTR) VAR;
    DCL DESCR_TERMS CHAR(1000) VAR;
    DCL  PROXY FIXED BIN(31) BASED(PROXY_PTR);
       DCL I FIXED BIN;
```

```
        ON ENDFILE(CBBR) GO TO FINIS;
        PROXY_PTR=ADDR(P);
        KTR=0;
NEXTREC:
        KTR=KTR+1;
        IF KTR> MAX_REC_NO THEN GO TO FINIS;
        READ FILE (CBBR) SET (MAIN_P);
        % IF ITEST=0 % THEN % GO TO IT20;
/*  PRINT OUT HEADER INFORMATION  */
        PUT PAGE;
        PUT SKIP LIST('CITATION_NO',CITATION_NO);
        PUT SKIP LIST('DISREGARD2',DISREGARD2);
        PUT SKIP LIST('NC_J_TITL_WDS',NO_J_TITL_WDS);
        PUT SKIP LIST('YR_PUBL',YR_PUBL);
        PUT SKIP LIST('TYPE_PUBL',TYPE_PUBL);
        PUT SKIP LIST('MERGE_DATE',MERGE_DATE);
        PUT SKIP LIST('NO_DESCRIPT',NO_DESCRIPT);
        PUT SKIP LIST('NO_AUTHORS',NO_AUTHORS);
        PUT SKIP LIST('LANG_CODE',LANG_CODE);
        PUT SKIP LIST('NC_TITL_WDS',NO_TITL_WDS);
        PUT SKIP LIST('NO_PG_WDS',NO_PG_WDS);
        PUT SKIP LIST('NC_DATE_PUBL_WDS',NO_DATE_PUBL_WDS);
        PUT SKIP LIST('NO_VERN_WDS',NO_VERN_WDS);
        PUT SKIP LIST('DISREGARD3',DISREGARD3);
        PUT SKIP LIST('NO_ABSTR',NO_ABSTR);
        PUT SKIP LIST('FILE_SOURCE_CODE',FILE_SOURCE_CODE);
        PUT SKIP LIST('AUTHOR_TYPE_CODE',AUTHOR_TYPE_CODE);
        PUT SKIP LIST('COMB_CIT_SEARCH_CODE',COMB_CIT_SEARCH_CODE);
        PUT SKIP LIST('J_TITL_CODE',J_TITL_CODE);
        PUT SKIP LIST('STARTING_PG',STARTING_PG);
        PUT SKIP LIST('DISREGARD4',DISREGARD4);
        PUT SKIP LIST('NC_ADDR_WDS',NO_ADDR_WDS);
        PUT SKIP LIST('NO_ABSTR_WDS',NO_ABSTR_WDS);
        PUT SKIP LIST('DISREGARD5',DISREGARD5);
% IT20:;
        P = ADDR(REMAINDER);
        DO I=1 TO NO_AUTHORS;
        % IF ITEST=0 % THEN % GO TO IT30;
        PUT SKIP LIST ('AUTHORS',AUTHORS);
% IT30:;
        PROXY=PROXY+78;
        END;
        I = NO_AUTHORS*78 +1;
        NO_TITL_CHAR = 6*NO_TITL_WDS;
        TITLE = SUBSTR( REMAINDER,I,NO_TITL_CHAR);
        I = I + NO_TITL_CHAR;
        NO_J_TITL_CHAR = 6*NO_J_TITL_WDS;
        J_TITL_ABBR = SUBSTR( REMAINDER,I,NO_J_TITL_CHAR);
        I = I + NO_J_TITL_CHAR ;
        NO_PG_CHAR = 6* NC_PG_WDS;
        PAGES = SUBSTR ( REMAINDER,I, NO_PG_CHAR);
        I = I + NC_PG_CHAR;
        NO_DATE_PUBL_CHAR = 6* NO_DATE_PUBL_WDS;
        PUBL_DATE = SUBSTR( REMAINDER,I, NO_DATE_PUBL_CHAR);
        I = I + NO_DATE_PUBL_CHAR;
        NO_VERN_TITL_CHAR = 6* NC_VERN_WDS;
        VERNACULAR_TITL = SUBSTR ( REMAINDER ,I, NO_VERN_TITL_CHAR);
        I = I + NO_VERN_TITL_CHAR;
        NO_ADDR_CHAR = 6* NO_ADDR_WDS;
        ADDRESS = SUBSTR ( REMAINDER,I, NC_ADDR_CHAR);
```

```
          I = I + NO_ADDR_CHAR;
          NO_ABSTR_CHAR = NO_ABSTR* NO_ABSTR_WDS * 6;
          ABSTRACT = SUBSTR ( REMAINDER , I, NO_ABSTR_CHAR);
          I = I + NO_ABSTR_CHAR;
          NO_DESCR_TERMS_CHAR = NO_DESCRIPT * 48;
          DESCR_TERMS = SUBSTR ( REMAINDER,I, NO_DESCR_TERMS_CHAR);
          % IF ITEST=0 % THEN % GO TO IT50;
          PUT SKIP LIST( 'TITLE',TITLE);
          PUT SKIP LIST( 'J_TITL_ABBR',J_TITL_ABBR);
          PUT SKIP LIST( 'PAGES',PAGES);
          PUT SKIP LIST( 'PUBL_DATE',PUBL_DATE);
          PUT SKIP LIST( 'VERNACULAR_TITL ',VERNACULAR_TITL);
          PUT SKIP LIST( 'ADDRESS', ADDRESS);
          PUT SKIP(2) LIST('ABSTRACT',ABSTRACT);
          PUT SKIP LIST('DESCR_TERMS',DESCR_TERMS);
  % IT50:;
          GO TO NEXTREC;
  FINIS:
  END TESTCBB;
/*
//GO.CBER    DD DSNAME=CBBCCF,UNIT=TAPE9,VOLUME=SER=ILR005,                X
//    DCB=(RECFM=V,BLKSIZE=7200),DISP=OLD,LABEL=(3,SL,,IN)
//*
```

CITATION_NO	2688667
DISREGARD2	'11110000'B
NO_J_TITL_WDS	'0000000000000100'B
YR_PUBL	1968
TYPE_PUBL	'0000000000000001'B
MERGE_DATE	
NO_DESCRIPT	'0000000000000101'B
NO_AUTHORS	'0000000000000100'B
LANG_CODE	1
NO_TITL_WDS	'0000000000010011'B
NO_PG_WDS	'0000000000000000'B
NO_DATE_PUBL_WDS	'0000000000000011'B
NO_VERN_WDS	'0000000000000000'B
DISREGARD3	17
NO_ABSTR	'0000000000000001'B
FILE_SOURCE_CODE	2
AUTHOR_TYPE_CODE	0
COMB_CIT_SEARCH_CODE	2
J_TITL_CODE	EN4Z
STARTING_PG	
DISREGARD4	U
NO_ADDR_WDS	'0000000000001100'B
NO_ABSTR_WDS	'0000000010010011'B
DISREGARD5	

AUTHORS	0	1	ZOR U
AUTHORS	0	2	LOCKER D
AUTHORS	0	2	SCHLEIDER M
AUTHORS	0	2	SULMAN FG

TITLE	EFFECT OF A MONOAMINE OXIDASE INHIBITOR (MERANAZINE) ON BRAIN GLUCOSE-6-PHOSPHATE DEHYDROGENASE ACTIVITY.
J_TITL_ABBR	EUROP J PHARMACOL
PAGES	
PUBL_DATE	1968
VERNACULAR_TITL	
ADDRESS	(SCH. MED., CLIN. RES. UNIT, UNIV. PITTSBURGH, PITTSBURGH, PA.)

ABSTRACT THE EFFECT OF MEBANAZINE, A MONOAMINE OXIDASE INHIBITOR (MAOI), ON THE ACTIVITY OF GLUCOSE-6-PHOSPHATE-DEHYDROGENASE OF THE PITUITARY, HYPOTHALAMUS AND BRAIN OF YOUNG MALE RATS HAS BEEN STUDIED IN VIVO AND IN VITRO. IN VIVO, MEBANAZINE REDUCES THE ACTIVITY OF G-6-P-DEHYDROGENASE, AS DETERMINED IN THE WHOLE PITUITARY AND PER GRAM PITUITARY PROTEIN, WITHOUT AFFECTING THE ACTIVITY OF THIS ENZYME IN BRAIN AND HYPOTHALAMUS, IN VITRO, CONCENTRATIONS OF MEBANAZINE UP TO 4 X 10(-3)M DID NOT AFFECT THE ACTIVITY OF G-6-P-DEHYDROGENASE IN PITUITARY, BRAIN AND HYPOTHALAMUS. THESE RESULTS INDICATE THAT MEBANAZINE HAS A SPECIFIC EFFECT ON THE PITUITARY, AND THE IN VIVO INHIBITION OF G-6-P-DEHYDROGENASE BY MEBANAZINE MAY NOT BE RELATED TO DIRECT ACTION OF THE DRUG ON THE ENZYME, BUT TO INTERFERENCE WITH BIOCHEMICAL CHAIN REACTIONS WHICH CAUSE BIOSYNTHESIS OF G-6-P-DEHYDROGENASE.

DESCR_TERMS GLUCOSEPHOSPHATE DEHYDROGENASE RATS BRAIN
PITUITARY MONOAMINE OXIDASE INHIBITORS

APPENDIX D

READ ROUTINE FOR THE COMPUTERIZED ENGINEERING INDEX FILE
(G. Silva. July 1969)

I. IDENTIFICATION

Title	TEST
User group	Institute of Library Research
Category	CIS Text Processing
Source Language	PL/1
Machine configuration	IBM 360/91
Space requirements	Step region:
	PLIL 158K
	GO 150K
	Core used 84K
Author	G. Silva
Date	July 1969

READ ROUTINE FOR THE COMPUTERIZED ENGINEERING INDEX FILE

II. PURPOSE

1. To read magnetic tape records of the Computerized Engineering Index file. These records contain bibliographical information described under "Record Format" (see Section IV).
2. To identify each field within a record.
3. To provide a detailed printout for each record (header information and variable part).

READ ROUTINE FOR THE COMPUTERIZED ENGINEERING INDEX FILE

III. TAPE SPECIFICATIONS

1. 9 Track Tape
2. 800 BPI
3. No label. Note: There is a tape mark at the beginning of the tape, hence first subparameter of the LABEL parameter in the DD statement is 2. (See DD card description below).
4. One file
5. Volume serial number is ILR080
6. Data set name is ENGDEX
7. Record format is U (undefined)
8. Maximum record length is 8000 bytes
9. Average record length is 2000 bytes

The DD card used to read the tape is:

