DFSORT/VSE

Getting Started with DFSORT/VSE

Version 3 Release 4

SC26-7101-02

DFSORT/VSE

IBM

Getting Started with DFSORT/VSE

Version 3 Release 4

SC26-7101-02

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page vii.

Third Edition (May 1998)

This edition replaces and makes obsolete the previous edition, SC26-7101-01. The technical changes for this edition are summarized under "Summary of Changes," and are indicated by a vertical bar to the left of a change. A vertical bar to the left of a figure caption indicates that the figure has changed. Editorial changes that have no technical significance are not noted.

This edition applies to Version 3 Release 4 of DFSORT/VSE, Program Number 5746-SM3, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for reader's comments is provided at the back of this publication. If the form has been removed, address your comments to:

International Business Machines Corporation
RFC Processing Department
G26/026
5600 Cottle Road
San Jose, CA, 95193-0001
United States of America

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Notices . . . vii
Trademarks . . . vii

Preface . . . ix
About This Book . . . ix
DFSORT/VSE Publications . . . x
  DFSORT/VSE Library Softcopy Information . . . x
DFSORT/VSE on the World Wide Web . . . x
Related Publications . . . x
Referenced Publications . . . xi

Summary of Changes . . . xiii
Third Edition, May 1998 . . . xiii
  New Programming Support for Release 4 . . . xiii

Chapter 1. What is DFSORT/VSE? . . . 1
Sorting Files . . . 1
Merging Files . . . 2
Copying Files . . . 2
What Else Can You Do with DFSORT/VSE? . . . 2
Creating and Running DFSORT/VSE Jobs . . . 3
  Writing Jobs . . . 3
  Running Jobs . . . 3
Using the Sample Bookstore Files . . . 3
Creating Your Sample Input Files . . . 5

Part 1. Learning to Write JCL and DFSORT/VSE Control Statements . . . 7

Chapter 2. Sorting, Merging, and Copying Files . . . 9
Sorting Files . . . 9
  Sorting by Multiple Fields . . . 11
  Continuing a Statement . . . 13
  Sorting Files with the JCL EXEC Statement . . . 13
Merging Files . . . 15
  Writing the MERGE Control Statement . . . 16
  Merging Files with the JCL EXEC Statement . . . 17
Copying Files . . . 18
  Specifying COPY on the SORT or MERGE Statement . . . 18
  Copying Files with the JCL EXEC Statement . . . 19

Chapter 3. Tailoring the Input File with INCLUDE or OMIT . . . 21
Writing the INCLUDE Statement . . . 21
Writing the OMIT Statement . . . 24
  Allowable Comparisons for INCLUDE and OMIT . . . 25
  Writing Constants . . . 26

Chapter 4. Summing Records . . . 27
Writing the SUM Statement . . . 27
Suppressing Records with Duplicate Control Fields . . . 29

Handling Overflow 29

Chapter 5. Reformatting Records 31
Reformatting Records After Sorting 31
Reordering Fields to Reserve Space 32
Inserting Binary Zeros 34
Inserting Blanks 34
Inserting Constants 35
Setting Up the Report Format 36
| Using Edit and Lookup Features 38
Reformatting Records Before Sorting 42
Using Other Statements with INREC 42
Preventing Overflow When Summing Values 43

Chapter 6. Calling DFSORT/VSE from a Program 45
Passing Control Statements 45
Calling DFSORT/VSE from a COBOL Program 45
Sorting Records 45
Merging Records 47
Sorting with COBOL/VSE FASTSRT 49

Chapter 7. Overriding Installation Defaults 51
Writing an OPTION Control Statement 51

Chapter 8. Using DFSORT/VSE Efficiently 53
Be Generous with Virtual Storage 53
Enable Dataspace Sorting and Getvis Sorting 53
Use High–Speed Disks 53
Eliminate Unnecessary Fields with INREC 54
Eliminate Unnecessary Records with INCLUDE or OMIT 54
Reduce File Size with STOPAFT and SKIPREC 54
Consolidate Records with SUM 54
Run DFSORT/VSE with JCL 55
Use FASTSRT with COBOL/VSE 55
Avoid Options That Might Degrade Performance 55

Part 2. Learning to Use ICETOOL 57

Chapter 9. Using the ICETOOL Utility 59
ICETOOL Operators 59
Input Files 60
Creating an ICETOOL Job 61
Writing Required JCL Statements 61
ICETOOL Comment and Blank Statements 62
DEFINE Operator Statement 62
Printing Statistics For Numeric Fields 63
Continuing an Operator Statement 66
Statistics For Record Lengths 66
Creating Identical Sorted Files 67
Creating Multiple Unsorted Files 73
Counting Values in a Range 74
Printing Simple Reports 75
Printing Tailored Reports 77

Using Formatting Items 78
Edit Masks 78
Division 80
Leading, Floating and Trailing Characters 80
Printing Sectioned Reports 81
Printing How Many Times Fields Occur 84
Selecting Records by Field Occurrences 85
Complete ICETOOL Job and its Output 87

Part 3. Appendixes 95

Appendix A. Using the DFSORT/VSE Sample Files 97

Appendix B. The Sample Bookstore Files 99
Sample File - SORT.SAMPIN 100
Sample File - SORT.SAMPADD 100

Appendix C. Processing Order of Control Statements 103

Summary of Changes for Previous Releases of DFSORT/VSE 105
Second Edition, February 1997 105
Programming Support for Release 3 105
First Edition, October 1995 106
Programming Support for Release 2 106

Index 109

Notices

Trademarks

Preface

Getting Started with DFSORT/VSE is a user’s book and tutorial for DFSORT/VSE. You should read it if you are new to DFSORT/VSE Licensed Program 5746-SM3 and need to learn the basics of using DFSORT/VSE to process files. Experienced DFSORT/VSE users can use this book as a general guide to DFSORT/VSE.

