INFORMATION STORAGE AND RETRIEVAL SYSTEM

A USER'S MANUAL

BOLT BERANEK AND NEWMAN INC
50 MOULTON STREET, CAMBRIDGE, MASS. 02138

INFORMATION STORAGE AND RETRIEVAL SYSTEM

A USER'S MANUAL

Sally T. Castleman

Medical Information Technology Department
BOLT BERANEK AND NEWMAN INC
Cambridge, Massachusetts 02138

April 1968

The work described in this document received support through a contract, PH43-62-850, from the Institute of General Medical Sciences, National Institutes of Health, and through a grant from the American Hospital Association.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF TYPESCRIPTS

FOREWORD

The Information Storage and Retrieval System (ISR System), developed by Bolt Beranek and Newman Inc., is a general-purpose system designed for users whose information-handling problems are varied and whose needs cannot be determined and programmed in advance. The programs of the ISR System provide the individual user with the facility to select and implement the solutions that are suited to his own particular needs. The language is designed to enable the computer layman to use the system directly, with no need for a programmer as intermediary.

The basic procedure for using the ISR System is as follows. From a teletypewriter terminal located in his own office, laboratory, or classroom, a user requests loading of the desired program.* He then carries on an English-language dialogue with the computer. In a question-and-answer exchange,† the user interacts with the computer to tailor the general system to his specific information-handling application.

---

*A user calls a program by hitting the key on his terminal that is marked BREAK or SIGNAL. This causes a heading of the time and date and terminal line number (which often varies with each use) to be printed, followed by a carriage return and line feed. The user then types in the letters that designate the program that he wants (in this Manual, a program's call letters are always printed in capitals. They are usually 5 or 6 characters), a space, and then the 2 or 3 characters that are his own initials. As with all input, he terminates his entry by hitting the ENTER key. Then the program takes over the control and begins to ask questions.

†An ISR program, when called by a user, appears in the form of a questionnaire. The program prints a question number and a

The programs of the ISR System perform three major phases of data-file manipulation. The initial phase permits the user to define the structure and nature of his data file.

The second phase, assimilation of data into the prescribed file format, may be performed by using either or both of the data-assimilation methods. The user may enter data from punched-card images on magnetic tape or he may enter his data online from his teletypewriter terminal.

The third phase is the retrieval and abstraction of the data in ways meaningful to the user. In this phase, the user specifies the parts of the file that he wants to consider and what functions he wants performed on them. (Sophisticated retrieval, data reduction, and statistical analysis can be carried out with a special user-oriented language, which is described in the *STRCOMP and ISRCOMP User's Manual.*)

This Manual on how to use the programs of the ISR System is meant both as a guide for beginners and as a reference for established users.

---

question; then it waits for the user to type an answer. If the answer is acceptable, the process is repeated for the next question. If the user's answer is unacceptable, the question is reasked. The user may determine his error and correct his answer by one of several methods. (See Appendix A for a fuller description of the use of the program questionnaire.)

INTRODUCTION

Some preliminary definitions will provide an understanding of the way in which data are organized. The three basic divisions of data are the file, the record, and the field.

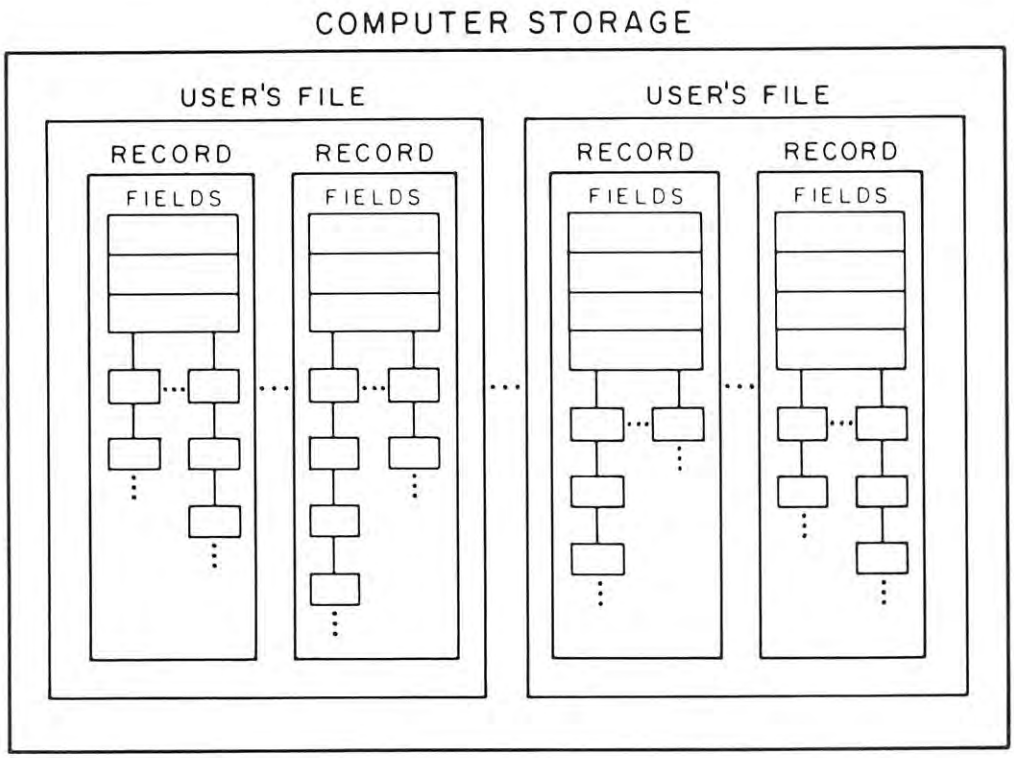
File: a group of related records stored together on magnetic tape or on the Fastrand drum (online storage device). All information for a particular investigation is contained in a file with a unique file number. A file generally describes the population of the study or the project.

Record: one item in a file. Each record in a file has the same field structure. Some field(s) of information in the record, the record-identifying field(s), distinguish it from all other records. A record describes one individual in the population.

Field: the elementary unit of data that is to be referenced in a computer record. Field *names* are names assigned to these units. Field *values* are the values that are stored. A field usually describes one attribute of the individual.

The following Figure illustrates the interrelationship of the field, the record, and the file. The relationship between files

also is shown. A user's file might be of part numbers. A record might pertain to one part. Part description might be one field.



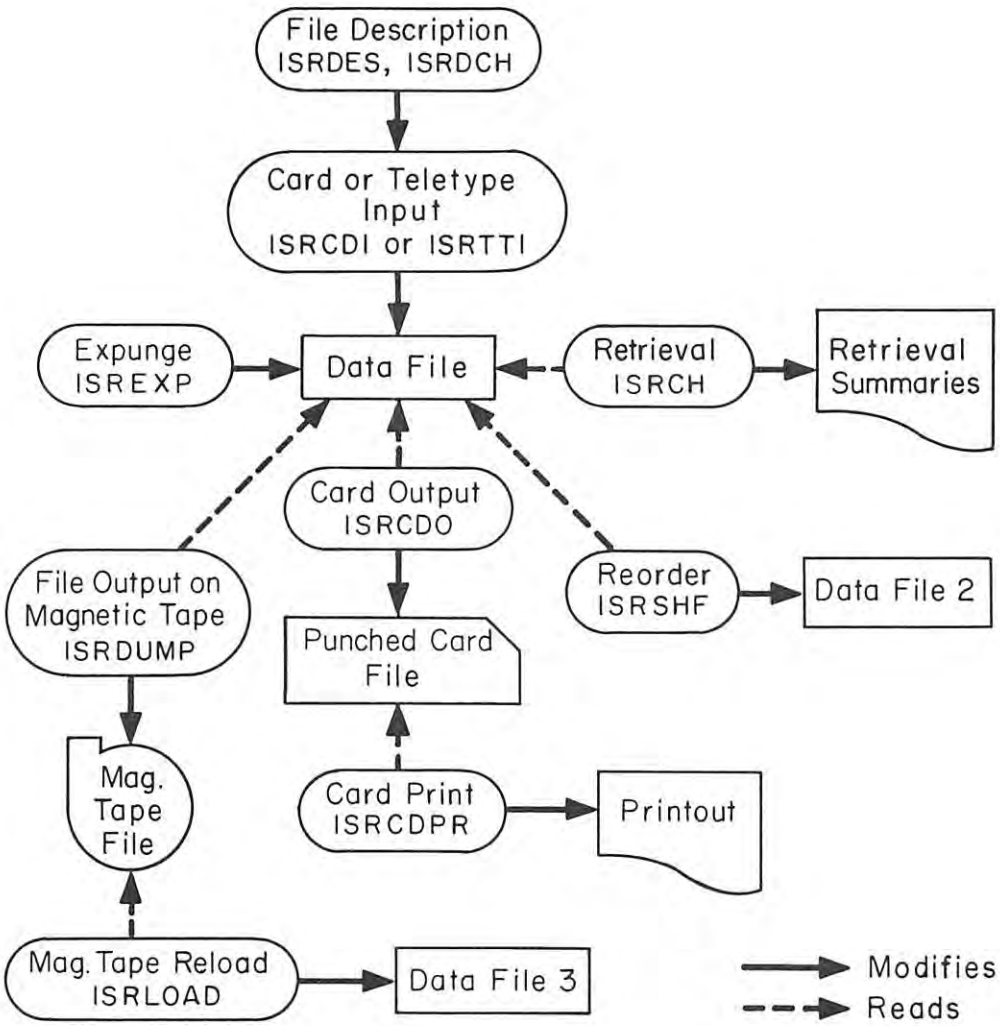
NOTATION

The symbol ⊕ represents an "end of message" (EOM), the character entered into the computer when the ENTER key is depressed. When the ENTER key alone is the full answer to a question, it is referred to as a null entry, sometimes simply called *null*.

The user can interrupt the running of most programs by hitting the key marked SIGNAL or BREAK (on the right of the keyboard). In this Manual, it is referred to as the BREAK key.

In sample typescripts produced by a user at the Teletype, using the ISR programs, the red print indicates the entries typed in by the user.

INFORMATION STORAGE AND RETRIEVAL SYSTEM



This is a graphic representation that shows the interrelationship between the programs and the data of the Information Storage and Retrieval System. The reader is referred to the Table of Contents for specific reference to each Section.

I. FILE-FORMAT DESCRIPTION PROGRAM ISRDES

The File-Format Description Program permits the user to describe the fields of information for records that will be contained in his own data file. As stated in the Introduction, a *field* is the basic logical unit of data that is to be referred to at retrieval; it is characterized by a name and it has one or more values. A field, for example, could refer to employee's sex. A *field name* is a text string that is assigned by the user in the manner detailed below. A field name could be Sex. A *field value* is any configuration of data that has been specified as allowable for a particular field, e.g., MALE or FEMALE.

With the description program, the user names the individual fields (e.g., Sex), states whether each will have one or many values (there will be only one value in each Sex field), syntactically or explicitly defines the permitted input values (MALE or FEMALE are the only allowable entries), and specifies any associated transforms that are to be performed on the input before storage (1 means MALE and 2 means FEMALE). The user also establishes the relationship between the fields (employee's previous jobs and dates of previous jobs are related). Because the data-assimilation process (discussed in Section II) is accomplished by using either a punched-card-input program or a Teletype-input program, all the necessary card-format information must be included in the initial description if input is to come from punched cards.

A. Initial File Description

Initial file description consists of file identification and the basic card-format description. This information must be entered before individual fields can be defined. The initial description

consists of responses to the questions below, which are discussed in detail in the next few pages. Entries typed in by the user are printed in red.

```
ØA FILE NAME: SAMPLE FILE
ØB CONFIDENTIAL CODE: 777
ØC NO. OF CARD TYPES: Ø
```

ØA FILE NAME:

Any title 1 to 30 characters long may be given to a new file. The user's initials (typed in when he initially called the program) will automatically be appended to the title. At the completion of a file description, the file-format information is stored on the Fastrand drum for on-line access, and a file number, to be used for subsequent reference, is printed. This number may change whenever a file is dumped and reloaded from magnetic tape.

ØB CONFIDENTIAL CODE:

If the user wishes to restrict the use of a file, he may assign the file a 3-character confidential code, which must be entered whenever any future access to the file is requested. The code must consist of printing characters, excluding ← and ↑. Nulling this question allows free access to the file.

ØC NO. OF CARD TYPES:

This question should be answered with a Ø or null by all users who expect to build and modify their files solely by Teletype. Subsequently, no other questions involving cards will be asked

throughout the program. (Section I-E explains the use of cards and how to answer the questions involving them when input will come from punched cards—and when ØC is answered with a positive number.)

B. Field Definition

The basic definition of a field of information in a file consists of responses to the following questions, which are discussed more fully in the following pages.

```
1 FIELD NAME: DOA, DATE OF ADMISSION
 1A UNIQUE? YES
 1B TYPE: DATE
 1C SYNTAX DEF. SX
```

If cards are being described (i.e., if question ØC was answered with a number greater than zero), more questions are asked concerning the field definition. See Section I-E.

1. Specifying Field Structure

FIELD NAME:

A field name must be 1 to 6 letters and/or digits in length; the first character must be a letter. There are certain words that are ISRCOMP reserved words and are not acceptable as field names (see Appendix B for the list of reserved words). A second, longer name consisting of any text string may be appended optionally, after a comma. This is considered the long form. In case the user assigns the same field name to more than one field in

his file, the program will print NAME ALREADY USED to notify the user of his error.

UNIQUE TO RECORD?

A YES or NO response is required. A unique-to-record field is one that may assume only one value in any given record. In a hospital patient record, for example, the Patient Name and Sex fields are unique. Other fields may assume one or more values. A Diagnosis field, for example, might be classified as multivalued to accommodate several entries. Whether a field is unique or multivalued affects the structure of the record that will be created. It is also important in the data-assimilation programs, which allow only one value for a unique field in any one record.

Figure 1 illustrates a record in which Name, Sex, and Age are unique fields and Disease and Operation are multivalued fields.

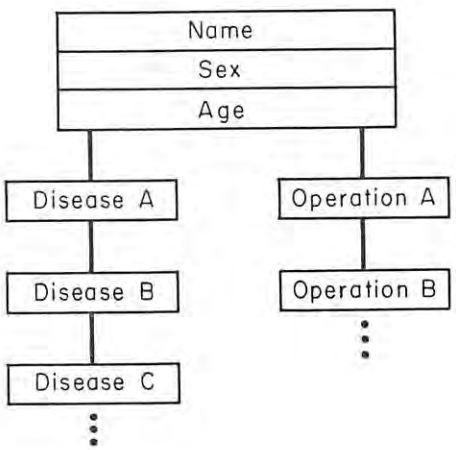


FIGURE 1. Unique and multivalued fields.

DATA TYPE:

One of five data classifications is entered. The data types are shown in Table I. During retrieval, arithmetic operations may be performed on all fields of types I, DN, or D, whereas text operations may be performed on fields of types F or T. All of these fields may have value U (for Unknown) or IND (for Indeterminate). The

value U can be entered only from the Teletype. Indeterminate values are entered either from the Teletype (as null entries or as IND) or from punched cards (as all spaces or from an illegitimate value).

TABLE I. Data types.

T or TEXT Text consists of 0 to 71. keypunched or Teletype characters. Text values are stored in variable-length format after any trailing spaces are removed.
F*n* or FIXED *n* (0<*n*<64.) Fixed *n* consists of *n* keypunched or Teletype characters. Fixed *n* data are truncated or blank-filled to store as *n* chatacters.
I or INTEGER Integer fields are converted to binary numbers for storage. The range of integers is 0 to 131,071. To be accepted syntactically, integer data must consist of a series of digits. Leading and/or trailing spaces are allowed. No signs, decimal points, or commas are permitted.
DN or DECIMAL NUMBER Decimal numbers are converted to floating-point numbers for computer storage with a magnitude restriction of 2.94E-39 to 1.7E38 or 0. Plus signs, minus signs, decimal points, and exponentiation may be used optionally.
D or DATE Dates are stored as cumulative days since January 1, 1849. Input may be of the format month/day or month/day/year, where month and day are one or two digits and year is two or four digits. If no year is given, the year is assumed to be the current year. If two digits are given for year, 19 is assumed to be the century. Standard date abbreviations—T, T*n*, Y*n*—also are acceptable input to a date field. (See Appendix C for standard date abbreviations.)

2. Defining Field Values

SYNTAX DEFINITION:

The syntax-definition question allows the user to specify what values may be entered for the field. He may specify, for example, that a field's entries must be numeric, alphabetic, or a combination. He may list specific allowable entries, e.g., "MALE" or "FEMALE". The user states his definition in a formal language; the elements are described below. A null response to the SYNTAX DEFINITION question gives the definition SX, which allows anything. The definition SX is described more fully in Section I-B2(f) below. U, IND, and null are acceptable Teletype input regardless of the definition. U is not acceptable punched-card input unless the syntax definition and data type allow it.

a. Basic Descriptors

The four descriptors below are used to describe allowable field input.

- A any alphabetic character (A - Z)
- 9 any digit (0 - 9)
- # blank column (in print, space, denoted in this Manual by U)
- X any character (any of the above, as well as punctuation, arithmetic symbols, etc.)

Each basic descriptor defines only one card column or Teletype character; the sequential combination of descriptors defines a

TABLE II. Basic descriptors in syntax definitions.

Definition	Acceptable Input	Unacceptable Input
999	134 ØØØ Ø13	␣134 13A 13 13.4
XX	AB A+	A 2
#A9	␣A6 ␣Q4	A6 ␣AB ␣88 ␣␣A B6␣

set of adjacent columns. For example, 999 defines a 3-digit number; XX defines 2 columns, any characters; #A9 defines blank, letter, digit. (See Table II.)

b. Literal Descriptors

A literal descriptor is an explicit definition of allowable input. Explicit definitions (as distinguished from syntactic) are expressed by using surrounding quotation marks. Literal descriptors may be concatenated. 9"ØØ", for example, permits the first digit to be (Ø-9) but restricts the final two to zeroes. Examples are shown in Table III.

TABLE III. Literal descriptors in syntax definitions.

Definition	Acceptable Input	Unacceptable Input
"DR."	DR.	DR "DR."
"999"	999	134
9"ØØ"	ØØØ 7ØØ	1ØØØ ØØ 9Ø1

c. Two or More Alternatives

If the acceptable input may assume more than one syntactic or explicit configuration, all possible configurations must be listed, separated from one another by semicolons. The semicolon (which may be read as "or") is used to set up alternatives and may be used to permit varying configurations within a single definition. Alternatives are interpreted from left to right, with concatenation preceding OR in interpretation. Square brackets are used to alter the order of interpretation, where necessary. Examples are shown in Table IV.

TABLE IV. OR feature in syntax definitions.