```
//GO.XYZ    DD    DSNAME=ENGDEX,UNIT=TAPE9,VOLUME=SER=ILR080,
//    DCB=(RECFM=U,BLKSIZE=8000),DISP=OLD,LABEL=(2,BLP,,IN)
```

Note. The subparameter IN (fourth subparameter of the LABEL parameter), tells the system that this tape is to be used as input only and eliminated the operator/computer dialogue on write rings.

READ ROUTINE FOR THE COMPUTERIZED ENGINEERING INDEX FILE

IV. FILE DESCRIPTION

The file contains variable length unblocked records.

Record Format

Records consist of fixed header information followed by a variable length field containing one data element.

Header Information

Block Length	4 bytes
Logical Record Length	4 bytes
ID Number	12 bytes
Security	1 byte
Microfilm	1 byte
Sequential Binary ID	2 bytes

Variable Part

Note. The data elements are stored in variable length fields. The fields are made up of one or more segments length 72 bytes.

Each segment consists of:

1.	A demarcation symbol	3 bytes
2.	A text line	69 bytes

The data elements and corresponding demarcation symbols are:

Demarcation Symbols	Data Elements	
00Ø	Title - 1st line	
ØØØ	2nd - nth line	
09Ø	Subject Headings, Subheading, EI No.	
10Ø	ID Number	
15Ø	Cite Number	
20Ø	Author	
299	Author	
30Ø	Citation - 1st line	
ØØØ	Citation - 2nd line to nth line	
40Ø	Abstract 1st line	
ØØØ	Abstract 2nd line to nth line	
60Ø	Subject Heading, Subheading	
61Ø	Sales Codes	
649	Sales Codes	Unofficial
65Ø	Access Words	
699	Access Words	

READ ROUTINE FOR THE COMPUTERIZED ENGINEERING INDEX FILE

V. CHECKOUT

1. Timing

The program processed 20 logical records with the following results:

Step Name	Step CPU Seconds:
PLIL	3.66s
GO	3.93s
Load timing	.21s
Processing Time	3.72s

186 milliseconds/logical record

2. Error Checking

The only data element checked was the demarcation symbol. No "illegal" symbols were detected.

3. Cost

The tape was opened in 29 runs which took a total of 139.34 cpu seconds at a total cost of $46.08.

4. Remarks

Working with the Engineering Index file has proved somewhat unsatisfactory.

a. The file has a tape mark at the beginning where one normally expects to find a label or nothing at all. This was not mentioned in the accompanying documenation, and led to some waste of time and effort.

READ ROUTINE FOR THE COMPUTERIZED ENGINEERING INDEX FILE

V. CHECKOUT

4. Remarks (continued)

b. An attempt to process 100 records led to the discovery of a bad spot at the 26th record. The 91 was unable to handle that and the tape had to be dumped on the printer on the IBM Model 20. More waste of time and money.

c. The data is input in segments of length 72 bytes (see section IV File Description). A longer data element, such as the abstract, consists of several of these segments. Each segment consists of some text with the remaining portion filled with blanks. In my opinion this is bad for the following reasons:

i. There is a considerable amount of wastage of tape.

ii. Titles, Citations and Abstracts, have to be edited before output, i.e., multiple blanks have to be reduced to one blank, to make the output more readable which increases the processing time considerably.

d. The Subject heading and subheading appear twice: In the 09Ø field accompanied by the EI Number, and a second time in the 60Ø field, without the EI Number. In the first instance the subject heading and the subheading are linked by two dashes (--), whereas

READ ROUTINE FOR THE COMPUTERIZED ENGINEERING INDEX FILE

V. CHECKOUT

4. Remarks (continued)

in the second, they are linked by a comma (,). No reason for the above discrepancies is given, nor is it clear why field 09Ø is repeated in field 60Ø (less EI Number).

e. Field 40Ø contains the abstract. The text of the abstract is always followed by the EI Number. This feature was not mentioned in the documentation.

f. Do we assume that EI "sales codes" are the same as the EI "card service category code"?

g. Access words are labelled "unofficial" in the documentation. None were encountered in the records processed. What is this field designed to contain? Will "access words" appear in future?

h. For neither the Subject Headings nor the Access words is a thesaurus provided. Are they available? Do they have controlled or uncontrolled vocabularies? (suggested by Dr. Reilly)

i. Note that the COMPENDEX file was copied from ILR050 on to ILR080 on the Model 20. This was a disasterous move, since the Model 20 destroyed all the control blocks on the tape. (See Section IV on File Description for original record format.)

READ ROUTINE FOR THE COMPUTERIZED ENGINEERING INDEX FILE

VI. PROGRAM DESCRIPTION

1. Preprocessor variables:

REC	structure name for header information	ENGINDX
REC_SIZE	length of variable part of record	4000 bytes
FLAGNO	number of different demarcation symbols	11
FLAGKAR	length in characters (bytes) of demarcation symbols	2

2. Processor variables: (In order of occurrence in deck)

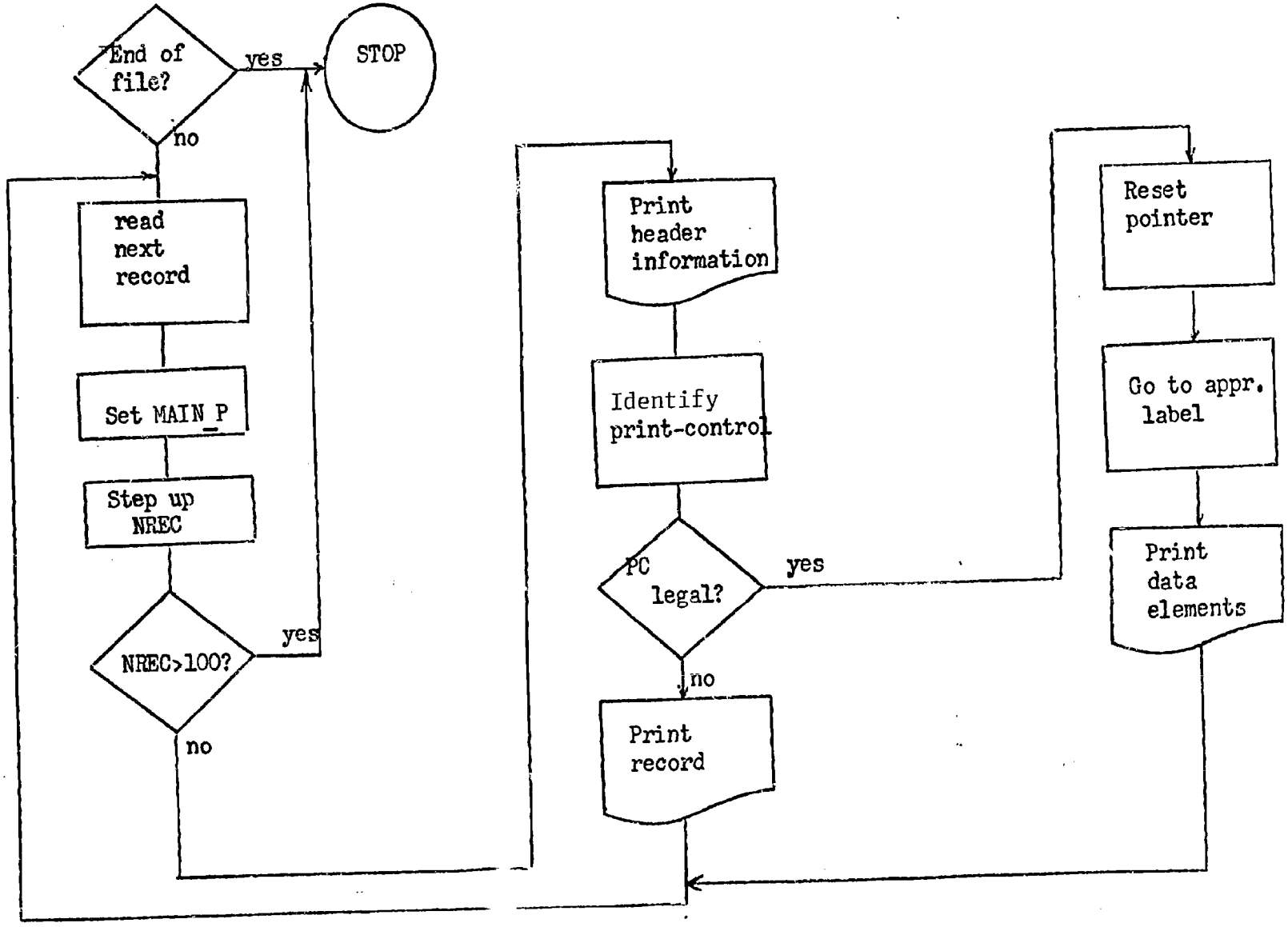
FIXED_PART	contains Header information
REMAINDER	contains the variable portion of record
DUMMY	contains one record
SWITCH	is a label used for returning to fields made up to repeating segments
FLAG	contains the possible demarcation symbols.
LB	is a label array containing the labels leading to the specific operations required by each field.

The next ten variables are self-explanatory

PRINT_CONTROL	serves as a temporary location for demarcation symbols
PROXY_PTR	points to P
LINE	serves as a temporary location for the 'segments' as defined under "Record format"
PC	fixed length character array of length 2 serves as temporary location for demarcation symbols.
NREC	counts the number of records processed

READ ROUTINE FOR THE COMPUTERIZED ENGINEERING INDEX FILE

3. Flowchart



READ ROUTINE FOR THE COMPUTERIZED ENGINEERING INDEX FILE

4. Program listing.