The chapters in this book assume that you have used job control language (JCL) and understand how to work with files. You should also know what files are available at your site.

About This Book

This book gives you all of the information and instructions you need to build and submit DFSORT/VSE jobs. You can use DFSORT/VSE by writing JCL and DFSORT/VSE control statements.

New users should work through *Getting Started with DFSORT/VSE* from cover to cover. Each task explained in this book builds on knowledge gained in previous tasks. The Table of Contents lists the main tasks, and summaries are included in the chapters. If you have previous experience with these tasks, you can proceed from here directly to the tutorials that begin in Chapter 2, “Sorting, Merging, and Copying Files” on page 9.

Chapter 1, “What is DFSORT/VSE?” is an overview of the basic principles of sorting, merging, and copying, and explains how to create and use the sample bookstore files for the examples in this book.

Part 1, “Learning to Write JCL and DFSORT/VSE Control Statements,” Chapters 2-8, shows you how to create and process DFSORT/VSE jobs by writing JCL and DFSORT/VSE control statements to sort, merge, and copy files. These chapters also detail DFSORT/VSE's methods for arranging files and generating reports.

Part 2, “Learning to Use ICETOOL,” shows you how to create and process ICETOOL jobs by writing JCL and ICETOOL statements. ICETOOL is a multipurpose DFSORT/VSE utility that uses the capabilities of DFSORT/VSE to perform multiple operations on one or more files in a single step.

Several appendixes and an index follow the chapters.

DFSORT/VSE Publications

Getting Started with DFSORT/VSE is a part of a more extensive DFSORT/VSE library. The additional books in the library are listed below.

Task	Publication Title	Order Number
Application programming	*DFSORT/VSE Application Programming Guide*	SC26-7040
Diagnosing failures and interpreting messages	*DFSORT/VSE Messages, Codes and Diagnosis Guide*	SC26-7132
Evaluating DFSORT/VSE	*DFSORT/VSE General Information*	GC26-7039
Planning for, installing, customizing, and tuning DFSORT/VSE	*DFSORT/VSE Installation and Tuning Guide*	SC26-7041
Quick reference	*DFSORT/VSE Reference Summary*	SX26-6008

You can order a complete set of DFSORT/VSE publications with the order number SBOF-6130, except for *DFSORT/VSE Licensed Program Specifications* (GC26-7038), which must be ordered separately.

DFSORT/VSE Library Softcopy Information

A softcopy version of the DFSORT/VSE library is available on the CD-ROM shown in the table that follows. The *IBM Online Library VSE Collection* contains all of the DFSORT/VSE books for Releases 2, 3, and 4, with the exception of the *DFSORT/VSE Reference Summary*, and books from other VSE libraries.

Order Number	Title
SK2T-0060	*IBM Online Library VSE Collection*

DFSORT/VSE on the World Wide Web

For news, tips, and examples, visit the DFSORT/VSE home page at URL:

`http://www.ibm.com/storage/dfsortvse/`

Related Publications

In the course of learning how to use DFSORT/VSE, you might also want to refer to the books listed in the table below.

Short Title	Publication	Order Number
JCL Reference	*VSE/ESA System Control Statements (for VSE/ESA Version 1 Release 3)*	SC33-6513
	VSE/ESA System Control Statements (for VSE/ESA Version 2)	SC33-6613
	VSE/VSAM User's Guide	SC33-6535
JECL Reference	*VSE/POWER Administration and Operation (for VSE/ESA Version 1 Release 3)*	SC33-6571
	VSE/POWER Administration and Operation (for VSE/ESA Version 2)	SC33-6633

Referenced Publications

Within the text of this document, references are made to the following books:

Short Title	Publication	Order Number
Application Programming Guide	*DFSORT/VSE Application Programming Guide*	SC26-7040
Installation and Tuning Guide	*DFSORT/VSE Installation and Tuning Guide*	SC26-7041

A more comprehensive list of related publications appears in *DFSORT/VSE Application Programming Guide*.

For more information on using DFSORT/VSE with COBOL or PL/I, see the Programmer's Guide describing the compiler version available at your site.

Summary of Changes

Third Edition, May 1998

New Programming Support for Release 4

DFSORT/VSE Version 3 Release 4 continues the strategy of providing performance improvements and productivity features. These improvements and features are described in more detail in the subsections that follow.