Definition	Acceptable Input	Unacceptable Input
9;A	Ø 1 A B	9A ? 88 ␣B
9;A;"?"	? 1 B	"?" A? +
9[9;A]A	58C 5BC	58BC ABC 587
99;AA#	45 AB␣	4B␣ 45AB 45B
"8;6"	8;6	8 6
["E";"Y";9]99[9"X";"XX"];999"SS"	Y587X 487XX 213SS	512X3 E99SS 9999S 7Ø123

d. Named Definitions

Any syntax definition may be assigned a name, which then may be used in place of future definitions. This is accomplished by following the definition with an equal sign (=) and then the name of the definition enclosed in angle brackets (< >).

Syntax definitions (1) and (2) below are assigned names that are used to construct syntax definition (3). Syntax definitions (4) and (5) are identical; once (4) has been named, the user need type in only the name for (5). When complicated definitions are used for several fields, the named-definition feature is helpful in avoiding typing errors and in reducing the typist's work.

```
(1)  999"-"99"-"9999=<SS#>
(2)  AA99A"."99=<AC#>
(3)  <SS#>;<AC#>
(4)  ["E";"Y";9]99[9"X";"XX"];999"SS"=<DOPC>
(5)  <DOPC>
```

Examples of acceptable and unacceptable input for definition (3) are shown below.

Acceptable	Unacceptable
296-36-92Ø9	2941M.Ø2
TH41M.Ø2	TH6-36Ø2
296-36-6792	A488Q.A2

e. Iterating Commands

The iterating commands are R*n*T for repeat *n* times and S for skip. One allows the description following the command to be repeated a

definite number (*n*) of times, the other as many as it appears. For example, the definition for a Blue Cross number that is seven digits could be specified by 9999999, or, in briefer form, by R7T9. The text description following the command R*n*T is interpreted as though it were written *n* times.

The command S causes the text description following it to be interpreted as many times as acceptable data are found. For instance, a Favorite Ad field might be given the definition S[A;#]. VOLKSWAGEN and AVIS RENT A CAR would be acceptable; ;AND ROVER would not.

A list of data separated by commas is readily specified using the S command. If <ICDA> has previously been defined to accept International Classification of Diseases, Adapted, codes, then the definition S[<ICDA>",]<ICDA> accepts one or more ICDA codes separated by commas.

f. Editing Commands

The ISR System provides the user with certain editing features that enable him to specify that input data are to be edited or changed before being filed. For example, the user may specify, as part of his syntax definition, that cards containing a code of 1 or 2 in the Sex field be converted and filed as MALE or FEMALE, respectively. (If the user wants the conversion to take place on retrieval only, and does not need to have the longer text string stored in the item, he can alternatively use the dictionary feature, which is discussed in Section III-B.)

The three basic editing commands are (i) :—the colon transform, which is used to perform conversions; (ii) D—deletion; and (iii) I—insertion.

The editing commands must be applied with discretion as there are storage maxima related to length of syntax-definition coding. (See Section I-F for further discussion of space limitations.) During input, a datum entry is tested against its syntax definition first, and, if it is found acceptable, the edited result is then validated and interpreted according to data type (specified above).

i. Colon Transform— :

Sometimes it is useful to take an input value, e.g., a space (as on a card), and convert it to another value, e.g., a zero, in the file. The user can specify that he wishes to store a substitute value for an input value by using the colon transform. He specifies a descriptor followed by a colon followed by the substitute value in quotation marks.

The following syntax definition for the Sex field illustrates the use of the colon transform.

```
"1":"MALE";"2":"FEMALE"
```

1 and 2 are literal descriptors and, thus, specify the only acceptable input. Whenever 1 is entered, MALE will be filed as the value of that field; 2 will cause FEMALE to be filed.

If the descriptor is part of an "or" phrase and does not apply to the given input, the colon transform is ignored along with the descriptor. (In the example above, if neither 1 nor 2 is input, the colon-transform command is ignored.)

Use of the colon transform does not interrupt the regular sequence of interpretation. The substitute code is understood to refer only to the descriptor that precedes it—i.e., a basic or literal descriptor, or a bracketed phrase. Note the relationships between the underlined descriptor and transform in the following examples:

```
"87":"EIGHTY-SEVEN"
"8""7":"SEVEN"
99:"SECOND DIGIT"
[99]:"BOTH DIGITS"
["A";"B"]:"A OR B"
```

ii. Deletion—D

The deletion editing command D may be used to suppress filing of input defined by the descriptor that immediately precedes it. The effect is equivalent to using a colon followed by an empty quotation. To file data of the format 99"/"99"/"99 without the slashes, the definition would be 99"/"D99"/"D99.

The more complex example below would accept a date entered in prose format, such as APR.Ø1,1897, and file it as all digits.

```
["JAN":"Ø1";"FEB":"Ø2";"MAR":"Ø3";"APR":"Ø4";
"MAY":"Ø5";"JUN":"Ø6";"JUL":"Ø7";"AUG":"Ø8";
"SEP":"Ø9";"OCT":"1Ø";"NOV":"11";"DEC":"12"]
"."D[["Ø";"1";"2"]9];"3Ø";"31"]","D9999
```

iii. Insertion—I

Data may be inserted into the input field for filing by use of the insertion command I followed by the value to be inserted contained in quotation marks. To file a date of the form 999999 with slashes, the definition would be 99I"/"99I"/"99.

The syntax definition below requires only 6-digit input: two for month, two for day, and two for year. If the first year digit is seven or greater, "18" will be inserted before it (assuming that all years after 69 refer to the nineteenth century), otherwise "19" will be inserted.

```
    9999[I"18"["7";"8";"9"]9;I"19"99]
```

month and day

Note that the left-to-right intepretation of "or" statements permits the use of 99 for the 20th-century alternative instead of ["Ø";"1";"2";"3";"4";"5";"6"]9.

As with the colon transform, the insertion command is interpreted only when the definition with which it is associated is satisfied. (If the definition above were modified by adding ;XXXXXX, input other than six digits would be accepted but would not be edited.)

g. Error Diagnostics

To aid in the correct formulation of syntax definitions, an extensive set of error diagnostics is included in the File-Format

Description Program. Table V illustrates typical errors and the error messages that result.

TABLE V. Description-program error diagnostics.

Definition	Printed Advice
3 "Q"R"	LITERALS MUST BE WITHIN QUOTES.
[9[9;A]	PARENTHESES MUST COME IN PAIRS.
"QR	QUOTES MUST COME IN PAIRS.
9;;A 9; [ ] [9;]	; AND [MUST BE FOLLOWED BY TEXT DESCRIPTION.
9:NINE 9:"NINE 9:NINE:	A STATEMENT WITHIN QUOTES MUST FOLLOW A COLON.
9;:"NINE" AA9I	I AND : MUST BE FOLLOWED BY TEXT IN QUOTES.
99=<TWO DIGITS 99=TWO DIGITS> 99=TWO DIGITS	<NAME OF DEF.> MUST FOLLOW AN EQUALS SIGN.
99=<TWO DIGITS> #99=<TWO DIGITS>	DIFFERENT DEFINITIONS MUST HAVE DIFFERENT NAMES.
99=<TWO DIGITS> <2 DIGITS>9	DEFINITION NAME MUST BE DEFINED.
99<TWO DIGITS	NAME BRACKETS MUST COME IN PAIRS.
R4T SR3TX S;A	RNT AND S MUST BE FOLLOWED BY TEXT DESCRIPTION.
[99:"Ø"]:"1ØØ"	EDITING COMMANDS CANNOT BE NESTED.
[[...[[[[9];9];99];999]...] 21	PARENTHESES LIMITED TO A DEPTH OF SIXTEEN.
Many very long definitions	DEFINITIONS MUST NOT BE UNREASONABLY LONG.
...and a host of others.	

TRY ME:

After the user has entered a legitimate syntax definition, the program types TRY ME, and the user may test various input strings against his definition. The program indicates whether the strings are OK or NOT OK, according to the definition. The user should always test entries for edited fields with the TRY ME because the error detection for prefix errors and double editing occur only on TRY ME. (If one acceptable input string is a prefix of another, the longer format must be specified first to be acceptable. Also, a piece of datum cannot be edited in and then out again.) If editing commands are specified, the program also types out the edited string as it would be filed. A question mark in response to TRY ME returns the user to the syntax question for reconsideration. Some examples follow.

```
1C SYNTAX DEF: 999D
    TRY ME:  456...OK....
     WOULD BE FILED AS: 45
    TRY ME:  45B...NOT OK.
    TRY ME:  SAX...NOT OK.
    TRY ME:  ⊕

2B TYPE: DATE
2C SYNTAX DEF: 99I"/"99I"/19"99
    TRY ME:  Ø1Ø167...OK....
     WOULD BE FILED AS: Ø1/Ø1/1967
    TRY ME:  Ø22967...OK.
     BUT ITS NOT OF THE TYPE INDICATED
     WOULD BE FILED AS: Ø2/29/1967
    TRY ME:  Ø22964...OK....
     WOULD BE FILED AS: Ø2/29/1964
    TRY ME:  ?
2C 99I"/"99I"/19"99
```

C. Record Identification

Another basic function of the description program is to permit the user to define the field or fields that will make one record

distinguishable from another and that will also define the overall structure of his file. A record might be identified, for example, by Disease Name, with other fields in the record being Patient Number, Patient Age, Patient Sex, etc. Each record in that file would contain data on a different disease. All patients with the same disease would be in the same (disease) record. If, instead, the record-identifying field were Patient Number, the entire structure of the file would be altered, for each record would then contain data pertaining to one patient only. All the diseases of the same patient would be in the same (patient) record.

When the user has finished defining all his other fields, he should enter only an EOM to the FIELD NAME question. The program will notify the user that it is then concerned with the RECORD-IDENTIFYING FIELD(S), as shown below. The user is asked to supply the numbers of the unique field(s) that will be used to identify and order the records in the file. If he specifies more than one field, the values are concatenated when comparisons are made. For example, if the record-identifying fields were Age and Sex, then records would exist for values F1 through M103 (assuming that 103 were the age of the oldest male in the file and 1 the age of the youngest female). A maximum of six fields can be designated as the record-identifying fields.

```
7 FIELD NAME: ⊕
8 RECORD IDENTIFYING FIELD(S): 2,1,5
```

D. Field Grouping

When all fields have been defined, the program inquires about groups. If the data in two or more multivalued fields are

logically related, the structure of the file can be made to reflect this relationship. For example, two multivalued fields, Author and University, may be logically related: the first author received his degree from the first university, the second author from the second university, and so forth. The user would then group these two fields. Multivalued fields that are logically related should always be grouped to make possible the most-meaningful retrievals. To designate a group, the user assigns a group name and indicates the numbers of the fields involved as follows:

```
17 GROUP NAME: DS, DOCUMENT SOURCE
 17A FIELDS IN THIS GROUP: 4,5
```

Group names are formed with the same rules that apply to field names.

The fields in a given group must be multivalued fields and they must not be included in any other group. The GROUP NAME question should be nulled if there are no further groups. It will be asked only if there are two or more remaining ungrouped multivalued fields.

```
18 OK?
```

The last question is OK?, which is an opportunity to use ←L or reconsider any answer before the file description is stored for further use. After the user responds with Y or YES, there is a slight pause while the information is stored. Then, the program informs the user of his file *number* (see end of Typescript I, p. 31). It is this number that will be used for future reference to the file.

If the user wishes to alter his file description after he has answered the OK? question with YES, he can call the Description-Change Program (see Section IV-A).

E. Card Usage

ISR records can be created by reading punched cards that contain data in one of several predefined formats. Questions asked at initial file-description time seek to determine criteria for recognizing each of the formats or card types. Further questions asked while specifying the structure of each field seek to determine the columns on each card type that contain values for the field. A detailed description of the card utilization questions, as well as a list of restrictions or precautions pertaining to card usage, is given below.

1. Questions Asked during Initial File Description

```
ØA FILE NAME: SAMPLE FILE
ØB CONFIDENTIAL CODE: 777
ØC NUMBER OF CARD TYPES: 2
 ØC1 DEFINING COLUMNS FOR TYPE #1: 79
  ØC1A SYNTAX DEFINITION: "1"
    TRY ME: 1...OK.
    TRY ME: 2...NOT OK.
    TRY ME: Ø
   ALL OTHER CARDS ARE ASSUMED TO BE OF TYPE #2.
```

ØC NO. OF CARD TYPES:

Because various formats, or types, of cards (e.g., personnel-information cards and year-to-date payroll cards) may be assimilated into one file, each field definition must include the

type(s) of card that will hold the data for that particular field. In order that a field definition refer to card types, an initial description must be made of all card types that may be assimilated into the file. Then, as each field is individually described, the user can specify what card types and on what columns that field occurs. Particular card types are referred to by the program as TYPE #1, TYPE #2, etc. Figure 2 shows how two different card types can be used to make up one file.

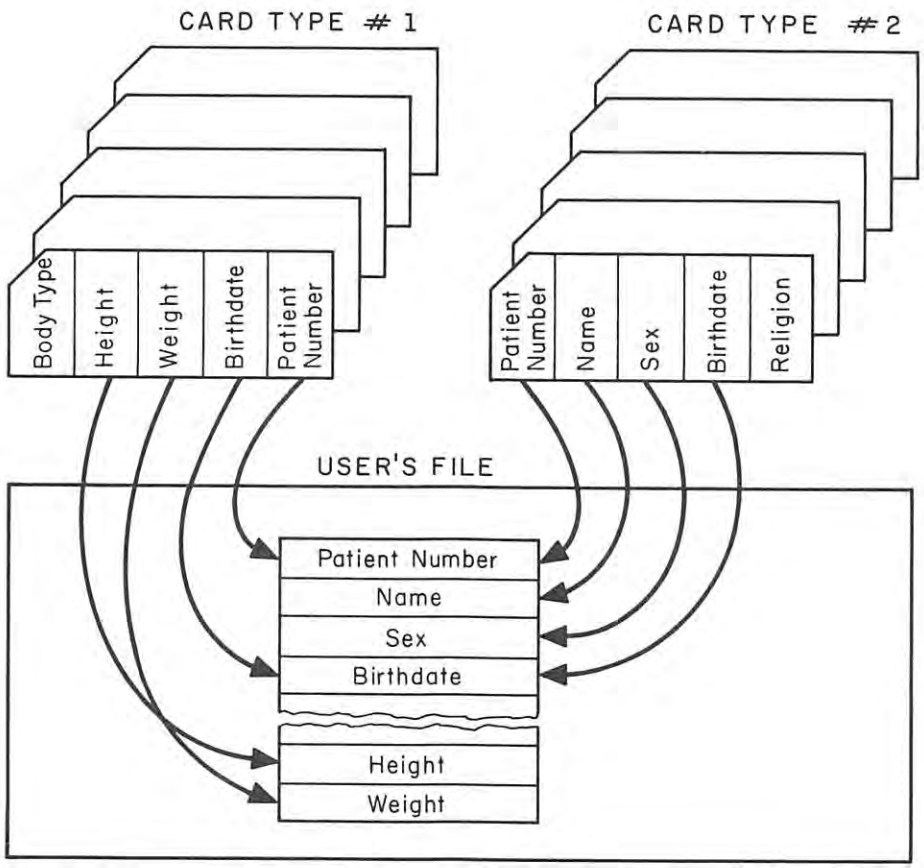


FIGURE 2. Different card types.

ØC1 DEFINING COLUMNS FOR TYPE #1:

ØC1A SYNTAX DEFINITION:

For each type included, a set of columns and their respective identifying codes are indicated. In the preceding program excerpt (p. 18), a file with two card types is defined (ØC). A 1 punched in column 79 defines TYPE #1 (ØC1,ØC1A); if anything else is punched in column 79, the card is TYPE #2.

Continuation cards may be specified implicitly for each basic card type for the case when the data exceed the 80-column capacity of one card. (See Fig. 3.) A continuation card is, in effect, an extension of the card preceding it, rather than a logical entity in itself. If columns 238 - 240, for example, on card TYPE #1 were defined as containing the field Age, and if 240 were the highest specified column number for card TYPE #1, then three 80-column cards (the first of which contains data distinguishing it as TYPE #1) will be interpreted as an extended card of TYPE #1. Columns 78 - 80 of the third one will be expected to contain Age data.

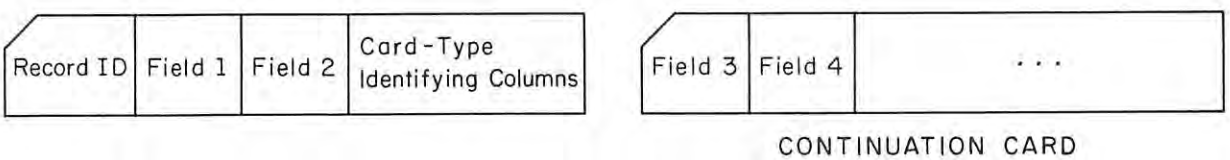


FIGURE 3. An extended card.

All continuation or extension cards must immediately follow the initial card of that type. During the operation of the Card-Input Program, such continuation cards are assumed to follow the first card of the given type; no check for sequence digits is

made. Note that the existence of the number of continuation cards specified is mandatory: one card missing from the middle of a sequence of continuation cards will result in an erroneous assimilation of data. That is, if the continuation-card rules are not followed, some data may be omitted from a record and other data may be filed in the wrong field. Note also that, with the exception of the extended cards, the order of card input does not matter. The card types themselves can be all mixed together, as can the cards for various records. Notice of the number of cards comprising an extended card for each card type is given at the end of the file description.

2. Questions Asked during Field-Structure Description

```
1 FIELD NAME: DOA, DATE OF ADM.
 1A UNIQUE TO RECORD? Y
 1B DATA TYPE: DATE
 1C SYNTAX DEFINITION: 99I"/"99I"/"9999
 1D CARD TYPE: 1
  1D1 COLUMNS: 14-21
 1E CARD TYPE: ⊕
```

CARD TYPE:

This question is asked only if more than one card type is specified (ØC). The user is required to enter the card-type number(s) of the card(s) containing the particular field value. When more than one card type contains the field in question, the different card types may all be specified together, separated by commas, if the field value appears in the same card columns in every type. If the columns are different, the type and columns must be given separately for each type. Input to multivalued fields must come from one card type only. Null is a valid response to the CARD TYPE question.

COLUMNS:

The answer required here is the actual card columns that contain the field values. The answer must be in the format # or # -#, where # stands for any number. Null is a valid response to the COLUMNS question if there is only one card type. Numbers greater than 80. specify continuation cards.

3. Restrictions Relating to Card Usage

Extreme care is recommended in giving card definitions.

(1) All type-defining columns must be within columns 1 - 80 of the card type. They need not be contiguous columns.

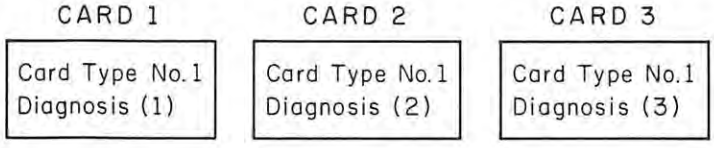
(2) Each record-identifying field must be within columns 1 - 80 of each card type.