```
COMPILE-TIME MACRO PROCESSOR
MACRO SOURCES LISTING

  1     TEST:
  2             PROC OPTIONS(MAIN);
  3             % DCL REC CHAR;
  4             % REC = 'ENGINDX';
  5             % DCL REC_SIZE FIXED;
  6             % REC_SIZE = 4000;
  7             % DCL FLAGNO FIXED;
  8             % FLAGNO=11;
  9             % DCL FLAGKAR FIXED;
 10             % FLAGKAR=FLAGNO*3;
 11             DCL 1 REC BASED (MAIN_P),
 12                   2 FIXED_PART,
 13                     3 ID_NO CHAR(12),
 14                     3 SECURITY CHAR(1),
 15                     3 MICROFILM CHAR(1),
 16                     3 SEQ_ID_NO BIT(16),
 17                   2 REMAINDER CHAR(REC_SIZE);
 18             DCL DUMMY CHAR(REC_SIZE)VAR;
 19             DCL SWITCH LABEL;
 20             DCL FLAG CHAR(FLAGKAR);
 21             DCL LB(FLAGNO) LABEL INIT(L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11);
 22             DCL TITLE CHAR(140)VAR;
 23             DCL SUBJ_HEAD CHAR(70)VAR ;
 24             DCL CITE_NO CHAR(69)VAR;
 25             DCL AUTHORS CHAR(140)VAR;
 26             DCL CITATION CHAR(140)VAR;
 27             DCL ABSTRACT CHAR(3[illegible])VAR,
 28             DCL SUBJECT_HEADING CHAR(140)VAR;
 29             DCL SALES_CODES(3) CHAR(70)VAR;
 30             DCL ACCWDS1 CHAR(70)VAR;
 31             DCL ACCWDS2 CHAR(70)VAR;
 32             DCL  PRINT_CONTROL CHAR(2)BASED(P );
 33             DCL PROXY FIXED BIN(31)BASED(PROXY_PTR);
 34             DCL LINE CHAR(69)BASED(P);
 35             DCL PC CHAR(3);
 36             ON ENDFILE(XYZ) GO TO FINIS;
 37             FLAG = 'A[illegible]A09A10A15A20A30A40A60A61A65A69';
 38             NREC=0;
 39             PUT PAGE;
 40     NEXTREC:
 41             K=1;
 42             NSALES=0;
 43             DUMMY=(REC_SIZE)' ';
 44             READ FILE(XYZ) INTO (DUMMY);
 45             NREC=NREC+1;
 46             IF NREC> 20  THEN GO TO FINIS;
 47             PUT PAGE;
 48             MAIN_P = ADDR(DUMMY);
 49     /*  PRINT HEADER INFORMATION  */
 50             PUT SKIP LIST('ID_NO',ID_NO);
```

TEST:

MACRO SOURCE LISTING

```
 51              PUT SKIP LIST('SECURITY', SECURITY);
 52              PUT SKIP LIST('MICROFILM', MICROFILM);
 53              PUT SKIP LIST('SEQ_ID_NO',SEQ_ID_NO);
 54              SUBJ_HEAD = '';
 55              TITLE = '';
 56              AUTHORS='';
 57              CITATION='';
 58              ABSTRACT='';
 59              SUBJECT_HEADING='';
 60              SALES_CODES='';
 61              PROXY_PTR = ADDR(P);
 62              DUMMY= SUBSTR(DUMMY,17);
 63              P = ADDR(DUMMY);          LDUMMY = LENGTH(DUMMY);
 64      S10:
 65      /*  IDENTIFY PRINT CONTROL  */
 66              PC = 'A'||PRINT_CONTROL;
 67              I = ( INDEX(FLAG,PC)+2)/3;
 68              IF I=0 THEN DO;
 69              PUT SKIP LIST('PRINT_CONTROL NOT FOUND IN LIST',PRINT_CONTROL);
 70              PUT SKIP LIST(DUMMY);
 71              GO TO S30;
 72                END;
 73                PROXY = PROXY+3;
 74      /*  P NOW POINTS TO THE BEGINNING OF THE TEXT LINE  */
 75              K=K+3;
 76              GO T  LR(I);
 77      S20:
 78              PROXY = PROXY + 69;
 79      /* P NOW POINTS TO THE DEMARCATION SYMBOL  */
 80              K=K+69;   LDUMMY
 81              IF K> LENGTH(DUMMY) THEN GO TO S30;
 82              IF PRINT_CONTROL='  ' THEN DO;
 83              PROXY=PROXY+3;
 84              K=K+3;
 85              GO TO SWITCH;
 86              END;
 87              GO TO S10;
 88      L1:
 89              TITLE = TITLE||' '||LINE;
 90              SWITCH = L1;
 91              GO TO S20;
 92      L2:
 93              SUBJ_HEAD = SUBJ_HEAD||' '||LINE;
 94              SWITCH = L2;
 95              GO TO S20;
 96      L3:
 97              ID_NO = LINE;
 98              SWITCH=L3;
 99              GO TO S20;
100      L4:
```

```
      TEST:

MACRO SOURCE2 LISTING

101               CITE_NO=LINE;
102               GO TO S20;
103         L5:
104               AUTHORS=AUTHORS||' '||LINE;
105               SWITCH=L5;
106               GO TO S20;
107         L6:
108               CITATION=CITATION||' '||LINE;
109               SWITCH=L6;
110               GO TO S20;
111         L7:
112               ABSTRACT=ABSTRACT||' '||LINE;
113               SWITCH=L7;
114               GO TO S20;
115         L8:
116               SUBJECT_HEADING=SUBJECT_HEADING||' '||LINE;
117               SWITCH=L8;
118               GO TO S20;
119         L9:
120               NSALES = NSALES +1;
121               SALES_CODES(NSALES)=LINE;
122               GO TO S20;
123         L10:
124               ACCWDS1=LINE;
125               GO TO S20;
126         L11:
127               ACCWDS2=LINE;
128         S30:
129         /*  EDIT AND PRINT DATA ELEMENTS  */
130               CALL EDIT(TITLE);
131               PUT SKIP LIST('TITLE',TITLE);
132               PUT SKIP LIST('SUBJ_HEAD',SUBJ_HEAD);
133               PUT SKIP LIST('ID_NO',ID_NO);
134               PUT SKIP LIST('CITE_NO',CITE_NO);
135               PUT SKIP LIST('AUTHORS',AUTHORS);
136               CALL EDIT(CITATION);
137               PUT SKIP LIST('CITATION',CITATION);
138               CALL EDIT(ABSTRACT);
139               PUT SKIP LIST('ABSTRACT',ABSTRACT);
140               PUT SKIP LIST('SUBJECT_HEADING',SUBJECT_HEADING);
141               PUT SKIP LIST('SALES_CODES',SALES_CODES);
142               PUT SKIP LIST('ACCESS WORDS',ACCWDS1);
143               PUT SKIP LIST('ACCESS WORDS',ACCWDS2);
144               GO TO NEXTREC;