Performance

Performance enhancements for DFSORT/VSE Version 3 Release 4 include the following:

- Improved data processing methods for:
  - Dataspace and getvis sorting applications using work space
  - Merge and copy applications
- Improved input/output processing techniques for:
  - SAM output files
  - Non-VSAM input and output files
  - VSAM (and SAM ESDS accessed as VSAM) input and output files
  - Work files
- Improved ECKD disk device support for input, output, and work files by using the ECKD command set.
- New VSAMBSP installation option which allows users to control the number of buffers DFSORT/VSE can use for VSAM (or SAM ESDS accessed as VSAM) input and output file processing.
- Improved work file processing:
  - All work files are now closed at the end of an application.
  - Additional work file extents can now be used, if available, when end of extent is encountered regardless of whether STXIT is in effect.
  - All extents of an SD work file can now be used instead of only the first extent.

Productivity

Additional Year 2000 Formats: New formats give users more flexibility in sorting, merging, or transforming two-digit year dates:

- Y2S interprets two-digit character year data according to the century window and allows special handling of indicators X'00' (binary zeros), X'40' (EBCDIC blanks), X'20' (ASCII blanks) and X'FF' (binary ones) in the year field.
- Y2B interprets two-digit binary year data according to the century window.

OUTREC Enhancements: The OUTREC control statement supports the following new features:

- Sophisticated editing capabilities such as hexadecimal display and control of the way numeric fields are presented with respect to length, leading or suppressed zeros, symbols (for example, the thousands separator and decimal point), leading and trailing positive and negative signs, and so on. Twenty-six pre-defined editing masks are available for commonly used numeric editing patterns, encompassing many of the numeric notations used throughout the world. In addition, a virtually unlimited number of numeric editing patterns are available via user-defined editing masks.

- Selection of a character or hexadecimal string for output from a lookup table, based on a character, hexadecimal, or bit string as input (that is, lookup and change).

INCLUDE/OMIT Enhancements: The following INCLUDE/OMIT enhancements are supported:

- DFSORT/VSE can now handle a significantly larger number of INCLUDE and OMIT conditions.
- ALL and NONE allow users to include or omit all records.

ZDPRINT Option: With the new ZDPRINT installation and run-time options, users can choose to have summed (totalled) positive zoned decimal fields converted to printable numbers.

Online Message Explanations Support: New Online Message Explanations (OME) allow users to request an explanation of a DFSORT/VSE message. The message explanation is displayed on the console.

Additional Enhancements

The following additional enhancements are supported:

- The IBM-supplied default has been changed from STXIT=YES to STXIT=MIN. The STXIT=MIN installation option and the MINSTXIT run-time option allow users to specify that DFSORT/VSE should use its STXIT routine for abend recovery processing, **not** restoring its STXIT every time control is returned from a user exit routine. Unlike STXIT=YES (or STXIT), STXIT=MIN (or MINSTXIT) does not degrade performance when COBOL or PL/I programs invoke DFSORT/VSE and use E15/E35 user exit routines to process records.
- The DIAGINF installation option provides a new DFSORT/VSE capability that allows users to request diagnostic information (diagnostic messages and a dump), regardless of the options in effect at run time.
- The new NRECOUT installation and run-time option allows users to specify the action DFSORT/VSE should perform when it does not write any records to the output file. This gives users control over the action (continue or terminate), type of message (informational or error), and return code (0,4 or 16) when no records are written to the output file.

Chapter 1. What is DFSORT/VSE?

The DFSORT/VSE licensed program is a high-performance data arranger developed by IBM for VSE/ESA users.

With DFSORT/VSE, you can sort, merge, and copy files. You can use DFSORT/VSE to do simple tasks such as alphabetizing a list of names, or you can use it to aid complex tasks such as taking inventory or running a billing system. You can also use DFSORT/VSE's record-level editing capability to perform data management tasks.

The information you process with DFSORT/VSE is contained in *files*. The term *file* refers to a data set that contains one or more records. Any named group of records is called a *data set*. The terms *data set* and *file* are synonymous; however, for the sake of consistency, this book refers only to files.

A file contains the information that you want to sort, copy, or merge. For most of the processing done by DFSORT/VSE, the whole file is affected. However, some forms of DFSORT/VSE processing involve only certain individual records in that file.

Throughout this book, the term *record* refers to a collection of related information used as a unit, such as one item in a data base or personnel data about one member of a department. The term *field* refers to a specific portion of a record used for a particular category of data. A field is the smallest addressable unit of data in a file.

Sorting Files

You can use DFSORT/VSE to rearrange the records in your files. *Sorting* is arranging records in either ascending or descending order within a file. Figure 1 shows a sample file of names, first sorted in ascending order, then in descending order.

Figure 1. DFSORT/VSE Arranges Information in Ascending and Descending Order

Unsorted File	Sorted Ascending	Sorted Descending
Andy	Andy	Edward
Edward	Betty	Dan
Carol	Carol	Carol
Dan	Dan	Betty
Betty	Edward	Andy

The fields in the records can be in any of these formats: EBCDIC character, decimal, or binary. All of the examples in this book use the EBCDIC formatting sequence (the standard DFSORT/VSE collating sequence).

You can sort data in several different formats. Figure 2 shows the most common data formats and the codes you use to specify them.