(3) No field is permitted to spread over physical card boundaries. For example, a field can be in columns 71 - 80 or 81 - 90 but not in 76 - 85.

(4) All fields in the same group must occur on the same card type.

(5) Multiple field values may be entered as illustrated in Fig. 4. Data are entered sequentially on cards, with one field value per card. The cards must be of the same card type. It is not possible for two incidences of a field value to occur on the same card or its continuation cards.

FIGURE 4. Card input to multivalued fields.



F. Space Restrictions

In most cases, users will not have to be concerned about reaching the space maxima. When they are defining files with many fields, however, they should take especial heed of the various limits.

(1) The title of a file cannot exceed 30. characters.

(2) Combined field and group names must not exceed 600. characters.

(3) A maximum of 50. fields per file is permitted.

(4) The number of groups in one file cannot exceed 15.

(5) All of the fields and groups defined with ISRDES specify the structure of the file's items. The items are allotted space according to the "point" system shown in Table VI.

The error comment TOO MUCH PRIMARY INFORMATION is printed at the end of the program if the total number of points for unique fields plus additional points for groups and ungrouped multivalued fields exceeds 183.

The error comment GROUP CONTAINS TOO MUCH INFORMATION is printed at the end if the total number of points for fields within any group exceeds 381.

In both cases, the OK? question is then reasked. These limitations may be overcome by going back and defining long fields as type text.

TABLE VI. Space allocation in items.

Element	Points
Text field	3
Each character of a fixed field	1
Date	3
Integer	3
Decimal number	6
Group	3 (additional)
Multivalued field (not in any group)	3 (additional)

(6) The length of a syntax definition is limited to 149. points. Table VII shows the point value of each element of a definition.

TABLE VII. Space allocation for syntax definitions.

Element	Points
× A 9 # [A;9;#] or any combination of one each of the above, separated by semicolons and contained within brackets S followed by any of the above	1
;	1 except as included above
[ ]	3 except as included above
=<name>	0
<name>	3 + cost of named definition
S	3 except as included above
R*n*T	1
"literal"	integer part of $\frac{\text{no. of characters}}{3}$ + 3
:	3
I	3
D	6

A detailed example of an actual file description appears on the following pages. Explanatory notes accompany the example.

Notes to Typescript I *(see facing page)*.

ØA	The user requests that the program ask all questions in long form.
ØB	This is a confidential code to ensure the privacy of the file.
ØC1	The computer must look at columns 32 - 35 to distinguish this type from the other type.
ØC1A	The computer must find a slash, two numbers, and a slash in columns 32 - 35 to know that it is TYPE #1. Cards of this type will contain basic patient information.
	TYPE #2 will contain doctor and test data.
1	The user asks HOW the program wants the question answered.
1	The user assigns both a short field name and a fuller (long-mode) name.
1A	There is one unit number per patient admission.
1B	The value of this field is merely a string of characters (as distinguished from an integer,* a decimal number, or a date). It will always be exactly five digits.
1C	The 5-digit number will have a Ø or a 1 as the first digit.
TRY ME:	These questions allow sample input to be tested against the definition given in 1C. OK indicates that the sample input fits the specifications of the definition.
1D	Unit-number values are in columns 76 - 80 on both patient-type and doctor-type cards.

*Since arithmetic operations on Patient Unit # would make no sense, it should not be specified as an integer.

TYPESCRIPT I. File-Format Description Program.

```
4:14 PM 1/8 BBN45-45
ISRDES SGT


FILE-FORMAT DESCRIPTION

 ØA NAME: ←/L
 ØA FILE NAME: DIABETES FILE
 ØB CONFIDENTIAL CODE: BBN
 ØC NUMBER OF CARD TYPES: 2
  ØC1 DEFINING COLUMNS FOR TYPE #1: 32-35
   ØC1A SYNTAX DEFINITION: "/"99"/"
      TRY ME:  /2A/...NOT OK.
      TRY ME:  9/8/...NOT OK.
      TRY ME:  /Ø24...NOT OK.
      TRY ME:  /Ø5/...OK.
      TRY ME:  ⊕
    ALL OTHER CARDS ARE ASSUMED TO BE OF TYPE #2
1 FIELD NAME: ←HOW ASSIGN A 1-TO-6-CHARACTER NAME ACCEPTABLE
TO ISRCOMP; IF DESIRED, ADD A COMMA AND A LONG-MODE NAME.
1  PATNO, PATIENT UNIT #
 1A UNIQUE TO RECORD? ←HOW TYPE YES IF THE FIELD HAS ONE
VALUE PER RECORD.  NO IF IT HAS A LIST OF VALUES.
 1A  Y
 1B DATA TYPE: ←HOW TYPE INTEGER, DECIMAL NUMBER, DATE, TEXT,
OR FIXED N (WHERE N = NO. OF CHARS.).
 1B  F5
 1C SYNTAX DEFINITION: ["Ø";"1"]9999
      TRY ME:  21114...NOT OK.
      TRY ME:  111111...NOT OK.
      TRY ME:  11111...OK.
      TRY ME:  14567...OK.
      TRY ME:  Ø789Ø...OK.
      TRY ME:  ⊕
 1D CARD TYPE: ←HOW ON WHICH CARD TYPE(S) CAN THIS FIELD BE
FOUND?
 1D  1,2
  1D1 COLUMNS: 76-8Ø
2 FIELD NAME: PN, PATIENT NAME
 2A UNIQUE TO RECORD? MAYBE FIX
 2A Y
 2B DATA TYPE: T
 2C SYNTAX DEFINITION: ⊕SX
      TRY ME:  ⊕
```

Notes to Typescript I (continued).

3B The field is a date, and, as a result, date arithmetic may be done at retrieval time. It must be *filed* in standard format.

3C The definition converts the input to standard format.

5A There may be more than one doctor per patient admission.

5D Doctor is found only on TYPE #2 cards.

TYPESCRIPT I (continued).

```
 2D CARD TYPE: 1
  2D1 COLUMNS: 1-15
 2E CARD TYPE: ⊕
3 FIELD NAME: BD, BIRTHDATE
 3A UNIQUE TO RECORD? Y
 3B DATA TYPE: DATE
 3C SYNTAX DEFINITION: 99I"/"99I"/"["I"18"["7";"8";"9"]9;I"19"99]
     TRY ME:  Ø1Ø165 ...OK.
      WOULD BE FILED AS:  Ø1/Ø1/1965
     TRY ME:  Ø1Ø198 ...OK.
      WOULD BE FILED AS:  Ø1/Ø1/1898
     TRY ME:  Ø229ØØ ...OK.
      WOULD BE FILED AS:  Ø2/29/19ØØ
      BUT IT'S NOT OF THE TYPE SPECIFIED.
     TRY ME:  12X344 ...NOT OK.
     TRY ME:  ⊕
 3D CARD TYPE: 1
  3D1 COLUMNS: 4Ø-45
 3E CARD TYPE: 2
  3E1 COLUMNS: 21-25
4 FIELD NAME: DOA, ADMISSION DATE
 4A UNIQUE TO RECORD? Y
 4B DATA TYPE: D
 4C SYNTAX DEFINITION: 99"/"99"/196"["5";"6"]
     TRY ME:  12/12/1966 ...OK.
     TRY ME:  12/12/1955 ...NOT OK.
     TRY ME:  12/32/1966 ...OK.
      BUT IT'S NOT OF THE TYPE SPECIFIED.
     TRY ME:  ⊕
 4D CARD TYPE: 1
  4D1 COLUMNS: 3Ø-39
 4E CARD TYPE: ⊕
5 FIELD NAME: DR, DOCTOR
 5A UNIQUE TO RECORD? N
 5B DATA TYPE: T
 5C SYNTAX DEFINITION:  SX
     TRY ME:  ⊕
 5D CARD TYPE: 2
  5D1 COLUMNS: 3Ø-44
6 FIELD NAME: FBS, BLOOD SUGAR
 6A UNIQUE TO RECORD? N
 6B DATA TYPE: DECIMAL NUMBER
 6C SYNTAX DEFINITION: ⊕ SX
```

Notes to Typescript I (continued).

7 Nothing was inserted; therefore, all fields have been described.
8 The record or episode is defined by a patient unit number.
9A Doctor and Blood Sugar are logically linked into one group. During retrieval, "a certain doctor and a certain blood sugar" can be interpreted as "a certain blood sugar and a certain doctor who ordered it."

Note that the number assigned the file for future reference is 16.

TYPESCRIPT I (continued).

```
     TRY ME:  23.4...OK.
     TRY ME:  23..4...OK.
      BUT IT'S NOT OF THE TYPE SPECIFIED.
     TRY ME:  -2.4E5...OK.
     TRY ME:  ⊕
 6D CARD TYPE: 2
  6D1 COLUMNS: 7Ø-75
7 FIELD NAME: ⊕
8 RECORD-IDENTIFYING FIELD(S): 1
9 GROUP NAME: BTESTS, BLOOD TESTS
 9A FIELDS IN GROUP: 5,6
1Ø OK TO FILE? Y

 16 DIABETES FILE (SGT) 4:14 PM 1/8/1968

4:52 PM
```

II. DATA ASSIMILATION

Once a file format has been specified, two programs are available to assimilate data into the file. One program, Teletype Input (ISRTTI), accepts data entered online via Teletype; the other program, Card Input (ISRCDI), accepts data from magnetic-tape images of punched cards. In addition to entering data into a new file, these programs may be used to update files that already contain data. A file may accept data interchangeably from cards or Teletype.

Users are advised to enter initially only a few records into a new file, and then to do some retrievals, in order to make sure that the file as they described it is exactly as they meant it to be. It is wisest to enter all of the records only after making sure that the description is accurate.

A. Teletype-Input Program — ISRTTI

ISRTTI is useful for correcting or making minor additions to previously assimilated large files, as well as for developing new files.

Like all of the ISR programs, except ISRDES, the program begins by asking the file number of the file that the user wishes to use (see below). The number is the one assigned to the file by the File-Format Description Program. The program prints the file name, initials of the originator, and time and date of file creation or last update as verification. (Possible errors and the respective error comments are discussed in Appendix D.) If the file was given a confidential code by the user who described the

file initially, the confidential code is requested. The user is denied entry to the file until he supplies the code.

```
1 FILE NUMBER: 16  DIABETES FILE (SGT)  4:14 PM 1/8/1968
1A CONFIDENTIAL CODE: NBB FIX
1A BBN
```

After successfully gaining entry to a file, ISRTTI prints a field name (that specified when using ISRDES) and the user types in the appropriate field value; this is done for each field in turn until a complete record is compiled. (See Typescript II, p. 41.) The record is then added to the file and the process is repeated.

The order in which field values are requested is determined by the order in which fields were defined with the Description Program, with the following exceptions. Those fields specified as the record-identifying fields will be asked first. Grouped multi-valued fields will be asked last.

1. New Records

To place a new record in a file, the user must supply values for all of the record-identifying fields. (Null entries are not permitted and will cause the program to halt.) The user may then place data in any or all of the remaining fields. Null entries may be made to nonidentifying fields and will cause IND to be typed by the program. These entires will be stored as Indeterminate and will appear in all future printouts as IND. The value U may be entered to indicate an Unknown field value. These entries will appear as U in all future printouts.

Unique fields will accept one field value, and the program then proceeds to the next field. Multivalued fields, on the other hand, can accept up to 62. field values. To terminate input to a multivalued field, the user types the character -. The program will then proceed to the next field.

Groups consist of two or more multivalued fields (which were defined as being a group). One entry is requested for each multivalued field within the group before the first multivalued field is repeated. For example, a group Amount Order (Amtord) consisting of Quantity (Quant), Price, and Date would be asked in the following order:

```
4 AMTORD
  4A1 QUANT
  4A2 PRICE
  4A3 DATE
  4B1 QUANT
  4B2 PRICE
  4B3 DATE
```

When a new value is entered for the first field in a group, Indeterminate values are automatically established for all other fields in that group at the same subscript level. When data are entered for the other fields, the values are changed from Indeterminate.

Groups too are terminated by typing the character -. If - is entered for any but the first multivalued field in a group, null entries are assumed for all subsequent fields at that level. For example, for the above group Amtord with fields Quant, Price, and Date, the entries in the two examples below are equivalent.

```
4 AMTORD
  4A1 QUANT 12
  4A2 PRICE -
```

```
4 AMTORD
  4A1 QUANT 12
  4A2 PRICE ⊕IND
  4A3 DATE ⊕IND
  4B1 QUANT -
```

A record may be terminated in two ways. When the last field has been supplied with a value or terminator, the program will automatically ask the first record-identifying field for the next record. Alternatively, if the user wishes to proceed to the next record without answering all fields, he may type ←NR (next record). This will cause nulls or Indeterminate values to be supplied to all remaining fields in the record. The first record-identifying field for the next record will then be asked. ←NR may be used only after all record-identifying field values have been entered.

2. Changing and Deleting Records

After a record has been filed, it is possible to both examine and change the contents of any nonidentifying field. When all identifying fields have been entered, the program determines whether a record with the same identifier already exists. If so the text (OLD) will be printed to notify the user. Field values are requested as in the case of new records.

To change a field, the user simply enters new data when that field question is asked. It is important to note that the record in the file is not changed until all fields have been requested and answered. Halting the program (with ←-) while altering an old record will leave that record unchanged in the file. A null entry to a field that has previously been filled leaves the contents unchanged. Thus, entering ←NR, after making all desired

changes to a record, leaves the rest of the record unchanged and causes the program to proceed to the next record. To change a value already in a record to Indeterminate, the user must enter IND.

a. Print

The ←P (print) and ←PA (print all) features may be used to examine the contents of an old record as it appears in the file. The ←P and ←PA features may only be used after (OLD) has been printed. ←P instructs the program to type out the contents of the field currently being requested, e.g.,

```
4 DOCTOR ←P  FREUD, S.
4
```

The field is reasked without the field name being printed, and then the program waits for the user's entry—either null to leave the value as it is or a new value.

←PA is equivalent to typing ←P to all fields starting with the field currently being asked, e.g.,

```
3 NAME ←PA  JONES, J.
3 ⊕
4 D.O.B.    3/21/42
4 3/23/42
5 OPERATIONS
 5A PULMONARY EMBOLECTOMY
 5A ⊕
 5B MITRAL COMMISSUROTOMY
 5B ⊕
 5C FIX
 5C ←P
 5C EXODENTIA
 5D -
```

The contents of each field are typed out. The field is reasked and the program waits for the user's entry. ←P terminates the ←PA feature.

With multivalued fields, such as 5 above, it is possible that a print request will be made for a field for which no previous entry has been made (5C above). When this occurs, FIX is typed and the field reasked. (In the above example, the user then terminated the PA command by entering ←P.)

Both the ←P and ←PA features may be used to skip over some fields and examine a field or fields occurring later in a record. This is done by appending the field question number desired to either ←P or ←PA, e.g.,

```
3 NAME ←P5B
5 OPERATIONS
 5B MITRAL COMMISSUROTOMY
 5B ⊕
 5C
```

Similarly:

```
3 NAME ←PA5B
5 OPERATIONS
 5B MITRAL COMMISSUROTOMY
 5B ⊕
 5C EXODENTIA
 5C
```

When a ←P*n* or ←PA*n* command is given, a null entry is made to the current field and all of the skipped fields between it and the field specified by the *n*. If the field *n* specified in the ←P*n* or ←PA*n* command has no previously entered value, FIX will be typed and the current field reasked.

b. Append

Another feature, ←A (append), is provided to allow the user to find the next available level of multivalued fields, e.g.,

```
5 OPERATIONS
 5A ←A
 5D
```

In addition, ←A*n* may be used to skip over some fields and to append to a multivalued field, e.g.,

```
3 NAME ←A5
5 OPERATIONS
 5D
```

←A may be used at any point within a multivalued field, i.e.,

```
5 OPERATIONS
 5A ←P PULMONARY EMBOLECTOMY
 5A ⊕
 5B ←A
 5D
```

←A, as well as ←P, will terminate the ←PA feature.

c. Delete

The ←DEL (delete) feature is provided so that records may be deleted from a file. Typing ←DEL in answer to any question after the record has been identified as (OLD) causes the program to delete the record from the file and proceed to the next record—e.g.,

```
1 ID.NO. Ø4763 (OLD)
2 SERVICE ←DEL

1 ID.NO.
```

3. Error Comments

Two classes of errors are detected by ISRTTI: computer-system failures and erroneous data.

a. Computer-System Failures

In the event of machine failure, the program prints an error message (SAVE THIS PAGE AND NOTIFY BBN IMMEDIATELY) followed by the termination time, and then it halts.

b. Erroneous Data

A datum entry may be rejected for one of the two following reasons.

(1) The field does not fit the syntax definition specified by the user when establishing his file with the File-Format Description Program.

(2) The field is not consistent with the system data-type requirements (i.e., date, integer, decimal number—see Table I, p. 5) as specified by the user, although the field value is syntactically correct by the definition. (For example, if the syntax definition for a date value is 99/99/9999, the data value Ø1/32/1964 would pass the syntax definition but would fail the data-type legality check.)

When the error is detected, the program prints FIX and the question is reasked. The program will not proceed until an acceptable entry is made.

A detailed typescript from the running of the Teletype-Input Program follows. In this example, ISRTTI was used to enter records into the file created by ISRDES in Typescript I, p. 27.

Notes to Typescript II *(see facing page)*.

1	16 is the file number assigned to the file at the end of ISRDES.
1	1ØØ94 is the patient's unit number (which is also the record-identifying field—see ISRDES question 8).
4	Date entry did not fit the definition (see ISRDES question 4C).
5A1-5D2	The patient record has four Doctor entries and four Blood Sugar entries. They are grouped. (See ISRDES questions 5A, 6A, 9A.)
5E1	- terminates the entry of multivalued field values.
1	A new patient record is started.
3	The user realized that he entered the wrong unit number and asked to return to 1.
3	His birthdate is not known.
5A1	He has no entries for the Blood Tests group.
1	This is an update of Peterson's record.
2	The user wants to append values to the Blood Tests group and asks to go directly to the next free place in the group. The other fields, Peterson's name, unit number, admission date, and birthdate are left unchanged.
5E1-5F2	Two new Blood Sugar values are appended. Since no corresponding Doctor entries were made, the Doctor values are entered as Indeterminate.