145
146         EDIT:
147               PROC(LINE);
148         /*  REDUCES MULTIPLE BLANKS TO ONE BLANK  */
```

TEST:

MACRO SOURCE LISTING

```
140             DCL  LINE CHAR(*)VAR;
150     EDO1:
151             I=INDEX(LINE,'  ');
152              IF I¬=0 THEN DO;
153                 IF I=LENGTH(LINE)-1 THEN DO;
154                 LINE=SUBSTR(LINE,1,I);
155                 RETURN;
156                 END;
157               LINE= SUBSTR(LINE,1,I)||SUBSTR(LINE,I+2);
158               GO TO EDO1;
159               END;
160             RETURN;
161     END EDIT;
162     FINIS:
163       END TEST;
```

NO ERROR OR WARNING CONDITION HAS BEEN DETECTED FOR THIS MACRO PASS.

ID_NO FIX69XC30001
SECURITY
MICROFILM N
SEQ_ID_NO '0000000200000001'B
TITLE CALCULATING TRUE VOLTAGE ON PULSE- CHARGED CAPACITO
SUBJ_HEAD RADIO CIRCUITS--PULSE
ID_NO FIX69XC30001
CITE_NO 69- 03 A00- 32745
AUTHORS DAVERIO AJ
CITATION ELECTRO- TECHNOLOGY V 82 N 2 AUG 1968 P 73- 4.
ABSTRACT METHOD FOR REDUCING TIME TO DESIGN CIRCUITS HAVING
TF EQUATIONS; LOWERING OF VOLTAGE DUE TO SERIES ELEMENTS IN CHARGING CIRCUIT
R DURING OFF TIMES OF PULSE ARE CONSIDERED. 01763
SUBJECT_HEADING RADIO CIRCUITS, PULSE
SALES_CODES 00-A168

ACCESS WORDS
ACCESS WORDS

SE- CHARGED CAPACITORS,

01763

G 1968 P 73- 4.
IGN CIRCUITS HAVING CAPACITOR CHARGED BY PULSE TRAIN, USING SET OF APPROXIMA
IN CHARGING CIRCUIT AND RIPPLE PRODUCED BY DISCHARGING ACROSS LOAD CAPACITO

CO-A297

APPENDIX E

READ ROUTINE FOR THE NSA SELECTOR FILE

(G. Silva. November 1969)

I. IDENTIFICATION

Title	TESTCA
User Group	Institute of Library Research
Category	CIS Text Processing
Source Language	PL/1
Machine Configuration	IBM 360/91, one input tape, printer
Space Requirements	Step region: PL1L: 162K GO: 150K Core Used: 80K
Author	G. Silva
Date	November 1969

READ ROUTINE FOR NSA SELECTOR FILE

II. PURPOSE

1. To read magnetic tape records of the Nuclear Science Abstracts Selector File. These records contain bibliographical information described under "Record Format" (see section IV).

2. To identify each field within a record.

3. To provide a detailed printout for each record type (header information and variable part).

READ ROUTINE FOR THE NSA SELECTOR FILE

III. TAPE SPECIFICATIONS

1. 7 track tape
2. 556 BPI
3. Even parity
4. No label
5. Two files

1st file:	NSA Entry File	Volume 23, Issue 14
2nd file:	NSA Selector File	Volume 23, Issue 14

6. Tape identification: ILR 070
7. Record format is U (undefined)
8. Blocksize is 2044 bytes
9. Character coding is BCDIC

The DD card used to read the Selector File is:

```
//GO.KEYWD DD DSNAME=NSA,UNIT=TAPE7,VOL=SER=ILR070,
// DCB=(RECFM=U,BLKSIZE=2044,DEN=1,TRTCH=ET),LABEL=(2,NL,,IN),DISP=OLD
```

Note. The subparameter IN (fourth subparameter of the LABEL parameter), tells the system that this tape is to be used as input only and eliminates the operator/computer dialogue on write rings.

READ ROUTINE FOR THE NSA SELECTOR FILE

IV. FILE DESCRIPTION

Keyword File

Each keyword file will contain the selectors, i.e., keywords and various "additional terms", assigned to items within an issue of NSA.

The logical records, i.e., selectors and associated header for each item, will be ordered by abstract number, then by split*, then by type selector, then by alphabetic sorting sequence of selectors. Thus the file is "linear" with adjacent placement of selectors having split and type selector codes in common.

Record Format

A. Each Selector Record Header format, which is identical for all types of items, describes:

1. Year of the NSA volume
2. NSA issue number
3. NSA abstract number
4. Serial number
5. Overflow field--containing a 1 to indicate that the logical record overflows into the next physical record (lack of overflow is indicated by zero)
6. Type of item
7. NSA Section Subsection Code (revised effective NSA, Vol. 21, Issue 1)

*Split indicators are assigned to selectors which logically relate to each other, permitting search strategies which fail to coordinate selectors assigned to different splits. Thus, splits function as "link" indicators.

READ ROUTINE FOR THE NSA SELECTOR FILE

IV. FILE DESCRIPTION

Record Format (Continued)

8. Blank field

9. Total character count for the item, or portion of the item in a single block (The portion of a logical record continued in a second block begins with a duplicate Selector Record Header containing the number of characters in the second block in the total character count field.)

10. Terminal record mark

B. Each Variable Field is terminated with a record mark. These variable-length selector fields identify:

1. Split indication by means of a left adjusted letter code: A blnk blnk, B blnk blnk,.....A A blnk, A B blnk,.....AAA, AAB,.....ZZZ.

2. Type of selector (1 = keyword, 2 = inorganic compound, 3 = isotope, 8 = additional term, and 9 = "provisional" additional term)

3. Character count for the single selector field

4. Selector in BCD notation

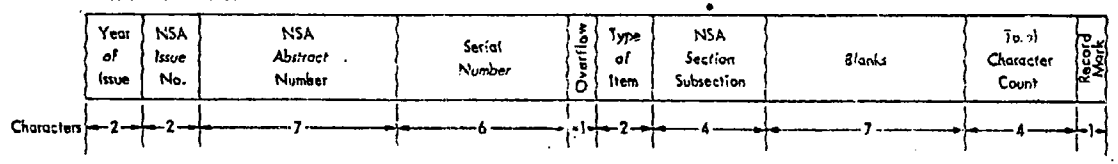
5. Selector number, i.e., unique numeric code for the selector

If a selector appears in more than one split, a separate field is provided for each time it is used.

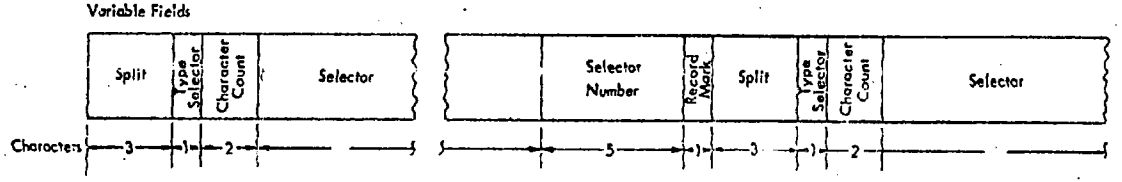
U. S. AEC, Division of Technical Information Extension

KEYWORD FILE (Revision 1, October 1967)

Selector Record Header



Variable Fields



*Revised fields indicated by asterisks.

READ ROUTINE FOR THE NSA SELECTOR FILE

V. CHECKOUT

1. Timing

The program processed 100 records with the following results:

Step Name	Step CPU Seconds
PL1L	2.29s
GO	4.62s
Load timing	.32s
Processing time	4.30s
Average processing time	43 ms/log.rec.

2. Error Checking

None

3. Cost

The total cost of opening the tape at the rate of $.12/MUS was $54.18.

4. Remarks

a. The NSA tape is a 7-track tape probably written on the IBM 7094. Each block therefore consists of a multiple of 6 bytes. In order to achieve this some blocks are padded with blanks at the end. The PL/1 program requires that the real end of the block be known, i.e., the end of the last selector field in the record. Terminal blanks, if present, are therefore removed before the record is processed.

b. None of the 100 records processed were continued in a second block, i.e., no records were found with overflow=1. The longest record found had a character count of 1944.

READ ROUTINE FOR THE NSA SELECTOR FILE

V. CHECKOUT

4. Remarks (Continued)

c. According to information received from the Information Systems Department (July 14th), any blocks of less than 18 characters on tape should be considered "noise" records and ignored. At present the program does not check for "noise" records, but it is advisable that such a check be built into it if this data base is used for production.

VI. PROGRAM DESCRIPTION

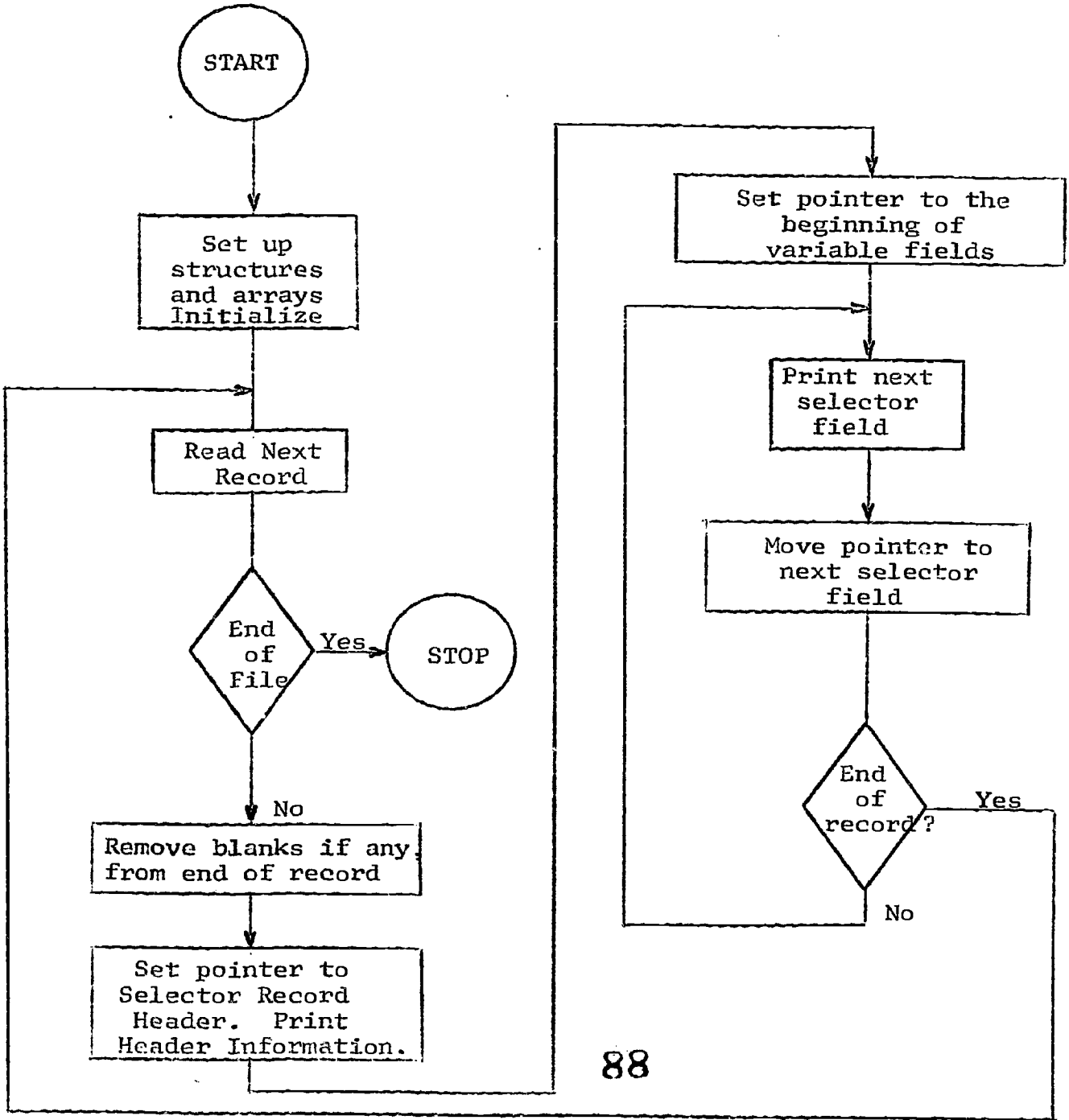
1. The logical flow of the program is fairly simple and a verbal description was therefore deemed redundant. The reader is referred to the attached flowchart and program listings.

2. All variable names used in the program are self-explanatory.

READ ROUTINE FOR THE NSA KEYWORD FILE

VI. PROGRAM DESCRIPTION

3. Flowchart