Figure 2. Data Format Codes

Data Format	Code
EBCDIC (Character)	CH
Binary (Numeric)	BI
Zoned Decimal (Numeric)	ZD
Packed Decimal (Numeric)	PD

Refer to *Application Programming Guide* for complete details of the available formats.

Merging Files

You can also use DFSORT/VSE to merge files. DFSORT/VSE *merges* files by combining two or more data sets of sorted records to form a single file of sorted records.

Figure 3. DFSORT/VSE Merges Two Files into One File

File 1	File 2	Merged File
Andy	Amy	Amy
Betty	Chris	Andy
Carol	Sue	Betty
Dan		Carol
Edward		Chris
		Dan
		Edward
		Sue

The files you merge must be previously sorted into the same order (ascending or descending).

Copying Files

DFSORT/VSE can also copy files without any sorting or merging taking place. You copy files in much the same way that you sort or merge them.

What Else Can You Do with DFSORT/VSE?

While sorting, merging, or copying files, you can also:

- Select a subset of records from an input file. You can include or omit records that meet specified criteria. For example, when sorting an input file containing records of course books from many different school departments, you can sort the books for only one department.

- Reformat records, add or delete fields, and insert blanks or binary zeros. For example, you can create an output file that contains only certain fields from the input file arranged differently.
- Sum the values in selected records while sorting or merging (but not while copying). In the example of a file containing records of course books, you can use DFSORT/VSE to add up the dollar amounts of books for one school department.
- Sort, merge, include, or omit records according to the collating rules defined in a selected locale.
- Alter the collating sequence when sorting or merging records (but not while copying). For example, you can have the lowercase letters collate after the uppercase letters.

Creating and Running DFSORT/VSE Jobs

Processing files with DFSORT/VSE involves two steps:

1. Creating a DFSORT/VSE job
2. Running a DFSORT/VSE job

You can run a DFSORT/VSE job by invoking processing in a number of ways:

- With a JCL EXEC statement, using the name of the program or the name of the cataloged procedure
- Within programs written in COBOL, PL/I, or assembler language

In this book, the phrase *JCL-invoked* means that the DFSORT/VSE program is initiated by a JCL EXEC statement. The phrase *program-invoked* means that the DFSORT/VSE program is initiated from another program.

Writing Jobs

You can use DFSORT/VSE by writing JCL and DFSORT/VSE control statements no matter how your site has installed DFSORT/VSE. Part 1 contains instructions on writing the JCL EXEC and DFSORT/VSE control statements. JCL statements are processed by your operating system. They describe your files to the operating system, and initiate DFSORT/VSE processing. DFSORT/VSE control statements are processed by DFSORT/VSE. They describe and initiate the processing you want to do.

Running Jobs

You can run DFSORT/VSE jobs directly with a JCL EXEC statement. Or, you can call DFSORT/VSE from a COBOL, assembler, or PL/I program.

Using the Sample Bookstore Files

Before you begin, turn to Appendix B, “The Sample Bookstore Files” on page 99. Many of the examples in this book refer to the sample bookstore files as the input files, so you should become familiar with them. The input files contain the data that you want arranged or sorted. You must specify an input file for every DFSORT/VSE job you run. The sample bookstore files are input files named **SORT.SAMPIN** and **SORT.SAMPADD**.

Each record in the bookstore file has 12 fields (book title, author's last name, and so on). A record can be represented by one horizontal row on the page. A field can be represented by one vertical column on the page.

To sort a file, you choose one or more fields that you want to use to order the records (arrange in ascending or descending order). These fields are called *control fields* (or, in COBOL, *keys*).

As you work through the exercises on the following pages, remember that each *entire* record is sorted, not just the control field. However, for the sake of simplicity, the figures in the text show only the control fields being discussed. The sorted records actually contain all the fields, but one page is not wide enough to show them. Appendix B, "The Sample Bookstore Files," is printed to show all the fields in each record. It is also arranged with headings and numbers that show the byte positions of each field. The numeric fields are in binary format (see Figure 2 on page 2) and therefore will not appear on most terminals as they do in this book. The methods used to arrange and view the data are explained in the chapters on DFSORT/VSE functions that follow.

Figure 4 shows an example of sorted fields. Notice the line of numbers above the sorted fields. These numbers represent the byte positions of those fields. You use byte positions to identify fields to DFSORT/VSE. The examples show the byte positions to help you while you are learning to use DFSORT/VSE. The byte positions do not actually appear in any of your processed files.

In Figure 4, the first two records, which show nothing in the course department fields, are general purpose books not required for a particular course. For this example, the control field is the Course Department field.