TYPESCRIPT II. Teletype-Input Program.

```
1:16 PM 1/9 BBN22-22
ISRTTI SGT


TELETYPE INPUT

1 FILE: ←/L
1 FILE NUMBER: 16  DIABETES FILE (SGT)  4:14 PM 1/8/1968
 1A CONFIDENTIAL CODE: NBB FIX
 1A  BBN

1 PATIENT UNIT # 1ØØ94
2 PATIENT NAME PETERSON, TIM
3 BIRTHDATE Ø81422
4 ADMISSION DATE Ø1/15/64 FIX
4  Ø1/15/1965
5 BLOOD TESTS
  5A1 DOCTOR RYDER, I.
  5A2 BLOOD SUGAR 12Ø
  5B1 DOCTOR  JAMISON, P.P.
  5B2 BLOOD SUGAR 1Ø4
  5C1 DOCTOR ←C5A1 RYDER, I.
  5C2 BLOOD SUGAR 52
  5D1 DOCTOR ←C5A1 RYDER, I.
  5D2 BLOOD SUGAR 43
  5E1 DOCTOR -

1 PATIENT UNIT # 11284
2 PATIENT NAME FREDRICKS, F.
3 BIRTHDATE ←1
1 PATIENT UNIT # 11284 12184
3 BIRTHDATE ⊕IND
4 ADMISSION DATE Ø6/2Ø/1965
5 BLOOD TESTS
  5A1 DOCTOR -

1 PATIENT UNIT # 1ØØ94    (OLD)
2 PATIENT NAME ←A5
5 BLOOD TESTS
  5E1 DOCTOR ⊕IND
  5E2 BLOOD SUGAR 1ØØ
  5F1 DOCTOR ⊕IND
  5F2 BLOOD SUGAR 124
  5G1 DOCTOR -
```

Notes to Typescript II (continued).

1	This does not fit the syntax definition (see ISRDES question 1C).
1-5G1	This creates a record for Semple.

1	This is a correction of Semple's record.
2	The user asks to skip to question 4 and print the value of the Admission Date field.
4	The user corrects Semple's admission date.
5A1	The user goes on to the next record. Only the Admission Date field was changed in this record.
1	Nothing was entered, so the program halts.

TYPESCRIPT II (continued).

```
1 PATIENT UNIT # 16Ø7R FIX
1  16Ø74
2 PATIENT NAME SEMPLE, JOHN
3 BIRTHDATE 121299
4 ADMISSION DATE Ø6/25/1966
5 BLOOD TESTS
  5A1 DOCTOR JONES, J.P., III
  5A2 BLOOD SUGAR 22Ø
  5B1 DOCTOR RYDER, I.
  5B2 BLOOD SUGAR 96
  5C1 DOCTOR RYDER, I.
  5C2 BLOOD SUGAR 135
  5D1 DOCTOR MATHEWS
  5D2 BLOOD SUGAR 125
  5E1 DOCTOR RYDER, I.
  5E2 BLOOD SUGAR 187
  5F1 DOCTOR RYDER, I.
  5F2 BLOOD SUGAR 2Ø4
  5G1 DOCTOR -

1 PATIENT UNIT # 16Ø74  (OLD)
2 PATIENT NAME ←P4
4 ADMISSION DATE 6/25/1966
4  Ø1/25/1966
5 BLOOD TESTS
  5A1 DOCTOR ←NR

1 PATIENT UNIT # ⊕

1:26 PM
```

B. Card-Input Program—ISRCDI

The user who has already collected data in the form of punched cards is provided with the program ISRCDI for entering the data into his file. It should be noted that ISRCDI can only change or update fields that on previous assimilations had no value entered. Multivalued field values are appended to any already existing values for the field. The Teletype-Input Program, ISRTTI, must be used to change a previously entered field value. ISRCDI is therefore used to create new records and to supplement but not change existing data in old records.

To be assimilated into an ISR file, the punched-card images must be recorded on magnetic tape. Once a card file is put on tape, data may be assimilated into any number of files. (Since the amount of drum space available for file storage is limited, arrangements should be made in advance for the processing of very large data files.)

Acceptable ISRCDI input is in high-density tape format (556 bpi) as follows. The first 80 characters of each tape record must be binary-coded decimal (BCD) images of cards punched in Hollerith code. An interrecord gap must separate each card image. An end-of-file mark must precede each file on a magnetic tape, including the first. Three end-of-file marks must be used to terminate a tape.

Magnetic tapes are maintained in the Computer Room. When new tapes are received, they are assigned a reel number. The tape number is used to identify the tape whenever it is used.

1. Program Use

The program requires responses to five questions, as shown below.

```
12:38 PM 1/9 BBN-46-46
ISRCDI SUR

CARD INPUT

1 ISR FILE: 16  DIABETES FILE (SGT)  1:16 PM 1/9/68
 1A CODE: BBN
2 REEL NO: 17
3 FL-ON-TP: 2
4 CD-IN-FL: 247
5 UPD-ONLY? N
```

ISR FILE:
CODE:

These questions are the same as in the other ISR programs (see p. 33).

REEL NO:

Question 2 asks for the number of the magnetic tape that holds the card images to be assimilated. This is the tape-reel number that was assigned to it when it was received in the Computer Room.

FL-ON-TP:

This question asks for the ordinal number ($\geq$ 1) of the card file on the tape that is to be assimilated.

CD-IN-FL:

For the tape file specified above, the user is now requested to enter the ordinal number of the first card to be read for this assimilation. If he wishes also to specify how many cards he wants assimilated, he may enter an end card as well as a first card, e.g., 1-1ØØ. The CD-IN-FL: facility makes it possible to assimilate a tape file in several short runs.

If the last card specified is part of an extended card (see p. 20) and not the end of the extended cards, the program will enter the data from all the continuation cards. Then, when this is completed, the program will print the number of the last card entered. For example, if 1-1ØØ were the answer to the CD-IN-FL: question and cards 100 and 101 were the continuation cards of card 99, 101 cards would be processed.

UPD-ONLY?

This question provides the option of selecting for assimilation a subset of the records from a tape. If the answer is YES, only those cards with data concerning records already in the file (i.e., with record-identifying fields identical to those already in the file) will be assimilated. This option permits the user with a small file to assimilate only pertinent update cards (containing data to append) from a large set of cards that may refer to many other records not in his particular file. For example, a file may contain data on all of the cancer patients in a hospital. If it is desirable to add laboratory data to these records, all of the laboratory cards could be run through the ISRCDI program because, with the UPDATE ONLY feature, only the data pertaining to the cancer patients would be processed.

After all of the questions have been answered, a message is generated in the Computer Room to notify the computer operator to mount the tape. When the tape is mounted, it is positioned to the beginning of the specified file and to the specified card number in that file. The program generates a carriage return and linefeed on the user's terminal at the start and at the finish of this activity. (The user might want to check with the computer operator if the second carriage return and linefeed does not occur within 5 minutes.)

Then the program runs until an end-of-file mark is encountered, printing any necessary error messages and making an occasional "cough" to let the user know that the assimilation process is going on. The message JOB IS COMPLETE - XX CARDS READ OR SPACED and the termination time are printed, and the program halts.

Blank or invalid field values are stored as Indeterminate values for unique fields; they create no value for multivalued fields unless the fields are grouped. In that case, a valid field value for one field in a group establishes corresponding Indeterminate values at that level for all other fields in the group whose values are blank, invalid, or not available on the same (extended) card. If all fields within a group have invalid values, a new level is not established. If no card of a certain card type is present for an item, all the fields from that card type are treated as blank entries.

The user may interrupt the assimilation by depressing the BREAK key. The program then prints the number of the next card to be processed and asks CONTINUE?. It will continue if the user enters YES, halt if he says NO. The interruption facility makes it possible to assimilate a tape file in several short runs.

2. Error Checking

In ISRCDI, the same error checks (i.e., syntax definition and data-type compatibility) are made as for the Teletype-Input Program. With ISRCDI, however, the error comments are printed while the program continues to run. When the card assimilation of data is complete, the user can consult the error comments and then use the ISRTTI program to edit and correct field errors. [In addition, the ISRCH program (see Section III) or ISRCOMP (see separate manual) may be called to scan the file for unwanted or absent values. For example, the field Age might be searched for values less than zero or greater than 100. Any records found with such an Age would be candidates for correction.]

There are three types of error comments, as discussed below. All error printouts give the ordinal tape-file number of the card, the number of the first column of the rejected field, and the field value on the card. As with the Teletype-Input Program, field values may be rejected if the field does not fit the syntax definition specified by the user when establishing his file (with the File-Format Description Program), or if the field is not consistent with the system data-type requirements. In addition, if a field designated as unique to the record has been found to differ on cards belonging to the same record or in already existing records, all values after the first will be treated as errors. For example, if one card specifies a patient's age as 31 and another card specifies it as 32, the Card-Input Program resolves the inconsistency by storing the first value encountered for this unique field and prints an error message when it encounters the second. The user may insert another value later by using the facilities of the Teletype-Input Program.

In each of the above cases, the erroneous or inconsistent field value is ignored and the rest of the card is assimilated, provided that the record-identifying field is syntactically correct.

If the record-identifying field is in error, the entire card is ignored; the effect is the same as in the case where a particular card is not present for an item. Skipping a whole card may generate a series of Indeterminate codes for unique fields found only on that card type. For example, suppose that card TYPE #1 contained only three fields—(1) record-identifying field, (2) geographic-location code, and (3) income code—and that the latter two were not replicated on other card types. If the record-identifying field is not acceptable, the entire card is rejected. As a result, that record in the new file will have Indeterminate codes for geographic location and income. If there is only one card type, no record is created.

Syntax or data-type errors for record-identifying fields cause messages of the following type:

```
INVALID RECORD-ID ON CARD 134 COL. 17: "12Q3"
```

The card number, beginning column, and contents of the offending field are printed. The rest of this card and its continuation cards are then ignored because there is no known place to file the data. Blanks are valid for record-identifying fields only if blanks are acceptable syntactically and by data type.

Syntax or data-type errors for non record-identifying fields cause messages of the following type:

```
INVALID FIELD ON CARD 138 COL. 19: "1Ø5L"
```

Only the particular field found invalid is ignored; the rest of the card is processed normally.

Unique field values on cards differing from values already stored in a record (possibly by previous cards during this run) cause messages of this type:

VALUE CONFLICT ON CARD 241 COL. 3Ø: "BOSTAN"

The old value is left intact. No message is generated if the values agree or the card contains blanks for the field.

C. Space Restrictions

There are no space restrictions on either record length or number of records in a file. Multivalued field values after the 64th are perfectly valid, although they cannot be changed or added to by the Teletype-Input Program (because of a limit to the number of questions that can be asked).

III. INFORMATION RETRIEVAL AND ABSTRACTION

Once data have been entered into an ISR file, the ISRCH program may be used to retrieve the data. The ISRCH program is a general-purpose retrieval program that allows each user to specify different retrievals each time that the program is called. After the user designates which file he is concerned with and specifies the criteria for his retrieval, the program reads the records in the file, selects the appropriate records, and prints the report requested by the user.

Often, a user wishes to store short coded entries in his file. At retrieval time, he may want the ISRCH program to print these coded entries, as well as their text equivalents. The program ISRDIC is available for users to create and maintain dictionaries of codes and associated text strings. These dictionaries—independent of any file—are created so that at retrieval time the user can specify which fields are to be treated as codes and can have the associated text strings retrieved from the appropriate dictionary.

The ISRCH program is discussed in Section III-A; Section III-B is concerned with ISRDIC.

A. Search Program—ISRCH

The retrieval program ISRCH can perform two major functions on files:

(1) print individual record summaries and

(2) compile and print statistical crosstabulations.

1. Definitions

Some definitions of the basic terms and concepts involved in using the ISRCH program will provide a background for the discussion that follows.

Population: a subset of records in a file, specified in the retrieval request as the records of current interest. When no subset is specified, the entire file is treated as the population of interest.

Sample: a subset of records within the population that have characteristics defined at retrieval time. The characteristics are described by specifying row and column descriptors to be logically ANDed together in determining which records are included in the sample. Various interrecord sums of a field may be accumulated for the records in the sample and displayed by means of a two-dimensional matrix.

Original Fields: those fields defined at the time of file description.

Derived Fields: new fields defined at each retrieval session by applying arithmetic operations, possibly conditionally, to original fields or previously derived fields.

Descriptor: a statement to describe selected records on the basis of certain characteristics of particular fields. An example of a descriptor statement for a record containing the field Age is AGE>4Ø.

Interrecord Summary Information: computed numerical field values obtained by adding values of an original or derived field over some sample of the population.

Group of Fields: fields of multivalued information declared as logically interrelated at the time of file description. The field values are stored in the record, with each set of corresponding field values identifiable by the same subscript. Field grouping, based on the logical nature of the data, makes it possible for multivalued fields from the same group to be referenced simultaneously in a single search descriptor at retrieval time.

Table VIII illustrates two groups of multivalued fields. The Professors group implies a logical correspondence between the first Office No. and first Prof; Office No. and Prof. may be referenced by the same descriptor. The first Course is unrelated to the first Prof. and therefore they may not be referenced by the same descriptor.

TABLE VIII. Grouped fields.

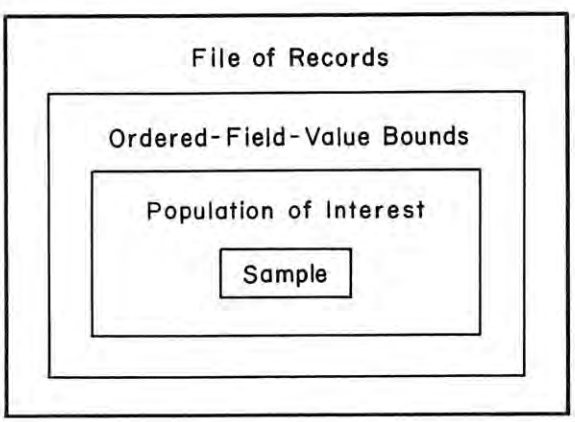
GROUP 1: Professors			GROUP 2: Exams		
Subscript	Office No.	Prof.	Subscript	Course	Date of Final
1	H123	JAMES	1	ALGEBRA 41	1/17/1968
2	M141	SMITOR	2	PHILOSOPHY 36	1/19/1968
3	H231	WHITE			

Indeterminate Field Values: occur when a field does not contain a legitimate field value as specified in the field syntax definition or data-type requirement. Absence of a valid field value from a punched card source or a null or IND entry from Teletype input may result in the storage of an Indeterminate code. On printout, such field values always appear as the text IND.

With multivalued fields, Indeterminate values may be generated when one member of a grouped set of data is missing or unknown. For example, if Name of Siblings and Age of Siblings are defined as grouped multivalued fields and no Age value is present for the second sibling, this field would be classed as Indeterminate.

Name of Siblings	Age of Siblings
JENNIFER	8
MICHAEL	NOT SPECIFIED—INDETERMINATE
THOMAS	13

2. File Partition



The ISRCH program abstracts a particular subset of the given file and performs certain operations on that subset. The user describes his desired subset by specifying restrictions that are to be applied to a search of his

FIGURE 5. File partition.

data. The restrictions are applied in a hierarchical fashion to resolve populations and samples. The partition of a file of records may be diagrammed as in Figure 5. The samples may overlap one another.

Table IX outlines the restrictions that the user may ask to be applied and the order in which they are interpreted.

TABLE IX. Application of population restrictions.

File
Ordered-field-value bounds
Maximum number records
Descriptors
Maximum number records in population
Sample descriptors
Maximum number records in sample

a. Ordered-Field-Value Bounds

A file of records is stored in collational (similar to alphabetical) order according to the values of the record-identifying field(s). A continuous sequence of records in a file may be isolated between an upper-bound field value and a lower-bound field value for the field(s) by which the file is ordered. Users may use the ordered-field-bounds feature to state such field values. Utilization of the ordered-field bounds whenever possible may reduce considerably the length of time required for searches because

the first and last record in sequence may be determined rapidly without reading any actual data records. If no ordered-field restriction is given, a linear search of the entire file of records is initiated.

b. Numeric Limits

Numeric bounds may be set, if desired, to limit the number of records searched, the maximum number of records in the population, and the maximum number of records falling into the sample. The search procedure terminates when any maximum is reached.

c. Boolean Descriptor Constraints

Generalized relational expression (e.g., SEX="MALE" AND AGE>4Ø) that pertain to field values may be entered to define either a population or (multiple) sample(s). Descriptors are discussed in detail later in this Section (page 66).

The order in which the restrictions described above are applied is important. First, any ordered-field constraints are looked up; then, numeric limits are checked; and, finally, Boolean descriptors are evaluated. Any of these methods may be used either alone or in combination to instruct the program as to what conditions to consider in order to find the desired records. If no restrictions are specified, the entire file is treated as being of interest.

3. Use of ISRCH

The ISRCH program is discussed below, question by question. Some preliminary questions are followed by the main series of questions, beginning again with question 1.

a. Preliminary Questions

```
1 FILE NUMBER:
 1A CONFIDENTIAL CODE:
```

The answers required for the user to gain entry to his file are the same as for the other programs (see p. 33).

Magnetic-Tape Search. It is possible to use the retrieval program on a file that has been dumped onto magnetic tape with the Magnetic-Tape Dump Program (see Section IV-E1), as well as for the more normal case where the data file is stored on the Fastrand drum. (The ISRTTI and ISRCDI programs automatically store data on the Fastrand drum.) A user may choose to dump his file onto magnetic tape because he has many files or one very large file and he does not wish to use such a large amount of space on the drum. The Search Program, in most cases, will take no longer and often less time to retrieve from a file on tape.

In answer to the FILE question, the user would respond with the word TAPE, the tape reel number and number of the file on the tape, e.g.,

```
1 FILE NUMBER: TAPE 562, 3
```

(See p. 45 for explanation of tape reel numbers and tape file numbers.)

b. Main Questions

i. Subset Specifications

1 FIELD DICTIONARY?

This question asks the user if he wants a printout of the field dictionary. The only allowable responses are Y for yes and either N or ⊕ for no.

When the file was described, the File-Format Description Program stored a set of the field names and associated syntax definitions, etc., for the file. The field names, the syntax for field values, data type, and group-membership relations can be printed out if desired at the beginning of each retrieval session. An illustration of such a printout for the Diabetes File defined earlier is given below. (The syntax-definition language is described in detail in Section I-B2.)

```
PATNO, PATIENT UNIT #   [ID]
FIXED 5, UNIQUE
["0";"1"]9999

PN, PATIENT NAME
TEXT, UNIQUE
SX

BD, BIRTHDATE
DATE, UNIQUE
99I"/"99I"/"[I"18"["7";"8";"9"]9;I"19"99]

DOA, ADMISSION DATE
DATE, UNIQUE
99"/"99"/196"["5";"6"]

GROUP: BTESTS, BLOOD TESTS

       DR, DOCTOR
       TEXT, MULTI-VALUED
       SX

       FBS, BLOOD SUGAR
       DEC NUMBER, MULTI-VALUED
       SX
```

The data shown are used by the ISRCH program primarily for validating retrieval statements. First, names of original fields are checked for exact spelling matches, and then field values are checked against the data type (integer, decimal number, date, etc.). This data-class check on specified field values is intended to prevent the initiation of extensive searches for impossible or illegal field values.

```
2 ORDERING-FIELD-VALUE BOUNDS? Y
 LOWER BOUND
  2A1   ID1   ØØ3ØØ
  2A2   ID2   Ø1/Ø1/51
 UPPER BOUND
  2B1   ID1   Ø5999
  2B2   ID2   12/31/55
```

2 ORD-FLD BOUNDS?

This question requires a Y, N, or EOM response. If the answer is N or EOM, further bounds processing is bypassed. If Y is given, further questions are asked (see example above). First, the program prints LOWER BOUND and asks for a value for each of the fields comprising the record identifying field. Then the same process is repeated for UPPER BOUND.