```
//NRB409MT JOB '1C,1CCC',SILVA
// EXEC PL1
 (SUBSCRIPTRANGE):(STRINGRANGE):
 KWDFILE: PROCEDURE OPTIONS(MAIN);
 /* NSA KEYWORD FILE */
 /* SECOND FILE ON NSA REEL */
         DCL 1 REC BASED(MAIN_P),
               2 HEADER,
                 3 YR_ISSUE CHAR(2),
                 3 ISSUE_NO CHAR(2),
                 3 ABSTR_NO CHAR(7),
                 3 SERIAL_NO CHAR(6),
                 3 OVERFLOW CHAR(1),
                 3 ITEM_TYPE CHAR(1),
                 3 SPACE1 CHAR(1),
                 3 SEC_SUBSEC_CODE CHAR(4),
                 3 BLANKS CHAR(7),
                 3 TOTAL_CHAR_CT CHAR(4),
                 3 RECORD_MARK CHAR(1),
               2 VAR_FIELDS CHAR(2000);
         DCL 1 SUBREC BASED(P),
               2 FIXED_PART,
                 3 SPLIT CHAR(3),
                 3 SELECTOR_TYPE CHAR(1),
                 3 CHAR_CT CHAR(2),
               2 SELECTORS CHAR(100);
         DCL PROXY FIXED BIN(31) BASED(PROXY_PTR);
         DCL N FIXED BIN;
         DCL SELECTOR_NO CHAR(5);
         DCL TESTAREA CHAR(2004)VAR;
         DCL SELECT CHAR(48)VAR;
         ON ENDFILE (KEYWD) GO TO FINIS;
         PROXY_PTR=ADDR(P);
         KTR=0;
 NEXTREC:
         TESTAREA=(2004)' ';
         KTR=KTR+1;
         IF KTR>100 THEN GO TO FINIS;
         READ FILE(KEYWD) INTO (TESTAREA);
         LTEST=LENGTH(TESTAREA);
 /* REMOVE BLANKS, IF ANY, FROM END OF RECORD */
 KF1:
         IF SUBSTR(TESTAREA, LTEST,1)¬=' ' THEN GO TO KF5;
         TESTAREA=SUBSTR(TESTAREA,1,LTEST-1);
         LTEST=LTEST-1;
         GO TO KF1;
 KF5:
         MAIN_P=ADDR(TESTAREA);
         PUT PAGE;
 /* PRINT OUT MAIN HEADER INFORMATION */
         PUT EDIT('HEADER INFORMATION')(SKIP, COL(60), A);
         PUT EDIT('_________________')(SKIP(0), COL(60), A);
         PUT EDIT('YEAR OF NSA VOLUME', YR_ISSUE)(SKIP(2), COL(40),
         A, COL(70), A);
         PUT EDIT('NSA ISSUE NO', ISSUE_NO)
         (SKIP, COL(40), A, COL(70), A);
         PUT EDIT('NSA ABSTRACT NO', ABSTR_NO)
         (SKIP, COL(40), A, COL(70), A);
         PUT EDIT('SERIAL NO', SERIAL_NO)
         (SKIP, COL(40), A, COL(70), A);
         PUT EDIT('OVERFLOW', OVERFLOW)
         (SKIP, COL(40), A, COL(70), A);
```