Figure 4. Sample Bookstore File Sorted by Course Department in Ascending Order

Book Title	**Course Department**	**Price**
1 75	110 114	170 173
LIVING WELL ON A SMALL BUDGET		9900
PICK'S POCKET DICTIONARY		295
INTRODUCTION TO BIOLOGY	BIOL	2350
SUPPLYING THE DEMAND	BUSIN	1925
STRATEGIC MARKETING	BUSIN	2350
COMPUTER LANGUAGES	COMP	2600
VIDEO GAME DESIGN	COMP	2199
COMPUTERS: AN INTRODUCTION	COMP	1899
NUMBERING SYSTEMS	COMP	360
SYSTEM PROGRAMMING	COMP	3195
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	ENGL	595
EDITING SOFTWARE MANUALS	ENGL	1450
MODERN ANTHOLOGY OF WOMEN POETS	ENGL	450
THE COMPLETE PROOFREADER	ENGL	625
SHORT STORIES AND TALL TALES	ENGL	1520
THE INDUSTRIAL REVOLUTION	HIST	795
EIGHTEENTH CENTURY EUROPE	HIST	1790
CRISIS OF THE MIDDLE AGES	HIST	1200
INTRODUCTION TO PSYCHOLOGY	PSYCH	2200
ADVANCED TOPICS IN PSYCHOANALYSIS	PSYCH	2600

Also notice that records in Figure 4 with *equally collating control fields* (in this case, the same department) appear in their original order. For example, within the Computer Science department (COMP), the title *Video Game Design* still appears before *Computers: An Introduction*.

You can control whether records with equal control fields appear in their original order or whether DFSORT/VSE orders them randomly. The system programmer sets defaults at installation time that you can change with some DFSORT/VSE options at run-time. The examples in this book assume that the default is for records with equal control fields to appear in their original order.

Note: The examples used in this book are for fixed-length records only. For information on processing variable-length records, see *Application Programming Guide*.

Creating Your Sample Input Files

The sample bookstore files are the input files you will use for most of the examples in this book. Your system programmer created these files when verifying the ICETOOL installation. If these files are no longer available, ask your system programmer to run the sample job, ILUDATA, to create the sample files, SORT.SAMPIN, SORT.SAMPADD, and SORT.BRANCH, which are used in many of the examples in this book.

Note: Some of the examples use files other than SORT.SAMPIN, SORT.SAMPADD, and SORT.BRANCH. You can either create files from scratch to match the ones used in the text, or else perform a similar exercise on files you already have.

Summary

So far in *Getting Started with DFSORT/VSE* you covered the following concepts:

- You can sort, copy, or merge files using DFSORT/VSE.
- You can write the JCL EXEC and DFSORT/VSE control statements to create and process DFSORT/VSE jobs.
- You can run DFSORT/VSE jobs directly or call DFSORT/VSE from a program.

In addition, this chapter covered how to use and read the sample bookstore file provided with DFSORT/VSE, and how to use the sample input and output files. Now continue with tutorials on how to write DFSORT/VSE control statements.

Part 1. Learning to Write JCL and DFSORT/VSE Control Statements

The seven chapters in this section explain how to write JCL EXEC statements and DFSORT/VSE control statements to process your files.

Chapter 2. Sorting, Merging, and Copying Files 9

Chapter 3. Tailoring the Input File with INCLUDE or OMIT 21

Chapter 4. Summing Records 27

Chapter 5. Reformatting Records 31

Chapter 6. Calling DFSORT/VSE from a Program 45

Chapter 7. Overriding Installation Defaults 51

Chapter 8. Using DFSORT/VSE Efficiently 53

Chapter 2. Sorting, Merging, and Copying Files

This tutorial shows you how to sort, merge, and copy files by writing DFSORT/VSE control statements that are processed with JCL.

DFSORT/VSE control statements are input in the JCL used to run DFSORT/VSE. To keep the instructions simple, the control statements are covered first and the related JCL statements are explained afterward. For most of the tutorials you will concentrate on JCL-invoked DFSORT/VSE, running DFSORT/VSE with JCL. Information on calling DFSORT/VSE from a program (program invocation) is presented in Chapter 6, “Calling DFSORT/VSE from a Program” on page 45.

Note: During the first few tutorials, you might want to refer to Appendix B, “The Sample Bookstore Files” on page 99, which contains the sample bookstore files. That way you can refer to the correct byte numbers.

Sorting Files

To run DFSORT/VSE with the JCL EXEC statement, write SORT and RECORD control statements to describe the control fields, the order in which you want them sorted, and the record format. The control statements you write are part of the input stream read from SYSIPT.

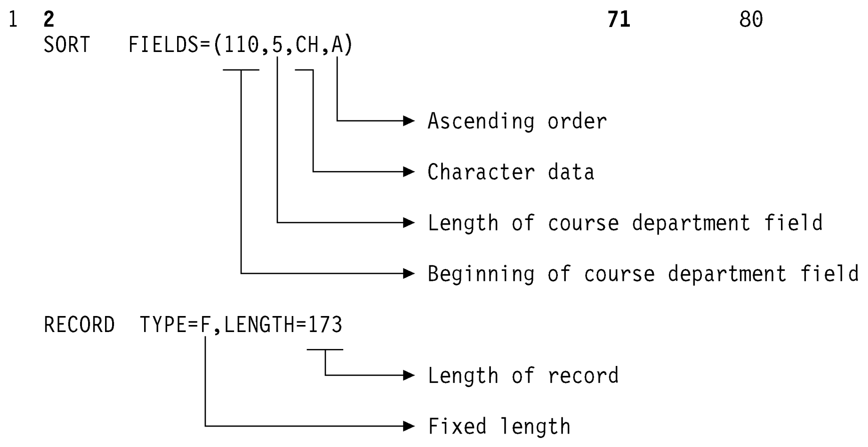
As you will see in later chapters, you can use SORT and RECORD with the other DFSORT/VSE control statements.