Answers to field-value questions must, of course, be of the type defined for the ordering field(s)—that is, date, decimal number, text, etc. A null response may be given for either of the bounds. When null is entered for the lower bound, ISRCH starts its search with the beginning of the file. When null is entered for the upper, ISRCH continues to the end of the file.

The comment LOWER BOUND EXCEEDS GREATEST VALUE IN FILE is printed when a given lower value is larger than any in the file. NO

VALUE IN FILE EXCEEDS UPPER BOUND is printed when the given upper-bound value is larger than any in the file. In the latter case, ISRCH continues to the end of the file. The user can, of course, return to the questions and change his values before searching begins. By limiting the bounds for searching, a great deal of computer processing and waiting time can be eliminated.

3 DERIVED FIELDS

Often a user wishes to define new fields based on the values for the fields that he already has, e.g., Average Score based on a multivalued-field Test Score, or Age based on Date of Birth and current date. The DERIVED FIELDS question allows the user to define new fields within each record that are functions of the previously defined fields, for the duration of the running of the program.*

All derived fields are based on at least one already existing field. That field may be combined with another field or with a constant, e.g., today's date, introduced by the user. The combination is an arithmetic one-addition, subtraction, multiplication, or division. The already defined field may be used to derive a new field by instead applying a special function to it, e.g., summation, frequency count, or sum of squares. And the derivation of the new field may be made conditional, e.g., IF AGE>14 THEN SUM TEST SCORES.

---

*One running of the program lasts until the program prints the time at the bottom of the typescript and then turns off. A new running of the program requires the user to enter the program's call letters again and identify his file again.

The program permits the kinds of expression concisely defined in Backus-Normal-Form notation shown in Table X. The symbol ::= is used to define new expressions from basic symbols or previously defined terms. Angle brackets < > are used to delimit retrieval terms. A vertical bar means "or." Order of interpretation is not implied by the notations used in Table X; precedence is discussed later. A field name is either an original-field name or a previously defined derived-field name. Constants are decimal numbers or integers. Readers who find it difficult to understand Table X should not despair; the rest of the explanation and the examples should be sufficient explanation of how to derive new fields.

TABLE X. Backus-Normal-Form notation for derived fields.

```
<logical operator> ::= AND|OR
<relational operator> ::= =|>|<|C|NOT=|NOT>|NOT<|B|E
<arith. operator> ::= +|-|*|/
<special operator> ::= SS|SUM|FREQ
<special value criterion> ::= =U|=IND|NOT=U|NOT=IND
<expression> ::= <field name>|<constant>|<express.>
                 <arith. operator><express.>|(<express.>)
<Boolean term> ::= <field name><relational operator><constant>|
                   <field name><special-value criterion>
<Boolean expression> ::= <Boolean term>|NOT <Boolean express.>|
                         <Boolean express.><logical operator>
                         <Boolean express.>|(<Boolean express.>)
<derived field> ::= <express.>|<special operator><field name>|
                    IF <Boolean express.>THEN <special operator>
                    <field name>
```

The user defines his derived field in one of the several ways that are discussed in detail below. He may follow his derived-field definition by a colon and a unique new name for the derived field or he may let the number of the current question (e.g., 3A) be automatically assigned to it. Thus, if <derived field> refers to his definition, the complete answer required in response to DERIVED FIELDS is either <derived field>:<new name> or <derived field>.

Unique fields in a single record may be combined or summarized in the form of new derived fields. Multivalued fields may be compressed or summarized as new derived unique fields. Note that all derived-field operations will result in decimal-number output regardless of the type of input values. The detailed rules for field derivation follow in the next several pages.

Arithmetic Operators. The binary arithmetic operators available for unique-field manipulation are shown in Table XI.

TABLE XI. Arithmetic operators.

Arithmetic Operators	Teletype Symbol
plus	+
minus	-
times	*
divide	/

Arithmetic expressions are evaluated from left to right for operations of the same order of precedence. Parentheses may be used to alter the left-to-right interpretation, as shown in the following examples.

Expression	Value
3 - 4/5	2 1/5
(3 - 4)/5	- 1/5

Date Arithmetic. Fields declared to represent dates are stored in the computer file in a standard date format so that they may be manipulated arithmetically. (By convention, all dates are stored in integer form as cumulative days since January 1, 1849.) The computation of a time interval in years from two date variables is illustrated in Table XII.

TABLE XII. Derivation of date field.

Expression	New Variable
3A ADMISSION DATE - BIRTHDATE	: DAY AGE
3B DAY AGE/365	: AGE IN YEARS

The user may wish to use a date that is not a field in his file, e.g., today's date. To do this, he must enter the date that he wants as an integer, i.e., as the number of days since January 1, 1849. A small program, ISRDATE, discussed in Section IV, provides a means of converting any given date to the correct integer. For example, to derive LENGTH OF STAY (L.O.S.) at some given point such as November 20, 1966, the user would first convert his date of 11/20/1966, using ISRDATE, to 43056. His

expression in answer to the derived-field question in ISRCH would then read 43Ø56 - ADMISSION DATE : L.O.S.

Special Arithmetic Functions have been implemented to prepare or reduce data for statistical inquiries. These functions are the following:

FREQ	count occurrences of a field within a record
SUM	sum numerical values of the field
SS	sum of the squares of multivalued numerical values

All of these functions operate on either a single unique field or a single set of multivalued fields. Their main use is to reduce a series of multivalued-field values to a unique derived-field value. The derived value for each of the arithmetic functions is shown in Table XIII for the values 100, 120, and 130 for the multivalued field Blood Sugar in a given record.

TABLE XIII. Special functions for field derivation.

Blood Sugar Field Values
100
120
130

Function	Value
FREQ BLOOD SUGAR	3
SUM BLOOD SUGAR	35Ø
SS BLOOD SUGAR	413ØØ

These special functions when applied to a unique field yield a frequency count of one and the field value as the sum. The SS (sum of squares) function yields the square of the value of the unique field.

Conditional Field Derivation. The special arithmetic functions may be applied conditionally by using IF and THEN statements. In this case, values are counted, summed, or squared and summed to form derived fields, depending on some set of Boolean conditions evaluated at the same subscript level. (The language used to define Boolean conditions and the precedence of operators are described in the section on ISRCH descriptors.) For example,

```
IF TEST DATE>41272 THEN SUM SUGAR : RECENT SUGARS
```

(41272 is the integer equivalent to 01/01/1962, obtained from ISRDATE).

If a group of test results (test date and test value) in a patient record were stored as diagrammed in Table XIV, the conditional application of the SUM function above would produce a cumulative result of 240 for the record shown.

TABLE XIV. Illustration of conditional use of the SUM function.

Test Date	Sugar Value	Truth Value	Cumulative SUM
02/13/1961	105	False	0
12/24/1962	115	True	115
08/02/1962	125	True	240
02/22/1961	110	False	240

Conditional evaluation may be used for conditions stated in terms of unique fields or multivalued fields in the same group. It is not permissible to refer in one statement to multivalued fields not in the same group. Certain multivalued-field derivations involving more than one group may be constructed in a sequential manner by setting indicator values (unique derived fields) to control subsequent multivalued-field derivations. For instance, assume that Blood Sugar and Blood Urea Nitrogen (BUN) values are stored as independent multivalued fields. The summation of Blood Sugar values over records that contain a BUN value greater than 40 could be requested as follows:

```
IF BUN NOT<4Ø THEN FREQ BUN : F1
IF F1 NOT<1 THEN SUM BLOOD SUGAR : F2
```

In this example, F1 serves as an indicator value to control the subsequent summation of Blood Sugar values.

4 DESCRIPTORS

This question permits the user to specify the various criteria (e.g., BIRTHDATE < Ø1/Ø1/193Ø) on which he may base a selective retrieval. The descriptors or criteria statements describe the records of interest in terms of field values. There are many operators—e.g., greater than, less than, equal to, and others—that, together with values, can be applied to the fields to construct descriptors, e.g., AGE>5Ø. (Descriptors such as AGE>5Ø are sometimes referred to as Boolean terms.)

There are also connectors—e.g., OR and AND—that can be used to combine many simple descriptor statements, e.g., AGE>5Ø AND SEX= "MALE" (which is called a Boolean expression). The word NOT may

also be used to introduce a negative value for field-value evaluations, e.g., SEX NOT="MALE".

Records will be evaluated according to the descriptors to determine whether or not each record meets the conditions of the descriptor. Each record assumes a "truth value" of true, false, or indeterminate according to how it compares with the descriptors.

As many as 20 different descriptor statements may be made in one running of ISRCH. They are of the form <descriptor>:<new name> or <descriptor>. As with derived fields, if no new name is given for a descriptor statement, it is automatically assigned the question number (e.g., 4A) at its reference symbol. The names assigned to derived fields and to descriptors must not be the same as for any other derived field or descriptor or the same as any field or group name. TAG CONFLICT FIX is typed to notify the user if he makes such an error.

Several typical descriptors are illustrated below.

```
4A SEX="M" : MALE
4B INCOME < 5ØØØ : LOW
4C AGE<4
4D SEX="M" AND DIAGNOSIS="111XX" : D1
```

Quotation marks must be placed around values for fields that are text and fixed—i.e., all fields that are not integers, decimal numbers, or dates.

Table XV shows the descriptor formats in Backus-Normal-Form notation, for readers who wish to see it expressed in this way.

TABLE XV. Backus-Normal-Form notation for descriptors.

```
<Boolean term> ::= <field name><relational operator><constant>

<Boolean expression> ::= <Boolean term>|NOT<Boolean expression>|
                         <Boolean expression><logical operator>
                         <Boolean expression>|(<Boolean express.>)

<descriptor> ::= <Boolean expression> : <unique name>|<Boolean
                 expression>
```

The next several pages discuss descriptors in more detail.

Relational Operators. The relational operators available for constructing Boolean terms are shown in Table XVI.

TABLE XVI. Relational Operators.

Operator	Teletype Symbol
Equal to	=
Greater than	>
Less than	<
Contains	CONTAINS or C
Begins with	BEGINS WITH or B
Ends with	ENDS WITH or E

The logic of any of these relational operators may be reversed by placing the term NOT in front of the operator symbol. Users should observe that SEX NOT="MALE" is more inclusive

than SEX=FEMALE because the former descriptor would include all records in which the field Sex was Indeterminate or Unknown as well as FEMALE.

The first three operators in Table XVI work both on numeric quantities and on alphanumeric text strings. CONTAINS, BEGINS WITH, and ENDS WITH work on alphanumeric text strings only—i.e., fixed and text fields. Quotation marks must always be placed around the character string that is to be used in a comparison with the CONTAINS, BEGINS WITH, or ENDS WITH operator. The following example describes a search of the Author field for the name *Lewis*:

```
AUTHOR CONTAINS "LEWIS":LEW
```

As a general rule, it is preferable to use CONTAINS rather than = when scanning text strings, in order to avoid problems of non-match owing to leading or trailing spaces. Comparisons are effected by taking the value in the descriptor statement and the stored text field as left-justified variable-length character strings and matching them character by character until the strings are exhausted. Conceptually, the shorter of the two strings is filled in with trailing spaces so that both strings are of equal length. Comparisons are based on the collational sequence shown in Table XVII.

TABLE XVII. Collational sequence of Teletype characters (from lowest to highest value).

(space)
!
"
#
$
%
&
'
(
)
*
+
,
-
.
/
DIGITS (Ø-9)
:
;
<
=
>
?
@
LETTERS (A-Z)
[
]
↑
←

Table XVIII shows examples of the use of the relational operators.

TABLE XVIII. Examples of the use of the six relational operators.

Comparison	Truth Value
"ABF" > "ABD"	True
"ABD" > "AB"	True
"A10" > "A1"	True
"A1" < "1A"	False
"A1" = "A"	False
"A" = "A1"	False
"A1" CONTAINS "A"	True
"A" CONTAINS "A1"	False
"ABC" B "A"	True
"A" B "ABC"	False
"ABD" E "BD"	True

The relational and logical operators must be used within the general rules of syntax for derived fields and search-descriptor expressions. For example, neither the field name nor the field value may be compound, as CITY C "BOSTON" OR "WASHINGTON":C1. Instead, this descriptor should be phrased CITY C "BOSTON" OR CITY C "WASHINGTON":C1.

Special-Value Criteria. There are two field values, U and IND, that are separate and distinct from all other values. U or Unknown can be entered only from Teletype. Absence of valid data from punched-card sources, as well as null entries or entries of IND as Teletype input, may generate Indeterminate field values.

The ISRCH program has two criteria, =U and =IND, that pertain to fields in which the value is U or IND, respectively. A user might, for example, want to see all the records in which SEX="MALE" OR SEX=U in case some of the Unknowns were, in fact, males. Populations that are defined by =U or =IND are treated like all other populations.

Evaluation. Descriptor expressions are evaluated according to a three-valued logic scheme: true, false, or indeterminate. The indeterminate value results when field-value information is missing in a record—so the evaluation cannot yield true or false.

Table XIX shows "truth tables" for the logical operators OR, AND, and NOT that may help to explain their application to field values.

TABLE XIX. Truth tables.

OR

	T	F	I
T	T	T	T
F	T	F	I
I	T	I	I

AND

	T	F	I
T	T	F	I
F	F	F	F
I	I	F	I

NOT

T	F
F	T
I	I

To illustrate the use of these tables, consider the three Boolean terms below and their truth values.

SEX="MALE"	True
AGE>4Ø	False
INCOME<1ØØØØ	Indeterminate

If the three truth values given were ANDed together, SEX="MALE" AND AGE>4Ø AND INCOME<1ØØØØ, a false value would result:

True AND False = False
False AND Indeterminate = False

If the three truth values given were ORed together, a true value would be obtained. If the first and last truth values were ANDed together, SEX="MALE" AND INCOME<1ØØØØ, the result would be an indeterminate value.

Operator Precedence Summary. Table XX shows the order in which the operators are applied to derived-field and descriptor statements.

TABLE XX. Operator precedence levels as applied to derived-field statements and descriptor expressions.

Operators		Precedence
Conditional[a]	IF, THEN	1
Logical	OR	2
	AND	3
	NOT	4
Relational	=, >, <, CONTAINS, BEGINS WITH, ENDS WITH	5
Arithmetic	+, -	6
	*, /	7

[a]Use is restricted to field derivations only; not used for descriptor formation.

Parentheses may be placed around Boolean terms to override the natural precedence—for example,

Arithmetic Precedence

```
AGE + 2 + 3 * 4:NEWAGE     (Newage = Age + 14)
AGE + (2 + 3) * 4:NEWAGE   (Newage = Age + 20)
```

Logical Precedence

```
A>B OR B>C AND D>E      (AND evaluated before OR)
A>B AND (B>C OR D>E)    (OR evaluated before AND)
```

The internal logic does not permit reversing precedence relations between the arithmetic, relational, and logical groupings as stated in Table XX.

To summarize the use of NOT, it may be used to reverse the logic of relational operators and logical expressions. NOT may precede a Boolean term to negate its truth value. For example,

```
NOT (SEX NOT = "M" OR AGE>4Ø) AND NUM>1ØØØØØ : D2
```

If FIX is printed after the user answers questions 3 or 4, it is because he spelled a field name or operator incorrectly (e.g., NAME CONTIANS "JONES"), typed a field value that is invalid according to the field's data type (e.g., AGE>B), or did not follow the rules of syntax for derived-field or descriptor formation (e.g., NAME CONTAINS AND AGE>14). The question will be reasked.

5 SEARCH LIMITS

After all the descriptors have been defined, question 5 allows the user to apply the descriptors to his retrieval. Question 5

also allows the user to limit the number of records to be read for retrieval.

5A POPULATION BOUNDS

It is in answer to this question that the user states which descriptors already defined in question 4 should form the criteria for his search. The question requires a single descriptor name, a logical combination of descriptor names, or a null as a response.

Thus, if D1, D2, D3, and D4 are defined descriptors, then legal population statements might be

```
D1
D1 AND D2 OR D3
NOT (D4 OR D1) AND D2
```

A null entry specifies the whole file by convention.

5B MAX NO OF RECORDS
5C MAX NO IN POP
5D MAX NO IN SAMPLE

The program also permits the specification of numeric maxima for one or more of the above. The user may specify in answer to question 5B the maximum number of records to be searched (after which the program should read no more records). Question 5C allows him to state the maximum number of records to be found that fit the population (defined in 5A), after which no more records should be read. And question 5D permits the specification of a numeric maximum pertaining to a sample. Any legal integer or a null is an acceptable answer to each of these questions. The searching terminates when any limit is reached.

ii. Data-Output Specification

After the user has specified his population criteria, he proceeds to designate the type of output that he desires from the records in his population. There are two types of output possible from a retrieval request. First, a Teletype print function permits the user to select fields to be printed or abstracted from records fulfilling the population criteria. Second, numeric variables may be accumulated over samples of records for printout in tabular arrays.

Print Function

```
6 FIELDS TO PRINT
 6A FIELD TEACHER'S NAME
 6B FIELD COLLEGE MAJOR
 6C FIELD ⊕
```

The user may specify which fields are to be printed from the records selected. Any original or derived-field name may be given to the print-function question. Group names are not acceptable.

The ordinal position of a record in the file determines its order on printout. The order of field output is dependent on the order in which the user enters the field names to the print-function request. The print format is fixed: each unique field and its value are typed on a new line; multivalued fields and values belonging to the same group are printed in columns, with a maximum of three across the page.

Figure 6 shows an excerpt from a printout by ISRCH for which question 6 was answered as follows:

```
6A PART NO.
6B PART DESCRIPTION
6C ⊕

.
.
.

PART NO.                    1Ø42
PART DESCRIPTION            INVOLUTED WIDGET

PART NO.                    1Ø61
PART DESCRIPTION            IRON CLAMP, 8"

PART NO.                    1Ø77
PART DESCRIPTION            JITNEY SPRING

.
.
.
```

FIGURE 6. Excerpt from a retrieval printout.

Dictionary Retrieval. The creation and maintenance of dictionaries are discussed in Section III-B. At retrieval time, the user specifies to the ISRCH program as part of the print function those fields that are to be treated as codes and the number of the dictionary that is to be used for decoding. For those fields that are to be treated as codes, the ISRCH program will retrieve the field value and use it as a code for a dictionary lookup. The lookup will yield a text expansion of the coded value. Both the field value (code) and the associated text will be printed. In answer to the print question, the user states the field name

followed by a comma and the dictionary number. For example, ICDA, 124 would mean that the user wants the field named ICDA decoded according to dictionary number 124. After the entry is typed in, the program responds by typing back the dictionary title, initials of originator, and time and date of last update:

```
6C ICDA,124
   124 DISEASE DICT. (SGT) 2:15 1/4/1966
```

As many as 16 different dictionaries can be used in one running of ISRCH, and there is no limit to the number of fields that can be treated as codes.