```
          PUT EDIT('TYPE OF ITEM', ITEM_TYPE)
          (SKIP, COL(40), A, COL(70), A);
          PUT EDIT('NSA SECTION, SUBSECTION', SEC_SUBSEC_CODE)
          (SKIP, COL(40), A, COL(70), A);
          PUT EDIT('TOTAL CHAR COUNT', TOTAL_CHAR_CT)
          (SKIP, COL(40), A, COL(70), A);
          P=ADDR(VAR_FIELDS);
          KT=36;
          PUT EDIT('SUBHEADER INFORMATION')(SKIP(5), COL(58), A);
          PUT EDIT('_____')(SKIP(0), COL(58), A);
          PUT EDIT('SPLIT', 'SELECTOR TYPE', 'CHAR COUNT', 'SELECTOR',
          'SELECTOR NO')(SKIP(5), COL(10), A, COL(35), A, COL(60), A,
          COL(85), A, COL(110), A);
KF10:
          I=1;
          N=CHAR_CT;
          SELECT=SUBSTR(SELECTORS,I,N);
          I=I+N;
          SELECTOR_NO=SUBSTR(SELECTORS,I,5);
          PUT EDIT(SPLIT, SELECTOR_TYPE, CHAR_CT, SELECT, SELECTOR_NO)
          (SKIP, COL(10), A, COL(35), A, COL(60), A, COL(85), A,
          COL(110), A);
          KT=KT+N+12;
          I=I+11;
          IF KT=LTEST THEN DO;
/* CHECK OVERFLOW */
          IF OVERFLOW='1' THEN
          PUT SKIP LIST('RECORD CONTINUED IN THE NEXT BLOCK.');
          GO TO NEXTREC;
          END;
          PROXY=PROXY+I;
          GO TO KF10;
FINIS:
          END KWDFILE;
/*
//GO.KEYWD DD DSNAME=NSA,UNIT=TAPE7,VOL=SER=ILR070,
//  DCB=(RECFM=U,BLKSIZE=2044,DEN=1,TRTCH=ET),LABEL=(2,NL,,IN),DISP=OLD
/*
```

HEADER INFORMATION

YEAR OF NSA VOLUME	69
NSA ISSUE NO	14
NSA ABSTRACT NO	2326175
SERIAL NO	040123
OVERFLOW	0
TYPE OF ITEM	J
NSA SECTION, SUBSECTION	2011
TOTAL CHAR COUNT	0432

SUBHEADER INFORMATION

SPLIT	SELECTOR TYPE	CHAR COUNT	SELECTOR	SELECTOR NO
A	1	24	ACTIVATION ANALYSIS	00050
B	1	12	BIBLIOGRAPHY	00523
C	1	18	CHEMICAL ANALYSIS	00991
D	1	18	DETERMINATION	01309
E	1	12	PHOTOMETRY	03534
F	1	18	SOLVENT EXTRACTION	04533
G	1	12	SPECTROSCOPY	04546
H	1	12	ZIRCONIUM	05515
	1	12	HAFNIUM	01989
	1	12	VANADIUM	05252
	1	12	NIOBIUM	03175
	1	12	TANTALUM	04694
	1	12	CHROMIUM	01040
	1	12	MOLYBDENUM	02947
	1	12	TUNGSTEN	05129

APPENDIX F

READ ROUTINE FOR THE CHEMICAL ABSTRACTS CONDENSATE FILE ILR060

By

Georgette Silva

July 1969

Institute of Library Research
University of California
Los Angeles, California

READ ROUTINE FOR THE CHEMICAL ABSTRACTS CONDENSATE FILE

I. IDENTIFICATION

Title	TESTCA
User Group	Institute of Library Research
Category	CIS Text Processing
Source Language	PL/1
Machine configuration	IBM 360/91, one input tape, printer
Space requirements	Step region: PLIL 158k Core Used 76k GO 150k
Author	C. Silva
Date	July 1969

READ ROUTINE FOR THE CHEMICAL ABSTRACTS CONDENSATE FILE

II. PURPOSE

1. To read magnetic tape records of the Chemical Abstracts Condensate file. These records contain bibliographical information described under "Record Format" (see section IV.).
2. To identify each field within a record.
3. To provide a detailed printout for each record type (header information and variable part).

READ ROUTINE FOR THE CHEMICAL ABSTRACTS CONDENSATE FILE

III. TAPE SPECIFICATIONS

1. 9 track tape
2. 800 BPI
3. Standard label
4. One file
5. Volume serial number is 000000
6. Data set name is CAISSV
7. Record format is variable blocked (VB)
8. Logical record length is 996
9. Block size is 1000

The DD card used to read the tape is:

```
//GO.CHEMAB  DD DSNAME=CAISSV,UNIT=TAPE9,VOLUME=SER=000000,          X
//     DCB=(RECFM=VB,BLKSIZE=1000,LRECL=996),DISP=OLD,LABEL=(1,SL,,IN)
```

Note. The subparameter IN (Fourth subparameter of the LABEL parameter), tells the system that this tape is to be used as input only and eliminates the operator/computer dialogue on write rings.

READ ROUTINE FOR THE CHEMICAL ABSTRACTS CONDENSATE FILE

IV. File description

The file contains variable length blocked records. The records are of five types, each type being designated within the record itself.

Record format

Records consist of fixed header information followed by a variable length field except for the type-1 record which is of fixed length throughout.

Header information:

Abstract number	7 bytes
Blanks	1 byte
Volume	2 bytes
Number	2 bytes
Blanks	1 byte
Record type	1 byte
Blanks	2 bytes

Variable part:

Record:	Data element:
Type-2	Title
Type-3	Author
Type-4	Where published
Type-5	Key words

The second part of the type-1 record is of fixed length (24 bytes) and consists of the following elements:

Journal CODEN	6 bytes
Volume	4 bytes
Number	4 bytes
Year	2 bytes
Starting pg. number	4 bytes
Ending pg. number	4 bytes

Note that each abstract may have several type-5 records (Keywords) associated with it.

READ ROUTINE FOR THE CHEMICAL ABSTRACTS CONDENSATE FILE

V. CHECKOUT

1. Timing

The program processed 100 logical records with the following results:

Step Name	Step CPU Seconds
PLIL	2.54s
GO	0.68s
Load timing	0.23s
Processing time	0.45s

i.e. 4.5msecs/logical record

The following is an approximation to the processing time required for a full bibliographical unit (a sequence of record type 1,4,2,3,4 followed by n type-5 records):

$$t=22.5 + 4.5 \times n$$

where t is in milliseconds.

2. Error checking

a. Rec-type field.

No "illegal" record types were detected.

b. A check was made for records consisting of header information followed by a blank variable part. Some were found.

3. Cost

The tape was opened in 10 runs which took a total of 28.07cpu seconds at a total cost of $10.88.

READ ROUTINE FOR THE CHEMICAL ABSTRACTS CONDENSATE FILE

VI. PROGRAM DESCRIPTION

1. Preprocessor variables:

REC	structure name for header information	CACOND
REC_SIZE	length of variable part of record	300 bytes
FLAGNO	number of different record types	5
FLAGKAR	length in bytes of field containing record type	1

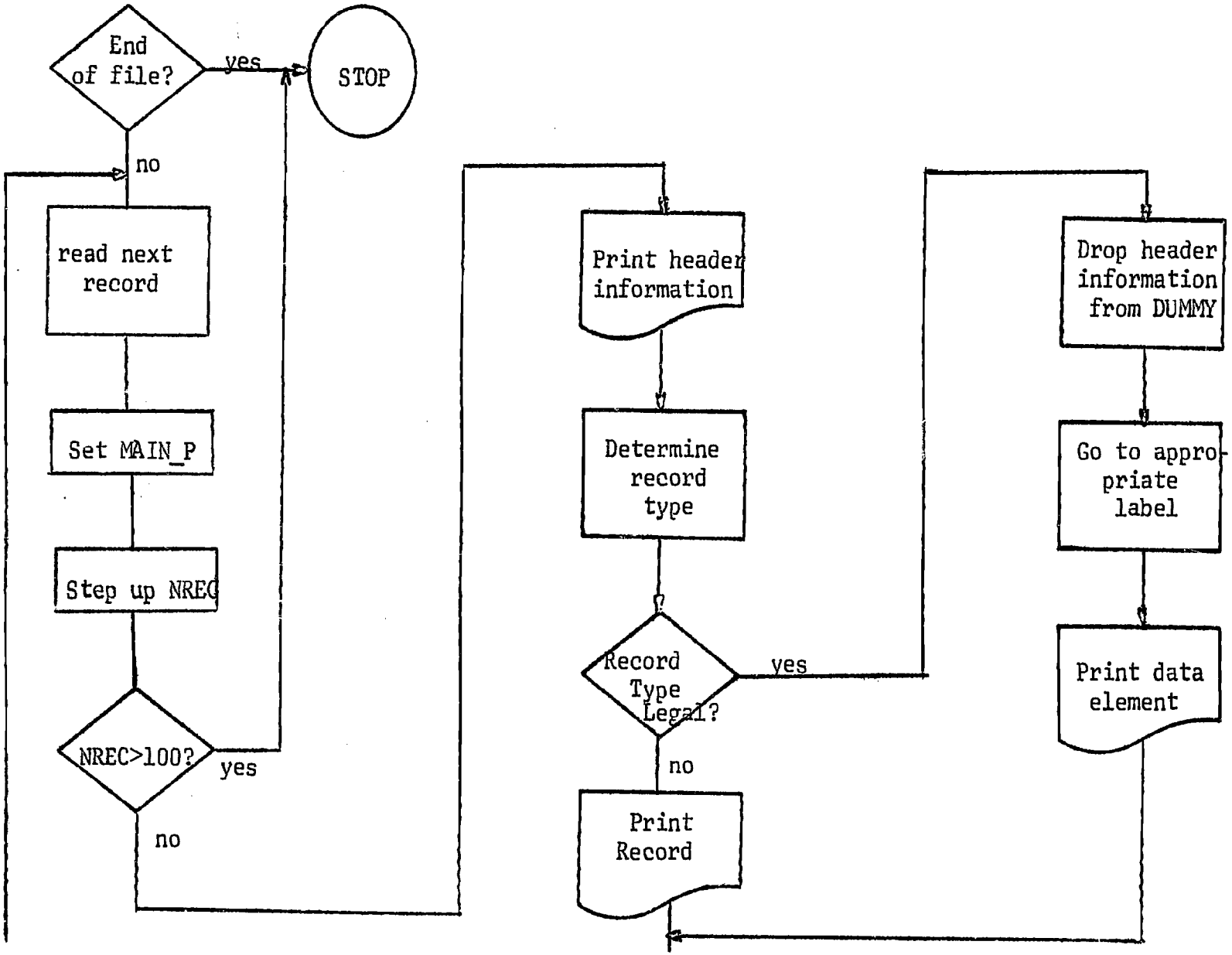
2. Processor variables: (in order of occurrence in deck)

FLAG	contains the possible record types.
LB	is a label array containing the labels leading to the specific operations required by each record type.
CACOND	structure name containing header information
REC1	structure containing second part of record type-1.
DUMMY	contains one logical record
NREC	counts the number of records processed.

The other variables are self-explanatory.

READ ROUTINE FOR THE CHEMICAL ABSTRACTS CONDENSATE FILE

3. Flowchart