To write a SORT statement that sorts the bookstore records by the course department field (as shown in Figure 6 on page 10):

Figure 5. Steps to Create the SORT and the RECORD Statements to Sort by Department

Step	Action
1	Leave at least one blank and type **SORT**
2	Leave at least one blank and type **FIELDS=**
3	Type, in parenthesis and separated by commas: 1. Where the course department field begins, relative to the beginning of the record in the bookstore file (the first position is byte 1). The course department field begins at byte **110**. 2. The length of the department field in bytes. The department field is **5** bytes long. 3. A code for the data format. The department field contains character data, which you specify as **CH**. (Figure 2 on page 2 shows the codes for the most common data formats.) 4. The letter **A**, for ascending order.
4	On the next line leave at least one blank and type **RECORD**
5	Leave at least one blank and type **TYPE=**
6	Type the letter **F**, for fixed length records
7	Type the comma and **LENGTH=**
8	Type the record length. Bookstore file record length is **173**.

Make sure that the statement is coded between columns 2 and 71. Your control statement should look like this:



```
1 2                                               71        80
   SORT   FIELDS=(110,5,CH,A)

   RECORD  TYPE=F,LENGTH=173
```

Ascending order
Character data
Length of course department field
Beginning of course department field
Length of record
Fixed length

Remember that although Figure 6 shows only certain fields, the displayed fields are not the only ones in the output file.

Figure 6. Sample Bookstore File Sorted by Course Department in Ascending Order

Book Title		**Course Department**
1	75	110 114
LIVING WELL ON A SMALL BUDGET		
PICK'S POCKET DICTIONARY		
INTRODUCTION TO BIOLOGY		BIOL
SUPPLYING THE DEMAND		BUSIN
STRATEGIC MARKETING		BUSIN
COMPUTER LANGUAGES		COMP
VIDEO GAME DESIGN		COMP
COMPUTERS: AN INTRODUCTION		COMP
NUMBERING SYSTEMS		COMP
SYSTEM PROGRAMMING		COMP
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		ENGL
EDITING SOFTWARE MANUALS		ENGL
MODERN ANTHOLOGY OF WOMEN POETS		ENGL
THE COMPLETE PROOFREADER		ENGL
SHORT STORIES AND TALL TALES		ENGL
THE INDUSTRIAL REVOLUTION		HIST
EIGHTEENTH CENTURY EUROPE		HIST
CRISES OF THE MIDDLE AGES		HIST
INTRODUCTION TO PSYCHOLOGY		PSYCH
ADVANCED TOPICS IN PSYCHOANALYSIS		PSYCH

To sort the records in descending order, specify **D** instead of A. For example, to sort the prices for each book in descending order, type:

```
SORT  FIELDS=(170,4,BI,D)
                ─┬─     │
                 │      └────► Descending order
                 │
                 └───────────► Price

RECORD  TYPE=F,LENGTH=173
             │        ─┬─
             │         └─────► Length of record
             │
             └───────────────► Fixed length
```

The sort order is bytes 170 through 173 as binary data in descending sequence. Figure 7 shows the results of the sort in descending order.

Figure 7. Sample Bookstore File Sorted by Price in Descending Order

Book Title		**Price**
1	75	170 173
LIVING WELL ON A SMALL BUDGET		9900
SYSTEM PROGRAMMING		3195
COMPUTER LANGUAGES		2600
ADVANCED TOPICS IN PSYCHOANALYSIS		2600
STRATEGIC MARKETING		2350
INTRODUCTION TO BIOLOGY		2350
INTRODUCTION TO PSYCHOLOGY		2200
VIDEO GAME DESIGN		2199
SUPPLYING THE DEMAND		1925
COMPUTERS: AN INTRODUCTION		1899
EIGHTEENTH CENTURY EUROPE		1790
SHORT STORIES AND TALL TALES		1520
EDITING SOFTWARE MANUALS		1450
CRISES OF THE MIDDLE AGES		1200
THE INDUSTRIAL REVOLUTION		795
THE COMPLETE PROOFREADER		625
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		595
MODERN ANTHOLOGY OF WOMEN POETS		450
NUMBERING SYSTEMS		360
PICK'S POCKET DICTIONARY		295

Sorting by Multiple Fields

You can further sort the records in the bookstore file by specifying multiple control fields. When you specify two or more control fields, you specify them in the order of greater to lesser priority. Note that control fields might overlap or be contained within other control fields.

Figure 8 on page 12 shows how the records would be sorted if you specified the following control fields in the order they are listed:

1. Course department
2. Course number
3. Instructor's last name
4. Instructor's initials
5. Book title

So, if two records have the same department, they are sorted by course number. If they also have the same course number, they are sorted by instructor's last name. If they also have the same last name, they are sorted by initials. Finally, if they also have the same initials, they are sorted by title.