The usual print format of ISRCH is modified in the case of fields treated as codes to have (:) (space) (text) following the field value (code). An asterisk (*) following the field value (code) indicates that there is no entry in the dictionary for the given

```
UNIT NO   888-33-21
SEX       F: FEMALE          (Code F has text FEMALE
                              associated with it.)

UNIT NO   888-37-28
SEX       M: MALE

UNIT NO   888-41-3Ø
SEX       J*                 (No code entry in dictionary
                              125 for J.)

UNIT NO   888-7Ø-31
SEX       I:                 (A null entry in the dictionary
                              corresponds to code I.)
```

FIGURE 7. Retrieval printout with dictionary lookup.

field value (code). An asterisk followed by text means that the field value was longer than the seven-character maximum for a code and that the program truncated the field value, used the first seven characters as the code, and printed the text corresponding to the truncated field value. If there is nothing following the colon, a null text exists in the dictionary.

Figure 7 shows an excerpt from a printout by ISRCH in which the field Sex was decoded to the values in dictionary number 125. Comments are in parentheses.

```
SAMPLE RECORD PRINTOUT
3/31/65

     UNIT NO                    1463871
     LENGTH OF STAY               16
     TYPE                       4: SURGICAL

     DIAGNOSIS                  DIAG DATE

     111XX                      5/12/1962
     264XX                      5/12/1962
     29Ø1X                      5/13/1962
     123XX                      5/21/1962
     124XX                      5/25/1962

     OPCODE                     OPTEXT              SURGEON

     444SS                      BILATERAL           3142
     221SS                      CHRONIC             4Ø21

     SPECIAL CHARGES

             114.35
             2ØØ
              25.25
```

FIGURE 8. Retrieval printout with multivalued fields and groups.

The answers to question 6 read as follows:

```
6A UNIT NO
6B SEX, 125
       SEX CODES (PLF) 8:ØØ PM 11/3Ø/1966
6C ⊕
```

A typical printout for one record composed of three unique fields, two groups of multivalued fields, and one other multivalued field is reproduced in Fig. 8. The Type field is a code. All fields of type text, fixed, or date are left-justified. Decimal numbers and integers are aligned on the explicit or implicit decimal points, respectively, to produce more-readable printouts. (The actual printout does not occur until all the questions asked by the ISRCH program have been answered.)

iii. Interrecord Sum Function

```
7 FIELDS TO SUM
 7A NUMBER OF RECORDS : REC
 7B AGE : AGE
 7C PATIENT'S BILL : PTB
 7D ⊕
```

This question pertains to the second type of retrieval output. Question 7 asks the user to specify any accumulations of numeric values across records; the crosstabulations that he specifies here may be printed in the form of a matrix or array, which he further describes in questions 8 and 9. This accumulation facility permits the summing of record values in order to obtain total-file counts, sums, or sums of squares. Any numeric unique-field values, original or derived, may be summed. (Unique fields may

be derived from multivalued fields. See question 3.) Interrecord summing is requested by typing any numeric unique-field name followed by a colon and then followed by a three-letter symbol (see example above). The three-letter symbol, or tag, is printed with the matrix as its title.

The fields may be summed over the entire population of records specified in question 5A or for a selected sample of those records. The selection conditions for summing values are specified by a two-dimensional matrix of sample descriptors. The specification of two-dimensional arrays is discussed below. If no sample is defined, the fields specified for summing in question 7 are summed for all records that fit the population, and the totals are printed out as single numbers.

If any interrecord sums have been requested, the following questions are asked to allow the user to define his sample and describe his matrix.

```
8  ROW SPECS
 8A R1  SAMPLE D1 AND NOT D2
 8B R1  LABEL DIABETIC CASES
 8C R2  SAMPLE D3
 8D R2  LABEL CONTROL CASES
 8E R3  SAMPLE ⊕

9  COLUMN SPECS
 9A C1  SAMPLE D4
 9B C1  LABEL OLD PATIENTS
 9C C2  SAMPLE D5
 9D C2  LABEL YOUNG PATIENTS
 9E C3  SAMPLE ⊕
```

Two-dimensional matrices or arrays are used to define samples for

summing. A set of row-and-column intersections defines an array. The program alternately requests a row (column) sample descriptor and a row (column) label. Sample descriptors are of the same form as required in the population statement, question 5A-descriptor names and logical relators. They define the specific subsamples of records to be summed. The field that will be summed is that designated in question 7. If more than one field is specified in question 7, separate matrices will result. Labels are English-text labels and are printed as headings to the respective rows and columns of the matrix. As stated above, if no row specification is given, the entire record population is assumed. Column specification is unnecessary if a Table with only one column is required. A maximum of 20 rows and 4 columns may be specified.

The requested matrices of summed values are printed in a fixed row-and-column format after the completion of a record search and optional printout of selected records. As stated above, each variable summed is displayed in a separate Table with the specified row-and-column captions.

REC MATRIX

	OLD PATIENTS	YOUNG PATIENTS
DIABETIC CASES	135	118
CONTROL CASES	214	2Ø1

FIGURE 9. Matrix printout.

A printout of the first of the two-by-two matrices defined by questions 7, 8, and 9 above is shown in Fig. 9.

1Ø STUDY TITLE

This question asks the user to type the text that he would like printed as a heading to the printout. The program accepts any character string or EOM.

11 OK TO SEARCH

The program expects one of three alternatives:

(1) Y for YES;

(2) ← followed by a conventional command, e.g., ←4, ←L; or

(3) N to halt the program.

Answer (1) to this question completes the user's participation and initiates the actual search. After the file-printout and the matrix-printout operations are completed, a summary of the search reports the number of

```
RECORDS SEARCHED
RECORDS SELECTED
RECORDS INDETERM
SAMPLE SELECTED
```

Then question 11 is reasked.

If the user interrupts the ISRCH program by hitting the break key while the program is searching, the program asks CONTINUE? If

the user answers YES, the program continues from where it left off. If the answer is NO, the program returns to question 11 so that the user can change his request by alternative (2) above.

TABLE XXI. Space allocation for ISRCH requests.

Element	Points
Fields to print	
Each field	2
Descriptors, derived fields, population statements	
Literals	(text characters +1)/3
OR, >, CONTAINS, <, =U, =IND, B, E	1
Reference to a field	1
Integer, decimal number	3

A detailed example of an actual retrieval appears on the following pages, accompanied by explanatory notes.

4. Space Restrictions

The following space limitations must be considered by those ISRCH users who have many descriptors. A total of 480 points is allowed for input to the combination of derived fields, descriptors, population specifications, and fields to print. Points are calculated according to Table XXI. When the total points typed in by the

user approach 400, STORAGE WARNING is printed. When the total points exceed 480, BUFFER OVERFLOW is printed and the program halts.

Notes to Typescript III *(see facing page).*

1 The user requests the printout of a field dictionary. The field names, types of data, whether the field is unique or multivalued, and syntax definitions are printed for each field.

3A-3E The user derives additional fields to be used during the retrieval. 3A-3D are derived from fields that exist

TYPESCRIPT III. Search Program.

```
4:28 PM 1/9 BBN22-22
ISRCH SGT


SEARCH

1 FILE ←/L
1 FILE NUMBER 16 DIABETES FILE (SGT) 4:38 PM 1/9/1968
 1A CONFIDENTIAL CODE BBN

1 FIELD DICTIONARY? Y

PATNO, PATIENT UNIT # [ID]
FIXED 5, UNIQUE
["Ø";"1"]9999

PN, PATIENT NAME
TEXT, UNIQUE
SX

BD, BIRTHDATE
DATE, UNIQUE
99I"/"99I"/"[I"18"["7";"8";"9"]9;I"19"99]

DOA, ADMISSION DATE
DATE, UNIQUE
99"/"99"/196"["5";"6"]


GROUP: BTESTS, BLOOD TESTS

       DR, DOCTOR
       TEXT, MULTIVALUED
       SX

       FBS, BLOOD SUGAR
       DEC NUMBER, MULTIVALUED
       SX

2 ORDERING-FIELD-VALUE BOUNDS? N

DERIVED FIELDS
3A FREQ PATIENT UNIT #: COUNTER
3B (ADMISSION DATE - BIRTHDATE)/365 : AGE
```

Notes to Typescript III (continued).

3A-3E in the file. 3E is derived from the fields derived in 3C and 3D. Either the fields' short names or long could be used; here the long names are used.

4A-4E These five descriptors establish criteria for the searching. Again the user chooses the long field names. No descriptor names are given, so the question numbers (4A-4E) become the reference tags.

5A Descriptors 4B and 4C are to be used as selectors for this retrieval.
5B The user wishes to search only the first 1000 records.
5C-5D The population and the sample are not further limited.
6A-6G The user lists the fields that he would like printed from the records selected. This time he uses the short field names.

7A-7C Three different values for matrix summation are specified.

8-9 The format of the matrices is specified by (a) sets of descriptors that determine the selection of records to be included in the matrix and (b) the row and column labels that are to be printed.

1Ø The user gives a title to his retrieval.
11 The actual search of the records is initiated.

TYPESCRIPT III (continued).

```
 3C  FREQ BLOOD SUGAR : BSFREQ
 3D  SUM BLOOD SUGAR : BSSUM
 3E  BSSUM/BSFREQ : AVERAGE BLOOD SUGAR
 3F  ⊕
DESCRIPTORS
 4A  AGE < 6Ø
 4B  DOCTOR CONTAINS "RYDER"
 4C  BLOOD SUGAR > 8Ø AND BLOOD SUGAR < 12Ø
 4D  DOCTOR C"RYDER" AND BLOOD SUGAR > 8Ø AND BLOOD SUGAR < 12Ø
 4E  AVERAGE BLOOD SUGAR > 8Ø AND AVERAGE BLOOD SUGAR < 12Ø
 4F  ⊕
SEARCH LIMITS
 5A POPULATION BOUNDS 4B AND 4C
 5B MAX NUMBER OF RECORDS TO SEARCH 1ØØØ
 5C MAX NUMBER OF RECORDS IN POPULATION ⊕
 5D MAX NUMBER OF RECORDS IN MATRIX SAMPLE ⊕
FIELDS TO PRINT
 6A  PATNO
 6B  PN
 6C  BD
 6D  AGE
 6E  AVERAGE BLOOD SUGAR
 6F  DR
 6G  FBS
 6H  ⊕
FIELDS TO SUM
 7A  COUNTER:CNT
 7B  BSSUM:SUM
 7C  BSFREQ:FRQ
 7D  ⊕
MATRIX ROW SPECS
 8A R1  SAMPLE 4A
 8B R1  LABEL YOUNG
 8C R2  SAMPLE NOT 4A
 8D R2  LABEL OLD
 8E R3  SAMPLE ⊕
MATRIX COLUMN SPECS
 9A C1  SAMPLE 4E
 9B C1  LABEL NORM.B.S.
 9C C2  SAMPLE NOT 4E
 9D C2  LABEL ABNORM.B.S.
 9E C3  SAMPLE ⊕
1Ø STUDY TITLE A NORMAL BLOOD SUGAR WITH RYDER AT SOME TIME ATTENDING
11 OK TO SEARCH? Y
```

Notes to Typescript III (continued).

TYPESCRIPT III (continued).

```
4:53 PM 1/9/1968
A NORMAL BLOOD SUGAR WITH RYDER AT SOME TIME ATTENDING

          :

PATIENT UNIT #          1ØØ94
PATIENT NAME            PETERSON, TIM
BIRTHDATE               8/14/1922
AGE                          42.452Ø5
AVERAGE BLOOD SUGAR          9Ø.5

DOCTOR                  BLOOD SUGAR

RYDER, I.                    12Ø
JAMISON, P.P.                1Ø4
RYDER, I.                     52
RYDER, I.                     43
IND                          1ØØ
IND                          124

          :

PATIENT UNIT #          16Ø74
PATIENT NAME            SEMPLE, JOHN
BIRTHDATE               12/12/1899
AGE                           66.16438
AVERAGE BLOOD SUGAR          161.1667

DOCTOR                  BLOOD SUGAR

JONES, J.P., III             22Ø
RYDER, I.                     96
RYDER, I.                    135
MATHEWS                      125
RYDER, I.                    187
RYDER, I.                    2Ø4

          :

RECORDS SEARCHED        1ØØØ
RECORDS SELECTED        814
```

Notes to Typescript III (continued).

TYPESCRIPT III (continued).

RECORDS INDETERM Ø
MATRIX SAMPLE 814

5:Ø4 PM 1/9/1968

CNT MATRIX

	NORM.B.S.	ABNORM.B.S.
YOUNG	211	272
OLD	9Ø	241

SUM MATRIX

	NORM.B.S.	ABNORM.B.S.
YOUNG	84936	145271
OLD	31Ø7Ø	13249Ø

FRQ MATRIX

	NORM.B.S.	ABNORM.B.S.
YOUNG	872	11ØØ
OLD	36Ø	964

Notes to Typescript III (continued).

11 The user wishes to change some of his search criteria and the title for another retrieval.

5A The user changes the selectors so that the new retrieval will reflect the fact that the Doctor and Blood Sugar values are grouped. Only descriptor 4D will apply to this retrieval.

1Ø There is no space on the line for the new entry. The user hits the carriage-return key and when the Teletype ball goes to a new line, he enters a backslash to erase the carriage-return from his answer and then he enters his new title.

TYPESCRIPT III (continued).

```
11  ←5A,1Ø
 5A POPULATION BOUNDS 4B AND 4C 4D
1Ø STUDY TITLE A NORMAL BLOOD SUGAR WITH RYDER AT SOME TIME ATTENDING
\A NORMAL BLOOD SUGAR ORDERED BY RYDER
11 OK TO SEARCH? Y

5:1Ø PM 1/9/1968
A NORMAL BLOOD SUGAR ORDERED BY RYDER

          :

PATIENT UNIT #          16Ø74
PATIENT NAME            SEMPLE, JOHN
BIRTHDATE               12/12/1899
AGE                           66.16438
AVERAGE BLOOD SUGAR          161.1667

DOCTOR                  BLOOD SUGAR

JONES, J.P., III              22Ø
RYDER, I.                      96
RYDER, I.                     135
MATHEWS                       125
RYDER, I.                     187
RYDER, I.                     2Ø4

          :

RECORDS SEARCHED        1ØØØ
RECORDS SELECTED         72Ø
RECORDS INDETERM         135
MATRIX SAMPLE            72Ø
```

B. Dictionary-Maintenance Program—ISRDIC

The dictionary function, as stated earlier, allows users to create dictionaries of codes and associated text strings independent of any file. The preceding description of the ISRCH program discussed the retrieval or decoding of dictionary entries. This section describes the ISRDIC program, which is used to create and maintain ISR dictionaries.

The dictionary facility makes it possible to

(1) assimilate data that are available only in coded form and yet obtain printouts in expanded form;

(2) enter data rapidly from a terminal with the ISRTTI program by using short codes and yet obtain expanded printouts;

(3) reduce the size of stored records by filing only the precoded data, which can be expanded on printout.

Because the dictionary that is to be associated with a field is specified only at print, or retrieval, time,

(1) different dictionaries may be used for the same code field;

(2) dictionaries may be created and used after the data files have been created and the code field values entered.

Because the dictionary texts are independent of the filed data, it is not possible to use the dictionary facility to store information in the records in decoded form or to use the dictionary texts as field values in descriptor statements.

ISRDIC is used to create, expunge, update, examine, punch (onto paper tape), and load (from paper tape) the ISR dictionaries.

Rather than ask a sequence of questions, this program allows the user to specify functions and their related dictionaries by using a control language. The program's main question merely types C: (in long form; or nothing in short form) and accepts either a Code or one of several Commands. This question is always reasked after the completion of specified functions.

All commands begin with the back arrow (←); any entry not beginning with ← is taken as a code. Codes are entered at this main question to create, alter, or look up individual entries, as described below under Teletype Input/Output. Commands may be entered for such functions as specifying the dictionary to be worked with, printing all the entries in a dictionary, and punching and reading paper tapes containing dictionary entries.

Although this program does not have the appearance of a typical question-and-answer program, the usual ← commands may be used: ←/S (to change to short form), ←1 (to return to the main question), and so on.

1. Dictionary Titles and Entries

Each dictionary has a number and a title and may contain any number of entries. Dictionary titles are established at the time that a dictionary is created (see ←DIC and ←NEW below). A title consists of up to 35 characters entered by the user and also has associated with it the initials of the originator and the time and

date of the last update. This closely parallels the number-and-title scheme of data files. The numbers associated with the dictionaries are greater than 100. (ISR data files will usually have numbers lower than 100.)

Each entry in a dictionary consists of a code of up to 7 characters and an associated text of up to 72 characters. These characters may be letters, numbers, spaces, or any other printing characters on the Teletype keyboard except ↑ and ←. If leading spaces are typed in, they are considered as part of the code or text. Trailing spaces that are entered with codes are ignored, but those entered with texts are retained.

At present, there may be 3995 dictionaries, with a total of almost 20,000 entries. (The number of entries could be expanded readily.)

2. Connection to a Dictionary

When the ISRDIC program is connected to a given dictionary, any Teletype input or output done by the user will be to or from that dictionary and certain commands (e.g., ←PRINT) will pertain to that dictionary.

Connection to a dictionary is established by commands such as ←DIC and ←NEW [Section III-B4(a)]. Subsequent commands will involve that dictionary, and the program remains connected to it upon the completion of each command. Connection can be established also by a command that is followed by a list of dictionary numbers. In such cases, the program connects to one dictionary

at a time, and, when it is through operating on the last dictionary in the list, it disconnects from all dictionaries.

Several users may be connected to the same dictionary at any one time.

3. Teletype Input/Output

This section concerns the Teletype input and output of single entries only; Teletype output of an entire dictionary is effected by the ←PRINT command, discussed below.

Typing the code of interest at the main question initiates the input or output of a single entry in the dictionary to which the program is connected. If the user attempts to enter a code when the program is not connected to a dictionary, NOT CONNECTED TO A DIC is printed and the question is reasked. Any string not beginning with ←, including null, is taken as a code. After an acceptable code has been entered, three spaces are printed. If there is a dictionary entry for this code, the text is printed next, followed by three more spaces. (The last three spaces are not printed if there is no entry in this dictionary for this code. This makes it possible to distinguish codes that are not in the dictionary from codes than have null texts.) Then the next question T: is asked on the same line. In short form, this question, like the main one, has neither question number nor text. Valid responses to this question are—

> any legal text string—the text is stored in the dictionary file with the given code, replacing any previous text for that code;

↑K—this removes the dictionary entry corresponding to the given code, or, if there was no entry, it prevents the creation of one;

⊕—null leaves the text unchanged (this is useful when the user wishes to look up individual entries) for an already existing dictionary entry or enters a null text if no entry existed before.

After the text question has been answered, the main question is reasked.