```
// EXEC PL1,PARM='M,SM=(2,72,1),ST',PG=200
//PL1L.SYSIN DD *
  TESTCA:
         PROCEDURE OPTIONS(MAIN);
         % DCL REC CHAR;
         % REC = 'CACOND';
         % DCL REC_SIZE FIXED;
         % REC_SIZE=300;
         % DCL FLAGNO FIXED;
         % FLAGNO=5;
         % DCL FLAGKAR FIXED;
         % FLAGKAR=1;
         DCL FLAG CHAR(FLAGNO) INIT('12345');
         DCL LB(FLAGNO) LABEL INIT(L1,L2,L3,L4,L5);
         DCL 1 REC BASED(MAIN_P),
               2 FIXED_PART,
                 3 ABSTR_NO CHAR(7),
                 3 SPACE1 CHAR(1),
                 3 VOL CHAR(2),
                 3 NO CHAR(2),
                 3 SPACE2 CHAR(1),
                 3 REC_TYPE CHAR(FLAGKAR),
                 3 SPACE3 CHAR(2);
         DCL 1 REC1 BASED(MAIN_P),
               2 J_CODEN CHAR(6),
               2 VOLUME CHAR(4),
               2 NUMBER CHAR(4),
               2 YEAR CHAR(2),
               2 STPGNO CHAR(4),
               2 ENDPGNO CHAR(4);
         DCL DUMMY CHAR(REC_SIZE)VAR;
         ON ENDFILE(CHEMAB) GO TO FINIS;
         NREC=0;
         PUT PAGE;
         MAIN_P=ADDR(DUMMY);
  NEXTREC:
         DUMMY=(REC_SIZE)' ';
         READ FILE(CHEMAB) INTO(DUMMY);
         NREC=NREC+1;
         IF NREC>100 THEN GO TO FINIS;
         PUT SKIP (3) LIST('ABSTRACT NUMBER', ABSTR_NO);
         PUT SKIP LIST('VOLUME', VOL   );
         PUT SKIP LIST('NO', NO);
         PUT SKIP LIST('REC_TYPE', REC_TYPE);
         I  = INDEX(FLAG,REC_TYPE);
   /* PROTECTION AGAINST ILLEGAL RECORD TYPES  */
         IF I=0 THEN DO;
         PUT SKIP LIST(' ILLEGAL REC_TYPE', REC_TYPE);
         PUT SKIP LIST(DUMMY);
         GO TO NEXTREC;
         END;
  /* PROTECTION AGAINST BLANK RECORDS */
         IF LENGTH(DUMMY)<17 THEN DO;
         PUT SKIP LIST('SUSPECT RECORD','NREC',NREC,DUMMY);
         NREC=NREC+1;
         GO TO NEXTREC;
         END;
         DUMMY= SUBSTR(DUMMY,17);
         GO TO LB(I);
   /* FOR TYPE-1 RECORDS DO:   */

         PUT SKIP LIST('J_CODEN',J_CODEN);
```

```
        PUT SKIP LIST('VOLUME',VOLUME);
        PUT SKIP LIST('NUMBER',NUMBER);
        PUT SKIP LIST('YEAR',YEAR);
        PUT SKIP LIST('STARTING PG. NO.',STPGNO);
        PUT SKIP LIST('ENDING PAGE NO',ENDPGNO);
        GO TO NEXTREC;
/* FOR TYPE-2 RECORDS DO:  */
L2:
        PUT SKIP LIST('ARTICLE TITLE',DUMMY);
        GO TO NEXTREC;
/* FOR TYPE-3 RECORDS DO:  */
L3:
        PUT SKIP LIST('AUTHORS', DUMMY);
        GO TO NEXTREC;
/* FOR TYPE-4 RECORDS DO:  */
L4:
        PUT SKIP LIST( 'WHERE PUBLISHED', DUMMY);
        GO TO NEXTREC;
/* FOR TYPE-5 RECORDS DO:  */
L5:
        PUT SKIP LIST( ' INDEX TERMS', DUMMY);
        GO TO NEXTREC;

FINIS:
END TESTCA;
*
/GO.CHEMAB  DD DSNAME=CAISSV,UNIT=TAPE9,VOLUME=SER=000000,
/   DCB=(RECFM=VB,BLKSIZE=1000,LRECL=996),DISP=OLD,LABEL=(1,SL,,IN)
*
```

```
ABSTRACT NUMBER      093027X
VOLUME               70
NO                   21
REC_TYPE             1
J_CODEN              IVNMAW
VOLUME               0005
NUMBER               0001
YEAR                 69
STARTING PG. NO.     0209
ENDING PAGE NO       0010

ABSTRACT NUMBER      093027X
VOLUME               70
NO                   21
REC_TYPE             4
WHERE PUBLISHED      IZV. AKAD. NAUK SSSR, NEORG. MATER.

ABSTRACT NUMBER      093027X
VOLUME               70
NO                   21
REC_TYPE             2
ARTICLE TITLE        COMPOUNDS OF VARIABLE COMPOSITION AND NONSTOICHIOMETRIC COMPOUNDS.==

ABSTRACT NUMBER      093027X
VOLUME               70
NO                   21
REC_TYPE             3
AUTHORS              ORMONT BF.

ABSTRACT NUMBER      093027X
VOLUME               70
NO                   21
REC_TYPE             4
WHERE PUBLISHED      (LENINGRAD. ELEKTROTEKH. INST. IM. UL'YANOVA (LENINA), LENINGRAD, USSR).

ABSTRACT NUMBER      093027X
VOLUME               70
NO                   21
REC_TYPE             5
 INDEX TERMS         FORMULAS CHEM REVIEW