Specify the location, length, data format, and order for each of the control fields, as follows:

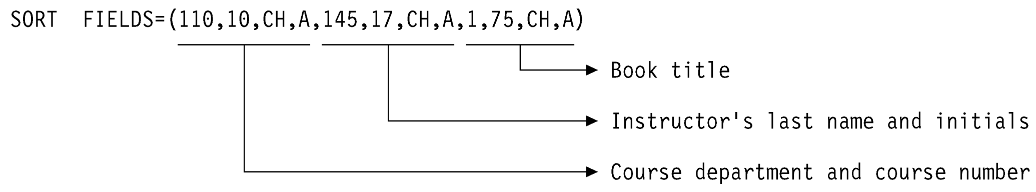
```
 SORT  FIELDS=(110,5,CH,A,115,5,CH,A,145,15,CH,A,160,2,CH,A,1,75,CH,A)
                                                              Book title
                                                              Instructor's initials
                                                              Instructor's last name
                                                              Course number
                                                              Course department

 RECORD  TYPE=F,LENGTH=173
                                                              Length of record
                                                              Fixed length
```

The records are sorted as shown in Figure 8.

Figure 8. Sample Bookstore File Sorted by Multiple Fields

Book Title	Course Department	Course Number	Instructor's Last Name	Instructor's Initials
1 75	110 114	115 119	145 159	160 161
LIVING WELL ON A SMALL BUDGET				
PICK'S POCKET DICTIONARY				
INTRODUCTION TO BIOLOGY	BIOL	80521	GREENBERG	HC
STRATEGIC MARKETING	BUSIN	70124	LORCH	HH
SUPPLYING THE DEMAND	BUSIN	70251	MAXWELL	RF
NUMBERING SYSTEMS	COMP	00032	CHATTERJEE	AN
COMPUTER LANGUAGES	COMP	00032	CHATTERJEE	CL
COMPUTERS: AN INTRODUCTION	COMP	00032	CHATTERJEE	CL
SYSTEM PROGRAMMING	COMP	00103	SMITH	DC
VIDEO GAME DESIGN	COMP	00205	NEUMANN	LB
SHORT STORIES AND TALL TALES	ENGL	10054	BUCK	GR
EDITING SOFTWARE MANUALS	ENGL	10347	MADRID	MM
THE COMPLETE PROOFREADER	ENGL	10347	MADRID	MM
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	ENGL	10856	FRIEDMAN	KR
MODERN ANTHOLOGY OF WOMEN POETS	ENGL	10856	FRIEDMAN	KR
THE INDUSTRIAL REVOLUTION	HIST	50420	GOODGOLD	ST
CRISES OF THE MIDDLE AGES	HIST	50521	WILLERTON	DW
EIGHTEENTH CENTURY EUROPE	HIST	50632	BISCARDI	HR
INTRODUCTION TO PSYCHOLOGY	PSYCH	30016	ZABOSKI	RL
ADVANCED TOPICS IN PSYCHOANALYSIS	PSYCH	30975	NAKATSU	FL

You can often shorten the length of control statements. You can specify fields together whenever they are next to each other and have the same data format. You can shorten this last statement by specifying the department and course number together as one field, and the instructor's last name and initials together as one field.



```
SORT  FIELDS=(110,10,CH,A,145,17,CH,A,1,75,CH,A)
```

Book title
Instructor's last name and initials
Course department and course number

Also, if all the control fields have the same data format, you can specify the data format just once, using the FORMAT= parameter. For example:

```
SORT FIELDS=(110,10,A,145,17,A,1,75,A),FORMAT=CH
```

Continuing a Statement

If you cannot fit your SORT statement (or any other DFSORT/VSE control statement) between columns 2 through 71, you can continue it on the next line. If you end a line with a comma followed by a blank, DFSORT/VSE treats the next line as a continuation. The continuation can begin anywhere between columns 2 through 16.

For example:

```
SORT FIELDS=(110,10,A,145,17,A,
             1,75,A),FORMAT=CH
```

Sorting Files with the JCL EXEC Statement

The job control language (JCL) you need to do a sort depends on whether you run DFSORT/VSE with the JCL EXEC statement or call DFSORT/VSE from a program. For now, concentrate on running DFSORT/VSE with the JCL EXEC statement. Information on calling DFSORT/VSE from a program is presented in Chapter 6, “Calling DFSORT/VSE from a Program” on page 45.

Your operating system uses the JCL you supply with your DFSORT/VSE control statements to:

- Identify you as an authorized user.
- Allocate the necessary resources to run your job.
- Run your job.
- Return information to you about the results.
- Terminate your job.

You must supply JCL with every DFSORT/VSE job you submit.

Required JCL includes a JOB statement, an EXEC statement, and several DLBL statements. The statements you need and their exact form depend upon whether you:

- Invoke DFSORT/VSE with an EXEC statement in the input job stream, or with a system macro instruction within another program.
- Choose to use EXEC statement cataloged procedures to invoke DFSORT/VSE.

- Want to use program exits to activate routines of your own.

Information on when you would choose each of the above options is detailed in *Application Programming Guide*.

The JCL statements you need for most jobs are described below.