In the example below, the user defines the codes Ø2138, Ø2139, and Ø2174.

```
C:  Ø2138    T: HARVARD SQUARE
C:  Ø2139    T: CENTARL SQUARE
C:  Ø2174    T: ARLINGTON
```

Here, he examines but does not change the entry for Ø2138:

```
C:  Ø2138    HARVARD SQUARE   T: ⊕
```

And here he corrects the entry for Ø2139 and deletes the entry for Ø2174:

```
C:  Ø2139    CENTARL SQUARE   T: CENTRAL SQUARE
C:  Ø2174    ARLINGTON   T: ↑K
```

The example below shows a user who wants to look up some ZIP codes without changing the dictionary file. He answers the text question with ↑K, if there was no entry for the code, or with a null, if there was an entry:

```
C:  Ø2174    T: ↑K
C:  Ø2139    CENTRAL SQUARE   T: ⊕
```

Teletype Input/Output error messages include—

NOT CONNECTED TO A DIC—when a code is typed at the main question and the program is not connected to a dictionary;

THAT'S TOO LONG—when the user's entry is over 7 characters for code, 72 for text;

ILLEGAL CHARACTER—when carriage return, linefeed, or a control character is typed.

In all cases, the question is reasked after the error message has been printed.

4. Commands

Commands are entered in answer to any asking of the main question; they must be introduced by a ←. Some commands cause other questions to be asked before the program will return to the main question. The use of some of the commands is shown in the sample typescript on page 109.

a. Creating, Connecting to, and Expunging Dictionaries

←NEW creates dictionaries. If the command is given alone, the next available number is assigned (numbers are assigned consecutively, beginning with 101). After the command has been entered, the program asks TITLE?... A response of up to 35 acceptable characters is allowed. It is stored, along with the time and date and user's initials, to form the complete title of the dictionary. NUMBER, followed by the number of the dictionary just created, is then printed. Entering ← at the title question will prevent the creation of a new dictionary. Entering more than 35 characters or illegal characters causes THAT'S TOO LONG or ILLEGAL CHARACTERS to be printed and the question to be reasked. ←NEW

leaves the program connected to the newly created dictionary. In the example below, the user creates a dictionary entitled ZIP CODES, which is assigned number 1Ø6.

```
C: ←NEW    TITLE?.. ZIP CODES
 NUMBER: 1Ø6
```

←NEW (number) will create a dictionary with the given number, provided the number is not currently in use but had previously been used (and expunged). TITLE?.. is asked as above. If the user enters a number less than 101 or greater than the maximum yet assigned, NUMBERS <1Ø1 ARE NOT ALLOWED or NUMBER TOO BIG COULD NOT HAVE BEEN USED BEFORE is printed and the question is reasked.

←TITLES causes a list of all existing dictionary numbers and titles to be printed.

←DIC (number) connects the program to the dictionary with that number. The title of the dictionary is typed to the right of the command as verification.

```
C: ←DIC 1Ø6    ZIP CODES    (JRB) 1Ø:19 PM 12/13/1966
```

If no dictionary exists with the specified number, NO SUCH DIC will be printed and the question reasked.

←EXPUNGE is used to expunge the dictionary to which the program is connected. All of the information in the dictonary is deleted. To make sure that the typist really wants to destroy the dictionary, the title of the dictionary is typed, followed by the question EXPUNGE?.. . A YES answer will cause the removal of all the entries from the dictionary (a slow process), after which the number of entries will be printed. EXPUNGE TITLES?.. is then asked;

a NO answer will retain the empty dictionary; a YES will eliminate it completely. Note that expunging does not release any useful storage space at the time of running; hence, it is bad practice to expunge and reload a dictionary repeatedly.

b. Printing, Punching, and Counting

←PRINT, ←PUNCH, and ←COUNT, respectively, are commands to print, punch on paper tape, and count the entries in a dictionary. Each of these commands may be followed by

(a) nothing, if the program is connected to a dictionary, or
(b) one or more dictionary numbers separated by commas.

If no dictionary number is given, the command will be executed for the dictionary to which the program is connected. If the command is given with no dictionary number and the program is not already connected to a dictionary, WHAT DIC? will be printed and the question will be reasked.

If several dictionary numbers are given, the command will be executed for each in turn and the program will disconnect from all dictionaries upon completion of the last dictionary. Execution of the command for each number is as follows: first, the number and title of the dictionary are printed; next, the specified function is executed for that dictionary; last, the number of entries in the dictionary is printed.

←PRINT causes a column of codes to be printed to the left of a column of corresponding texts. At present, the order of printout is random because it is based on the order of storage on the Fastrand drum.*

←PUNCH causes a paper tape to be punched in a format that is compatible with the ←READ function. The dictionary title is punched exactly as it is printed, and all that dictionary's codes and corresponding texts are punched too. If several numbers are given with the command, the dictionaries are punched in separate sections on the same tape, each preceded by its number and title. If the punch is already being used when this command is given, PUNCH BUSY will be printed and the command will not be executed. Before using this command, the user should telephone the computer operator, ask him if the punch is free, and advise him how to label the resulting tape.†

Users may find it useful to punch their dictionaries onto paper tape. A paper-tape copy of a dictionary serves as backup in case

---

*The codes are hash-coded to drum addresses. The command PRINT WITH HASH causes the hash codes to be printed in a third column to the left; this is probably of interest only to the programmer.

†Adequate labeling of tapes includes user's initials, time and date, and dictionary name and number. If a new tape is punched to replace a previous one, the operator should be instructed to discard the old.

the information on the storage drum should be destroyed.

←COUNT merely causes the entries to be counted and the number to be printed.

The print, punch, and count functions are extremely slow, even if the dictionaries specified have only a few entries, because the organization of the dictionaries is designed for fast retrieval of individual codes. To find all the entries in a given dictionary entails a time-consuming search of the entire dictionary storage space. To keep the user informed that the program is running, a Teletype "cough" occurs at intervals. If the user presses the BREAK key, INTERRUPTED, CONTINUE?.. is asked. YES will cause continuation from the point of interruption, without any repetition of searching; NO will cause the previous question to be re-asked.

c. Reading and Verifying Paper Tape

←READ causes the reading of a paper tape punched with the punch function. As soon as the dictionary number and title are read from the tape, the following are printed:

TAPE (number and title read from tape)
DIC (number): (title of this dictionary as filed)
READ?..

Answering NO to this question will cause tape to be skipped until the next dictionary number or the end of the tape is reached. A YES will cause this section of the tape to be read: all entries on the tape will be entered in the dictionary whose number was on the tape; if any of the codes on the tape had different texts in the dictionary, they will be changed to match the tape. Note that the read function leaves unchanged the entries for codes that are not on the tape. When the entries for one dictionary have been read completely, the number of new or different entries found on the tape is printed. If there are entries for another dictionary on the tape, the above process is repeated.

←READ AND PRINT (or ←READPRINT) acts like ←READ, with the additional feature that all new or change entries are typed in a format similar to ←PRINT.

←VERIFY and ←VERIFY AND PRINT (or ←VERPRINT) are like ←READ and ←READ AND PRINT, respectively, except that they do not ask READ?.. and do not change the dictionary. That is, these functions count (and print in the AND PRINT case) new and different entries. Thus, these functions are useful for determining if there have been any deletions from a dictionary punched at an earlier date.

The use of AND PRINT as a means of listing long tapes should be minimized, as it ties up the reader for the duration of the printing. Instead, the user should read the tape into a dictionary and use ←PRINT.

A sample of an actual dictionary maintenance typescript appears on the following pages, accompanied by explanatory notes.

Notes to Typescript IV. *(see facing page).*

The user creates a new dictionary.
It is assigned number 102.
The user enters two new codes and texts.

He examines an already created entry.
He examines the other already created entry and deletes it.
He enters two more codes and texts.

He requests a printout of his complete dictionary.

He asks to be connected to a different dictionary, number 107.
He looks at two existing entries.

He requests a count of the entries in dictionaries 107 and 108.

He requests a paper tape to be read.

The tape says that it contains entries for dictionary 102.
The title of dictionary 102 is printed.
The user says that he does want the tape read and the entries added to dictionary 102.
There were 39 new entries made.

The user dismisses the ISRDIC program.

TYPESCRIPT IV. Dictionary maintenance program.

```
1Ø:11 AM 1/15  BBN4-4
ISRDIC JRB

DICTIONARY MAINTENANCE

C: ←NEW   TITLE?.. ZIP CODE DICTIONARY
 NUMBER: 1Ø2
C: Ø2138    T: HARVARD SQUARE
C: Ø2136    T: BROOKLINE, MASS.
C: Ø2138    HARVARD SQUARE    T: ⊕
C: Ø2136    BROOKLINE, MASS.    T: ←K
C: Ø2146    T: BROOKLINE, MASS.
C: Ø214Ø    T: NORTH CAMBRIDGE
C: ←PRINT

1Ø2   ZIP CODE DICTIONARY  (JRB)  1Ø:19 AM 1/15/1968

Ø214Ø   NORTH CAMBRIDGE
Ø2138   HARVARD SQUARE
Ø2146   BROOKLINE, MASS.
...3 ENTRIES...
PRINT COMPLETE

C: ←DIC 1Ø7   MASS CERTIFICATION CODES (PLF) 1Ø:ØØ AM 11/7/1966
C: 47   SPEC SUBJ - ART    T: ⊕
C: 95   VOC - PRACTICAL NURSING    T: ⊕
C: ←COUNT 1Ø7, 1Ø8

1Ø7   MASS CERTIFICATION CODES (PLF)  1Ø:ØØ AM 11/7/1966
...63 ENTRIES...

1Ø8   TEST CODES  (JRB)  9:19 PM 2/14/1967
...419 ENTRIES...
COUNT COMPLETE

C: ←READ

TAPE: 1Ø4   LAB CODE DICTIONARY
DIC 1Ø4:   LAB CODE DICTIONARY (JRB)  1Ø:19 PM 2/15/1968
READ?.. Y

...39 NEW OR DIFFERENT ENTRIES...
READ COMPLETE

C: ←-
11:1Ø AM
```

IV. AUXILIARY PROGRAMS—ISRDCH, ISRSHF, ISRCDO, ISREXP, ISRDUMP, ISRLOAD, ISRCDPR, ISRDATE, ISRLIST, ISRTLIST, ISRSIZE

A. Description Change Program—ISRDCH

There are times when a user wishes to change his file description. He may, for example, find at input time that his data are different from what he expected and that he must change a syntax definition. Other times, he may need to add a field. Changes to a file description may be made with the Description Change Program, ISRDCH.

If the file as originally described contains no data, any kind of change may be made. If there are already data in the file, only the following kinds of changes can be made without loss of the data: file title, confidential code, all card information, field and group names, and syntax definitions.

If other kinds of changes are desired—data type, uniqueness, group membership, number of fields, order of fields—and the original file contains any data, the data must all be expunged (by the program) before the changes are made.

When the Description Change Program is called up, it requests a file number and confidential code (see p. 33); then the program goes to the last of the ISRDES questions for that file, e.g., 14 OK? The program will from then on act like ISRDES with all but the last question answered. Thus, the user can type ← and all the appropriate question numbers for the things he wishes to change. (If he does not have a copy of his description, he can begin by

by getting a listing of the description [←L] so that he will know which questions he wishes to return to for changes.) When the corrections are completed, the program will return to the OK? question. After the user types YES, the program makes the designated changes, unless data must be expunged. In that case, INCOMPATIBLE CHANGE. EXPUNGE DATA? is printed. Answering anything other than YES returns the program to the OK? question.

If the user dismisses the program at any time (by entering ←- in answer to a question), the file will be left as originally described.

A file cannot be changed with ISRDCH if it has derivative files or is a derivative file. (See File Shuffle Program, Section IV-B, for explanation of derivative files.)

B. Shuffle Program—ISRSHF

At assimilation time, records are stored in sequence according to the values of the record-identifying field(s). Users may wish to have their retrieval outputs printed in a sequence other than the one based on those values. There are also times when the user wishes to save time by limiting the retrieval process to prescribed population bounds within a sequence. For example, assume that records are reordered by diagnosis code as illustrated below.

Diagnosis Code

1	(Other diagnosis)
2	(Other diagnosis)
.	.
.	.
.	.
124	(Heart involvement)
125	(Heart involvement)
.	.
.	.
.	.
165	(Heart involvement)
166	(Other diagnosis)
.	.
.	.
.	.
.	.

If the ISRCH Program is instructed to find all records between a lower-bound code (=124) and upper-bound code (=165), the records of interest could be isolated rapidly without the need for scanning records outside those bounds.

The Shuffle Program, ISRSHF, is provided so that a user may construct derivative files or indexes that are ordered by fields other than the record-identifying field(s). The derivative files will have the same contents as the original. Their order may be any combination of the unique fields in the original file: a single field (as in Part A of Table XXII) or a combination of fields (as in Part B of Table XXII). The original file is left untouched. The ISRSHF program creates an additional new file that references the same data as the original, but in a new order. The order of the new file will be in accordance with the standard (numerical-alphabetical) collational sequence. (See p. 70.)

TABLE XXII. Examples of reindexing a file.

A. Indexing on Teacher	B. Indexing on Teacher and Class
ABBOTT	ABBOTT - HISTORY
JONES	JONES - ECONOMICS
JONES	JONES - GEOGRAPHY
JONES	JONES - HISTORY
JONES	JONES - POLITICAL SCIENCE
SMITH	SMITH - HISTORY
WHITE	WHITE - CALCULUS

1. Program Operation

In response to the first ISRSHF question, the user must identify his source file by file number and by confidential code if applicable (see p. 33). The user is then asked to enter the list of field names by which the new file is to be ordered. Fields of any data type (text, integer, decimal number, or date) may be used. They must be unique fields, however. If the user enters a multivalued field name or group name, PRIMARY FIELDS ONLY is printed and the question is reasked.

If more than one field is entered, the first field that is typed is interpreted as the major ordering field, the second of second-most importance, etc. A progressive ordering-precedence scheme is used. Table XXIII shows file construction for Doctor and Disease and for Disease and Doctor.

TABLE XXIII. Two possible index orders.

Record	Doctor and Disease	Disease and Doctor
1	DR. ADAMS - ARTHRITIS	ARTHRITIS - DR. ADAMS
2	DR. ALBERT - DIABETES	ARTHRITIS - DR. BROWN
3	DR. BROWN - ARTHRITIS	BRONCHITIS - DR. SAMUELS
4	DR. BROWN - COLITIS	COLITIS - DR. BROWN
5	DR. SAMUELS - BRONCHITIS	DIABETES - DR. ALBERT

When processing is completed, the program types out the file number of the new file and the number of records processed. The new file has the same name, confidential code, and date as the original indexed file, but it is distinguished by the printout of its field-order scheme, which appears on typeouts for files that are derived. The example below shows typeouts of original- and derived-file names.

```
2 JONES (CRJ) 6:1Ø PM 6/12/1966

3 JONES (CRJ) TEACHER, CLASS 6:1Ø PM 6/12/1966

5 BIOCHEM (CRJ) 9:23 AM 6/23/1966

7 BIOCHEM (CRJ) EXPERIMENT NO. 9:23 AM 6/23/1966
```

If the user hits the break key during the processing, the following messages are typed.

```
INTERRUPTED

X RECORDS PROCESSED

CONTINUE?
```

If the answer to the CONTINUE question is YES, processing continues; if NO, the program halts and no derivative file is created.

The ISRSHF program may be called several times to operate on the same original data file so that multiple indexes may be created for different purposes.

2. Maintenance of Derivative Files

Derivative files compliled by ISRSHF are updated automatically when records are added or revised in the original ISR file. However, derivative files cannot be changed by using ISRTTI directly on them. (Users should realize that updating a file with derivatives will take longer than updating the same file without any derivatives and also that the more derivatives there are to update, the longer the process will take.)

Below is a sample typescript of the ISRSHF program.

```
11:4Ø AM 2/2Ø BBN17-17
ISRSHF SGT

SHUFFLE

1 FILE: 16 DIABETES FILE (SGT) 2:38 PM 1/9/1968
 1A CONF CODE: BBN
1 ORDER: DOCTOR PRIMARY FIELDS ONLY
1 PATIENT NAME, BIRTHDATE
  18 DIABETES FILE (SGT) PATIENT NAME, BIRTHDATE 2:38 PM 1/9/1968
  2872 RECORDS PROCESSED

11:56 AM
```

C. Punched-Card-Output Program—ISRCDO

Data stored in an ISR file may be converted to standard punched card format and written on magnetic tape by using the ISRCDO program. This facility is useful for long-term storage of data that have been entered from Teletype sources. It is useful also for preparing data that are in one ISR file for merging into another file. Because the files created using ISRCDO are machine-independent, data may also be prepared for processing at other computer centers.

The output file created on magnetic tape is in standard format: 84 Hollerith-code characters per tape record, with records separated by interrecord gaps. The tape begins with an end-of-file mark; all files are separated by an end-of-file mark; and the tape is ended by three end-of-file marks. (The user should make sure that the magnetic tape that he uses has three end-of-file marks at the beginning of it before he starts using the Punched-Card-Output Program. The computer operator can do this for him.)

1. Input Specification

Operation of ISRCDO requires that the user specify which fields in his particular file are to be processed. He must also indicate the first and last columns into which each field is to be written. Up to three cards may be produced in one running of the ISRCDO program. Card columns 1-80 specify the first card, 81-160 the second, and 161-240 the third.

If necessary, the ISRCH program may be used to obtain the field dictionary for reference. It is important that a user consider

the full range of values for a field when specifying output columns because data loss may result from truncation of characters when too few columns are allocated.

2. User/Program Communication

```
REEL : 114

1 FILE: 16 DIABETES FILE (SGT) 2:38 PM 1/9/1968
 1A CODE: BBN
```

In the above example, 114 specifies the number of the tape reel that is to be used in the writing of the records. (See p. 45.)

```
1 F,C,C PATNO,1,5
2 F,C,C BIRTHDATE,6,15
3 F,C,C "5",8Ø,8Ø
4 F,C,C ⊕
```

The F,C,C stands for Field, Column, Column. The user specifies the name of the field from which a value is to be generated for output (or a literal text in quotation marks) and the column boundaries for the output. He separates the information by commas, as shown in the example. A group name is not acceptable. Columns may range from 1 through 240. A null response terminates the asking of this question and the program begins to write the file's data onto magnetic tape.

When all the items have been written, three end-of-file marks are written onto the magnetic tape and the program reports

X RECORDS PROCESSED

Y CARDS WRITTEN

FILE *Z* ON TAPE

The tape is rewound, and, if the user wishes to write another file, he must call the program again.

If the user interrupts the running of the ISRCDO Program while it is writing the card images onto tape, the program prints out the number of records processed and the number of cards written to that point, and then asks CONTINUE? Anything other than N or NO will cause the program to proceed.

3. Error Checking

The error-diagnostic comment YOU SPECIFIED A GROUP NOT A FIELD-FIX is printed when the input name is discovered to be a group name rather than a field name. The error comment COLUMN NUMBERS MUST BE BETWEEN 1 AND 24Ø INCLUSIVE-FIX is printed when a value outside the range of 1-240 is inserted for a column specification.