ABSTRACT NUMBER      093027X
VOLUME               70
NO                   21
REC_TYPE             5
 INDEX TERMS         REVIEW CHEM FORMULAS
```

APPENDIX G

INVENTORY OF PROGRAMS AND SUBROUTINES

1. PROGRAM:

   AUTHOR: STEVE SILVER

   STATUS: IN DEVELOPMENT

   DESCRIPTION: INTX is a system for assembling, loading, and interactively interpreting 360 assembly language source programs under the UCLA URSA time-sharing system. It permits URSA customers to produce completely interactive programs without any risk of system damage--a feature that is not currently guaranteed by true URSA processors. The system is fully operational now but minor tuning and development will continue.

2. PROGRAM: FMS

   AUTHOR: STEVE SILVER

   STATUS: COMPLETE

   DESCRIPTION: FMS is designed for high speed, cheap generation of natural language documents for mixed text and command input. It is very useful for producing final copies of technical reports requiring extensive minor alternation before publication. The system needs some work before the current version can be released as safe.

3. PROGRAM: DISCUS

   AUTHOR: STEVE SILVER

   STATUS: IN DEVELOPMENT

   DESCRIPTION: DISCUS is a computer aided instruction system oriented towards CRT graphic terminals. It is a two part system: a compiler that runs on any 360 and an executer whose I/O package must be tailored to the individual time-sharing system. It is currently under development but is expected to be completed very shortly.

4. PROGRAM: FAMULUS

   AUTHOR: JERRY PINE

   STATUS: COMPLETE

   DESCRIPTION: FAMULUS is designed to process personal reference collections maintained by researchers. However, its basic structure renders it suitable for a large number of other applications. For this purpose it can be regarded as a general-purpose text-handling system.

FAMULUS will maintain many types of information files which can be broken into units or records with sub-categories or fields which can be identified. In a personnel information file, the data on each person comprises one record. The record may have up to 10 distinct fields in which are entered name, date of birth, job title, etc. In bibliographic files, the citation is the record, and fields are used for author, title, date, keywords, abstracts, etc.

5. PROGRAM: WORDLST

AUTHOR: LINDA MIROFF

STATUS: COMPLETE

DESCRIPTION: WORDLST is a PL/I subroutine which breaks a given character string into individual words. The user specifies characters which are to act as word delimiters; these characters should not be embedded within words. WORDLST stores the individual words without any delimiters in an array. A number giving the relative position of the word in the string is stored in a parallel array. In this way other routines have access to the words as they appear within the string. A list of words, called an exclusion list, may be used to exclude non-content words. Words less than a given length may also be excluded.

6. PROGRAM: CONTEXT

AUTHOR: LINDA MIROFF

STATUS: COMPLETE

DESCRIPTION: CONTEXT is a PL/I subroutine which does most of the "work" involved in building a KWIC index. It finds the contexts of given words in a given string. It can be used in conjunction with an input routine, the WORDLST routine, a sort routine, and an input routine, to produce a KWIC index.

The context of a word consists of as much of the surrounding sentence as can be fitted in the space allowed. The number of characters in the context line is set by the user. When all of the sentence cannot be fitted in, parts of the sentence furthest from the word are truncated. The word always appears in the middle of the line. Wrap-around (from left to right or from right to left) is performed to make use of extra room on either side of the word. When there is a space a "-" indicates the beginning of the context; a "+" is used at the end of the context to indicate that part of the sentence is missing.

7. PROGRAM: CONTITL

   AUTHOR: LINDA MIROFF

   STATUS: COMPLETE

   DESCRIPTION: CONTITL is a special version of CONTEXT, to be used where the text always consists of titles or single sentences. CONTITL produces KWIC records in the same manner as CONTEXT but does not recognize sentence endings or beginnings. The output produced is like that produced by CONTEXT.

8. PROGRAM: DEMO1

   AUTHOR: GEORGETTE SILVA

   STATUS: COMPLETE

   DESCRIPTION: DEMO1 reads text from magnetic tape, breaks it into individual words, sorts them alphabetically and in order of decreasing frequency, and lastly, prints each word surrounded by context in order of occurrence in the text. All these tasks are carried out in internal memory. The program performs these tasks on a fairly methodological level. The sort, for example, is simply what is termed a "bubble sort". The text must be limited to 10,000 characters. The latter condition is not serious, however, since this is more than adequate to cope with short texts such as abstracts. The main virtue of this program is that it consists of one deck, and carries out all the tasks in one run. It may, therefore, be useful as a demonstration program for beginners.

9. PROGRAM: TEXTMT

   AUTHOR: GEORGETTE SILVA

   STATUS: COMPLETE

   DESCRIPTION: TEXTMT is somewhat more sophisticated than DEMO1 and has a few additional capabilities: it does character conversion by table lookup, retrieves index terms in context, and uses a binary tree sort for the dictionary and frequency sort. These operations are carried out in internal memory, and the amount of text processed at any given time is still limited.

10. PROGRAM: INDX

AUTHOR: GEORGETTE SILVA

STATUS: COMPLETE

DESCRIPTION: The INDEX program is designed to cope with texts in prose, poetry, and dramatic forms, as well as with transcriptions of oral discourse. The program reads input text from cards and produces records each containing one textual word or punctuation mark, and index information showing where the word/punctuation mark occurred in the text. Each word/punctuation mark is indexed as to volume, chapter, paragraph, sentence, and word within sentence, and word within sentence location. These records are stored in a temporary file which forms the input to program GAMMA.

11. PROGRAM: GAMMA

AUTHOR: GEORGETTE SILVA

STATUS: COMPLETE

DESCRIPTION: Program GAMMA produces a word index and word frequency list from the output of INDX. The word index is in alphabetical order by word and includes frequency count. The word frequency list is ordered alphabetically within descending frequency. GAMMA invokes the IBM sort/merge to do the necessary sorting and prints the final output.

12. PROGRAM: CONCORD

AUTHOR: GEORGETTE SILVA

STATUS: COMPLETE

DESCRIPTION: CONCORD reads input text from cards in stream form and produces concordance records (in KWIC format) on each word in the text which is not in an exclusion list. The records are written in a sequential data set which forms the input to program ALPHA.

13. PROGRAM: ALPHA

AUTHOR: GEORGETTE SILVA

STATUS: COMPLETE

DESCRIPTION: ALPHA sorts the output records produced by CONCORD into ascending sequence by word and identification. It invokes the IBM sort/merge and prints the concordance, underlining the keyword of each line of text.

14. PROGRAM: ASPEN (California Code and State Constitution) Read Routine

AUTHOR: GEORGETTE SILVA

STATUS: COMPLETE

DESCRIPTION: This routine reads magnetic tape records of the Aspen file (containing full-text images of 298 documents from the California Code and State Constitution), identifies each field within a record, and provides a detailed printout for each record type.

15. PROGRAM: COMPENDEX Read Routine

AUTHOR: GEORGETTE SILVA

STATUS: COMPLETE

DESCRIPTION: This routine reads magnetic tape records of the Computerized Engineering Index file (containing bibliographic information), identifies each field within a record, and provides a detailed printout for each record type.

16. PROGRAM: CHEMICAL ABSTRACTS CONDENSATE Read Routine

AUTHOR: GEORGETTE SILVA

STATUS: COMPLETE

DESCRIPTION: This routine reads magnetic tape records of the Chemical Abstracts Condensate file (containing bibliographic information), identifies each field within a record, and provides a detailed printout for each record type.

17. PROGRAM: COMMUNICATIONS OF BEHAVIORAL BIOLOGY (CBE) Read Routine

AUTHOR: GEORGETTE SILVA

STATUS: COMPLETE

DESCRIPTION: This routine reads magnetic tape records of the Communications of Behavioral Biology file (containing bibliographic information), identifies each field within a record, and provides a detailed printout for each record type.

18. PROGRAM: NUCLEAR SCIENCE ABSTRACTS (NSA) ENTRY Read Routine

AUTHOR: GEORGETTE SILVA

STATUS: COMPLETE

DESCRIPTION: This routine reads magnetic tape records of the Nuclear Science Abstracts Entry file (containing bibliographic information), identifies each field within a record, and provides a detailed printout for each record type.

19. PROGRAM: NUCLEAR SCIENCE ABSTRACTS (NSA) SELECTOR Read Routine

AUTHOR: GEORGETTE SILVA

STATUS: COMPLETE

DESCRIPTION: This routine reads magnetic tape records of the Nuclear Science Abstracts Selector file (containing condensed bibliographic information), identifies each field within a record, and provides a detailed printout for each record type.

20. PROGRAM: BINCDF

AUTHOR: STUART BEAL

STATUS: COMPLETE

DESCRIPTION: A FORTRAN function for computing the binomial cumulative distribution.

21. PROGRAM: PARAMETER P TEST

AUTHOR: STUART BEAL

STATUS: COMPLETE

DESCRIPTION: A FORTRAN program which tests the P parameter of the binomial distribution.

22. PROGRAM: EIRC

AUTHOR: AEINT DE BOER

STATUS: COMPLETE

DESCRIPTION: The ERIC file search system will search, maintain and provide listings from the ERIC file.

a) MULFSCH--This is a batch program to search an inverted index to an ERIC file or a sequential ERIC file by a Boolean combination of descriptors or a list of accession numbers. It is designed to be extended to search other files.

b) ERICMIF--This is a batch program to create an inverted index to an ERIC file. It reads a sequential ERIC file, extracts the descriptors and accession number, and writes a direct access file of descriptors with corresponding accession numbers. This file can be used by MULFSCH, ERICIFL, and ERICDFL.

c) ERICIFL--This is a batch program to list an inverted index to an ERIC file. The listing includes descriptors and frequency of use and can be either one up or two up. The one up listing can include the corresponding accession numbers as an option. The listing is ascending by accession number within descriptor.

d) ERICDFL--This is a batch program to list an inverted index to an ERIC file. The listing includes descriptors and frequency of use and is ordered by descending frequency of use.

23. PROGRAM: CENS60V

AUTHOR: AEINT DE BOER

STATUS: COMPLETE

DESCRIPTION: CENS60V is a batch program to verify a set of documented universes for the 1960 1/1000 sample census tape.

24. PROGRAM: UTILITY PROGRAMS

AUTHOR: AEINT DE BOER

STATUS: COMPLETE

DESCRIPTION:

a) STIMER (a modification of a routine borrowed by and from Stu Beal) is a short assembler language subroutine to enable a PL/I programmer to measure elapsed task CPU time and elapsed real time.

b) PGMPRNT is a batch program to print programs in a format suitable for inclusion in a thesis. Includes provisions for a figure title, figure number, number of parts, statement numbers and page numbers.