4. Miscellaneous Information

a. Spacing

If the number of columns specified is greater than that required by the size of the value of the field, then the value in the card image is left-justified and the remaining, unused, less

significant columns allowed for this particular field's value are filled with blanks.

It is legal for the user to overlap information by column assignment in order to compress data. Thus, suppose that columns 5, 6, and 7 are specified for storing a Town Code field of information, and column 5 is assigned the Sex-field information. Because of the sequence of assignment, the result is that the Punched-Card-Output Program stores the Sex value in column 5 and the right two digits of the Town Code in columns 6 and 7.

TABLE XXIV. Example of the decimal-number truncation into three column positions.

Original Number	Output Number
1Ø.ØØ	1Ø.
121.3	121
1.1Ø1	1.1

If the number of columns assigned by the user is insufficient to contain the entire field value, the less significant, right-hand portion of the value is truncated. See Table XXIV.

b. Dates

A date field requires 10 columns for output and the format is 01/01/1965.

c. Decimals

The decimal points for numerical values will not always fall in the same column for the same field. This is shown in Table XXV,

where the user specified nine columns for decimal-number output.

TABLE XXV. Decimal-number output.

Original Number	Output Number (left-justified)
123ØØØ.ØØØ	123ØØØ.ØØ
.ØØØØØ12467	.ØØØØØ124
123.456ØØ	123.456ØØ

d. Multivalued-Field Output

If all the fields specified for output contain unique information, then there are exactly y cards for each record in the file, y being one, two, or three, depending on the user's column specifications. If both unique and multivalued fields are specified, then y times n cards per record are generated, n being the largest of the numbers of entries for the multivalued fields. For example, if there is one card (80 columns) specified and if two fields of multivalued information, C and D, are specified and if C contains three pieces of information and D contains seven, then seven card images per item are written onto tape. Furthermore, the unique information appears on each of the seven cards, while card images 4-7 have the same value for field C as card image 3.

e. Multiple Card Types

Multiple card types may be established by using ISRCDO repeatedly for each type or by carefully specifying two or three cards at once. In general, an effort should be made to distinguish each card type from others in a file. If possible, certain columns should be established as card-type identifying columns. Thus the user might specify that "1" is to be punched in column 8Ø of each card of the first type, "2" in column 8Ø of each card of the second type, etc.

When a user wishes to restructure the data from his file, he may use the ISRCDO program to preserve his data on magnetic tape. Then he describes his new file structure with ISRDES and assimilates the data that he put on magnetic tape with ISRCDO into his new file structure by using the Card-Input Program.

D. Expunge a File Program—ISREXP

When an ISR file is no longer needed, it may be expunged from the Fastrand drum by using ISREXP. Also, ISREXP is used to make available the drum space of a file that has been dumped onto magnetic tape. (The magnetic-tape dump and reload programs are discussed below.) There are also cases when a user wishes to preserve his file description and expunge only his data items so that he can begin entering fresh data into his file.

ISREXP first requests the user to type in the file number of the file to be expunged. (See p. 33.) Then, if the file is original,

the program asks DATA ONLY? An answer of N or NO is taken to mean that the user wishes to expunge the entire file, including the file structure. The program gives the user one last chance to confirm that he wants the designated file expunged: its last question asks the user if it should proceed to expunge that file.

If the user attempts to interrupt the running of the ISREXP program by hitting the break key, the program will report how many records it has processed to that point and then will continue to expunge the rest of the file. Once Y or YES has been given as the answer to the last question, it is not possible to stop the expunge process.

```
12:Ø7 PM 2/1/68 BBN-17
ISREXP  SGT

1 FILE:  42  SOCIAL PROTEST BOOKS (SGT)  1:17 PM 1/18/1967
 1A  CONF. CODE:  SOC
2 DATA ONLY?  N
3 OK?  Y

12:12PM
```

E. Magnetic-Tape Handling

1. Magnetic-Tape Dump—ISRDUMP

Because the drum storage allocated to ISR files may not hold all the data needed by all the users at one time, it might become necessary to remove files from the drum when they are not in use.

At times, a user may choose to remove his file for long-term storage. A file may be removed from the drum by writing the file onto magnetic tape and then expunging the file from the drum. Files written on magnetic tape can still be accessed by the Search Program (see p. 57). A program to dump an ISR file onto magnetic tape is also useful for providing backup copies of files. ISRDUMP is included in the ISR system to provide these facilities.

The program writes ISR files onto magnetic tape. The files are still preserved on the drum. The magnetic-tape copies of the files are, however, no longer associated with file numbers or with the drum at all.

Before calling ISRDUMP, the user must have a magnetic tape available. It may be a tape containing previously dumped files or a new one that has three end-of-file marks written on it.

The program asks the following questions.

```
REEL NUMBER:
1  FILE NUMBER:
   1A CONFIDENTIAL CODE:
2  COMMENT
```

The first question requests the reel number assigned to the user's tape. (See p. 45.) The next question asks the number of the file to be dumped. (See p. 33.) Any comments given are dumped onto the tape so that at reload time they may be used to help identify the dump.

Upon obtaining the last user entry, ISRDUMP writes the file on tape and then prints the number of records dumped and the number of the file on the tape. The program then reasks FILE NUMBER? allowing the user to dump more than one file in one running of the program. The program is dismissed when the user enters only EOM as the answer to this question.

The amount of time required to dump a file varies, depending on the size of the file and the number of other computer users. A short file on a new tape could be written in a few seconds.

2. Magnetic-Tape Reload Program—ISRLOAD

ISRLOAD is used to load onto the drum files that were stored on magnetic tape by using ISRDUMP. A reloaded file contains the same data items as the original file, but it occupies a different portion of the drum and may be associated with a different file number. While stored on magnetic tape, files have no file number and are referred to only by their place on the tape.

The program asks four questions.

```
1  REEL:
2  FILE NUMBER PREFERRED ON DRUM:
3  FILE NUMBER ON TAPE:
4  LOAD?
```

The response to the first question should be the number of the magnetic-tape reel containing the desired dumped file. (See p. 45.) Question 2 permits users who are accustomed to using one particular file number or who desire a specific file number to state their preference. If nothing is entered, the program assigns the lowest free file number. The third question wants the user to specify the number of the file on the tape that he wants reloaded. When the program finds the tape-file number requested, it types the file title, time and date of last update, time and date of dumping, and the comment (if any) given by the user at dump time. If this is the proper file, the user should

answer YES to the LOAD? question. If not, he should enter N and question 3 will be reasked.

After the file is reloaded, the name and number of the new file and the number of records loaded are printed. The tape is rewound, and the program halts.

F. Card Print Program—ISRCDPR

The Card Print Program provides the ISR system users with a means for listing a file that is stored in punched-card format on magnetic tape. The questions asked and appropriate responses to them are described below.

1 REEL

The first question requests the reel number holding the file to be printed. Any octal number between 0 and 77777 is acceptable. (See p. 45.)

2 FILE

The ordinal number on the magnetic tape of the file to be printed must be entered here. If the number entered is greater than the number of files on the tape, ISRCDPR types the following comment:

```
FILE NUMBER TOO HIGH
THERE ARE x FILES ON THE TAPE
```

where x is the number of files found.

3 CARDS

This question demands the ordinal numbers (inclusive) of the cards that the user wants printed. For example, he may answer

4-2Ø, 3Ø-END

If the tape is read until the end-of-file mark (i.e., the user requested the last card number or END), ISRCDPR types

THERE ARE *x* CARDS IN THIS FILE

where *x* is the number of cards in this file. If the user requests END by itself, i.e., not *n*-END, the last card image will not be printed, but the number of cards in the file will be reported.

4 COLUMNS

Here the user must enter the card columns (inclusive) of the data to be printed. For example, he may say

1-5, 6Ø-71

If the user enters a number greater than 80, the program types FIX and reasks the question.

The following is a sample typescript from ISRCDPR.

```
3:45 PM 1Ø/15/1968 BBN1(Ø1) RRI
ISRCDPR BPC

CARD PRINT PROGRAM

1 REEL  42
2 FILE  1
3 CARDS  1-3, END
4 COLUMNS  1-7, 1Ø-12, 16, 29-33, 68-7Ø

CARD NO.                    COLUMN

            1-7      1Ø-12   16     29-33    68-7Ø

   1      ØØØ99Ø1     AN     I      HMEDX    REN

   2      ØØ4ØØ57     GLI    L      HNEUR    RLM

   3      ØØ97589     CLI    Ø      HMEDX    ECK

THERE ARE 78 CARDS IN THIS FILE

3:47 PM
```

G. Utilities—ISRDATE, ISRLIST, ISRTLIST, ISRSIZE

There are four small utility programs that perform one small specific task each.

1. ISRDATE, as mentioned in the discussion of ISRCH, translates text dates to the correct integer equivalent. Only the integers are meaningful values in derived-field statements for ISRCH. RISDATE ask the user DATE: and will accept dates of the standard formats. (See Appendix C.)

2. ISRLIST allows the user who has forgotten his file number to obtain a list of all files by name and number. The program asks no questions; it simply lists the existing files by number, title, originator's initials, time and date of last update, derivative files (if any), original file (if file is a derivative), and, if a derivative file, the ordering fields. The program terminates automatically when the list is completed.

3. ISRTLIST is used to list the files that have been dumped onto a magnetic tape. The program asks the user to specify the tape reel number and then it lists the files dumped with ISRDUMP. This is useful if a user has forgotten what file on his tape he wishes to load or magnetic-tape search.

4. ISRSIZE is a program that reports the total file size and the size of each component of any ISR file. The program operates by asking the user to specify his file number. After the program types the size summary, it asks the user for the next file that he is concerned with. The user can terminate the program by entering only an EOM in answer to the question requesting file number. This program is useful for users who wish to keep track of the amount of drum space that their data files are using. It is also useful when the space for ISR files on the Fastrand drum is nearly full and it becomes necessary to determine which files may be dumped onto magnetic tape and expunged from the drum to make room.

APPENDICES

APPENDIX A. Use of the Program Questionnaire

Question Numbering

All questions have numbers, although they are sometimes not printed. Question numbers are of the form 1, 2, 3, 1A, 1B, 1C, 1A2, 1A3. Each number consists of one to six alternating groups of digits and letters; the numbers run from 1 to 64; and the letter groups run from A through Z then AA, BB through ZZ then AAA through KKK. The number of these groups is sometimes referred to as the "level" of the question number. Each question number is one higher on the current level than the previous question (e.g., 1B follows 1A), unless the level of question changes.

Questions

Questions are usually printed on a new line, preceded by their number and indented in proportion to their level to produce an outline format. Questions have a long and a short form. Most programs begin printing in short form when called. The user may type ←/L to cause all succeeding questions to be printed in long form; short form printing may be returned to with ←/S. The experienced user may wish to use "nothing mode" (only question numbers printed), which is initiated by typing ←/. Typing ←HOW produces a brief explanation of the current question or an example of a legitimate answer.

Comments

All printouts other than questions are termed comments. Comments sometimes, but not always, have a long and a short form.

Answers

After a question, the program usually waits for an answer to be typed. All entries are terminated by the ENTER key (EOM). If typing errors are made, the character \ (shift L) may be used to delete the previous character, or the rubout key (which prints as #) may be used to erase the whole answer and cause the question to be reasked. If the answer is unacceptable, FIX or some other comment is printed and the question reasked. Here, and in any other case where the same question is asked twice in a row, only the question number is printed the second time.

Typist Control Commands

In addition to answers to the questions, there are a number of control commands which may be entered. All begin with the character ←. They may not include spaces. Typeins that include the character ← but which are not commands or similar to some command are treated as answers to the question; thus some programs have special commands beginning with ←.

The standard commands and their meanings follow. The character # is used here to represent any legal question number, such as 2 or 5A3.

←/, ←/S, ←/L

Control the form of questions and comments as discussed above.

←HOW

Types an explanation of how to answer the question, then reasks the question.

← -

Halts program immediately.

←CRASH

Halts program immediately, prints PROGRAM CANNOT CONTINUE, and saves the partially run program in the computer. This command may be used to save a malfunctioning program for later examination by BBN personnel.

←C#

Copies the answer to the specified question as the answer to the current question. The program also prints the specified answer after the current question.

←K#,#,#...

←K followed by one or more question numbers separated by commas causes the specified question(s) to be reasked. This command Kills or renders unanswered the questions; the typist must type in an answer for each killed question even if it is the same as before. If more than one number is given with the ←K, the

questions are reasked in the order originally asked, regardless of the order in which the numbers were given by the user.

←R#,#,#...

←R followed by one or more question numbers separated by commas is used to "Reconsider" answers. Reconsidering means that the question is reasked and the old answer printed. The user may type in a new answer or the command ←` to leave the old answer. If more than one number is given, the questions are reconsidered in the order originally asked. After the reconsidering is completed, the current question is reasked.

←#,#,#...

At present, ← followed by a list of question numbers does the same thing as ←R followed by a list of question numbers.

←N

←N causes the program to start Nulling questions—i.e., null answers are provided to all questions (as though the user hit the ENTER key without typing anything). Nulling continues until a question which will not accept a null answer is reached. During nulling, nothing is printed, so the next question seen by the typist is the first one which will not accept a null answer (this may be the question at which ←N was typed). Thus, the ←N command can be used to skip quickly over a group of questions the user does not wish to answer. If the program comes to a previously

answered question while nulling, nulling ceases. Thus if ←2 were typed at question 8, then ←N typed at question 2, only question 2 will be nulled, because questions between 2 and 8 were previously answered.

←L and variations

←L is used to List some or all of the completed questions and answers. Listing means that questions, answers, and comments are printed in their current form, disregarding any rubouts, reconsiderations, changes or copying of previous answers that may have occurred.

←L alone lists everything from the beginning to the current question.

Sections of the program may be listed as follows.

←L#	lists specified question.
←L#,#,#...	lists all questions given, in order originally asked.
←L#-#	lists between specified questions, inclusive.
←L#-	lists from specified question to current question.

Combinations may be strung together with commas, as ←L1,3A-5B2,7-. The experienced user may wish to take advantage of the fact that in ←L#-#, listing starts at the first number and stops after the second, regardless of which number is higher. Thus, ←L,5-2A means list everything except questions after 2A and before 5.

Interrupts and Error Comments

If the user presses the SIGNAL (BREAK) key, INTERRUPTED is printed and the current question reasked. Depending when the key is pressed, this may be the previous question or the question about to be asked. A few programs may have a long delay before responding.

The question-asking routines have three error comments which occur as a result of mistyped reconsider, copy, or list commands. Each is preceded by a question number or whatever was typed where a question number was expected:

IS NOT A LEGAL QUESTION NUMBER.
e.g., ←L12345 typed.

HAS NOT BEEN ASKED.
question number given has not been used.

HAS NOT BEEN ANSWERED.
e.g., user tried to reconsider the current question.

If the comment applies to only one entry in a list of numbers, the rest of the numbers are processed normally. After processing of the correct numbers, if any, the question is reasked.

There are also the following error comments for cases when the storage capacity of the program is exceeded. In normal use, these should never occur. (The storage sizes vary with each program.)

ANSWER TOO LONG.
Current answer too long. Question reasked.

TOO MUCH TEXT.
Space for answer storage full. Program halts.

TOO MANY QUESTIONS.
Program halts.

INTERNAL ANSWER TOO LONG.
A program error. Program halts.

Finally,

JBH FASTRAND ERROR.

Indicates a program or system error involving the Fastrand drum. Program crashes.

APPENDIX B. ISRCOMP Reserved Words

The following words cannot be used as field names because they are functions, operators, or keywords to the ISRCOMP Program.

ALL
ALPH
AND
AS
ATN
COND
COS
CTNS
DDT
DICT
DP
DTM
ELSE
EVAL
EXP
EXT
FCN
FCNS
FIELD
FIELDS
FILE
FILES
FN
FOR
FORM
FP
IF
IP
IS
LN
LOC
LOG
MAX
MIN
NOT
NST
NUM
OK
ON
OR
PART
PARTS
RAN
RISDIC
SGN
SIN
SQRT
STEP
STEPS
STL
STP
TAB
TITLE
TITLES
UNTIL
WHILE
XP

APPENDIX C. Allowable Input Formats for a Date Field

Input	Meaning
3/6	3/6/current year
3/6/64	3/6/1964
3/6/1864	3/6/1864
3/6/ØØ	3/6/1900
T	Today (day that the program started)
T1	Tomorrow
T2	Day after tomorrow
T*n*	*n* days hence
Y1	Yesterday
Y2	Day before yesterday
Y*n*	*n* days ago

APPENDIX D. File Checkin

Some error checking and error comments are common to all ISR programs. First, when a file number is specified by the user, the program checks to make sure that such a file exists. If there is no such file, NO FILE PRESENT is printed, and the question that asks file number is reasked. Second, if a user attempts to use ISRCDI, ISRTTI, ISRCDO, or ISRDUMP on a derivative file, FILE IS NOT ORIGINAL - DERIVED FROM FILE # - is printed and the program halts.

Further, before a user may access a file, that file must be checked to ensure that no other program is using that file in a way that would injure the file or the running of either program if both programs were to access the same file simultaneously. Table D-I shows which programs permit more than one user to access the same file simultaneously and which do not. One of two messages will be printed when a user attempts to access a file that is already being used in a way that does not allow the new program to run safely: FILE IS BEING MODIFIED ON TELETYPE(S) __,__; FILE IS BEING USED ON TELETYPES(S)__,__. When the message says USED, it indicates that one or more users are running ISRCH, ISRSHF, ISRCDO, or ISRLOAD on the file. When any of the other programs is running, the message says MODIFIED. The Teletype numbers are printed so that the would-be user can contact the user(s) if he wishes.

If the file specified has other files derived from it, the derivative files are also checked. If any of the derivatives is being used by a noncompatible program, its number and name will be printed followed by the FILE IS BEING USED... message.

TABLE D-1. Sharing files. Y indicates that the two programs may refer to the same file at the same time. N indicates that the later program will not be allowed to refer to the common file, but will type out a message and halt.

	ISRDES	ISRTTI	ISRCDI	ISRCH	ISRDCH	ISRSHF (O)	ISRSHF (D)	ISRCDO	ISREXP	ISRDUMP	ISRLOAD
ISRDES	N	N	N	N	N	N	N	N	N	N	N
ISRTTI	N	N	N	N	N	N	N	N	N	N	N
ISRCDI	N	N	N	N	N	N	N	N	N	N	N
ISRCH	Y	N	Y	N	Y	N	Y	N	Y	N	
ISRDCH	N	N	N	N	N	N	N	N	N	N	
ISRSHF (original file)	Y	N	Y	N	Y	N					
ISRSHF (derived file)	N	N	N	N	N						
ISRCDO	Y	N	Y	N							
ISREXP	N	N	N								
ISRDUMP	Y	N									
ISRLOAD	N										