// JOB jobname	Signals the beginning of a job. At your site, you might be required to specify information such as your name and account number on the JOB statement.
// ASSGN	Assigns the device to be used in an application to the appropriate symbolic name. This is not required for VSAM data sets and for devices that have been previously assigned.
// DLBL VSESPUC	Defines the VSE/VSAM user catalog.
// DLBL SORTINn	Defines the input file.
// DLBL SORTWKn	Defines a work storage file for a sort. For most applications, one work storage file is sufficient. (Increasing the number of work storage files does *not* improve performance.)
// DLBL SORTOUT	Defines the output file.
// EXTENT	Defines direct access device area limits for the application. It is not required for VSAM data sets.
// EXEC	Signals the beginning of a job step and tells the operating system what program to run. To run DFSORT/VSE, write the EXEC statement like this: `//  EXEC SORT,SIZE=64K`

The SYSIPT input is used for DFSORT/VSE control statements.

The SYSLST output is used for DFSORT/VSE output messages.

Below is some sample JCL that will run DFSORT/VSE. It assumes the input and output record lengths are the same.

```
// JOB EXAMP JOBA,PROGRAMMER
// ASSGN SYS001,X'287'       SORT OUTPUT UNIT ADDRESS
// ASSGN SYS003,X'E5A'       SORT WORK STORAGE UNIT ADDRESS
// DLBL VSESPUC,'VSESP.USER.CATALOG',,VSAM
// DLBL SORTIN1,'SORT.SAMPIN',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// DLBL SORTWK1,'SORT.WORK',0,SD
// EXTENT SYS003,,,,500,20
// DLBL SORTOUT,'MASTER',,SD
// EXTENT SYS001,338000,,,580,20
// EXEC SORT,SIZE=64K
    SORT  FIELDS=(110,10,A,1,75,A),FORMAT=CH
    RECORD  TYPE=F,LENGTH=173
/*
/&
```

Application Programming Guide contains additional information on running DFSORT/VSE with the JCL EXEC statement.

So Far

So far in this chapter you covered how to write a SORT control statement and how to run that sort with the JCL EXEC statement. The next tutorial explains how to use the MERGE control statement to merge two files.

Merging Files

Generally, the reason for merging files is to add more records to a file that is already sorted.

For example, assume that the bookstore file is already sorted by course department and book title (as shown in Figure 9), and you want to update it by merging it with a file that contains 22 new records, also sorted by course department and book title.

Figure 9. Sample Bookstore File Sorted by Course Department and Book Title

Book Title		**Course Department**
1	75	110 114
LIVING WELL ON A SMALL BUDGET		
PICK'S POCKET DICTIONARY		
INTRODUCTION TO BIOLOGY		BIOL
STRATEGIC MARKETING		BUSIN
SUPPLYING THE DEMAND		BUSIN
COMPUTER LANGUAGES		COMP
COMPUTERS: AN INTRODUCTION		COMP
NUMBERING SYSTEMS		COMP
SYSTEM PROGRAMMING		COMP
VIDEO GAME DESIGN		COMP
EDITING SOFTWARE MANUALS		ENGL
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		ENGL
MODERN ANTHOLOGY OF WOMEN POETS		ENGL
SHORT STORIES AND TALL TALES		ENGL
THE COMPLETE PROOFREADER		ENGL
CRISES OF THE MIDDLE AGES		HIST
EIGHTEENTH CENTURY EUROPE		HIST
THE INDUSTRIAL REVOLUTION		HIST
ADVANCED TOPICS IN PSYCHOANALYSIS		PSYCH
INTRODUCTION TO PSYCHOLOGY		PSYCH

For this example, use a new file such as the one shown in Figure 10 on page 16.

Figure 10. 22 New Records Sorted by Course Department and Book Title

Book Title		Course Department
1	75	110 114
ANOTHER ITALIAN DICTIONARY		
COMPLETE SPANISH DICTIONARY		
FRENCH TO ENGLISH DICTIONARY		
GUNTHER'S GERMAN DICTIONARY		
GUIDE TO COLLEGE LIFE		
A SMALLER WORLD: MICROBES		BIOL
CELLS AND HOW THEY WORK		BIOL
DNA: BLUEPRINT FOR YOU		BIOL
THE ANIMAL KINGDOM		BIOL
ANTICIPATING THE MARKET		BUSIN
KNOW YOUR CONSUMER		BUSIN
QUEUE THEORY		BUSIN
THE ART OF TAKEOVERS		BUSIN
ZEN BUSINESS		BUSIN
DESIGNING APPLICATIONS		COMP
THE TOY STORE TEST		COMP
NOVEL IDEAS		ENGL
CIVILIZATION SINCE ROME FELL		HIST
POLITICS AND HISTORY		HIST
REBIRTH FROM ITALY		HIST
FREUD'S THEORIES		PSYCH
MAP OF THE HUMAN BRAIN		PSYCH

To merge files, you write a MERGE control statement and several JCL statements. Whenever you merge files, you must make sure that their records have the same format and that they have been previously sorted by the same control fields. You can merge up to 9 files at a time.

As you will see in later chapters, you can also use INCLUDE, OMIT, SUM, and OUTREC control statements for merge applications.

Writing the MERGE Control Statement

The format of the MERGE statement is the same as that of the SORT statement. To merge the bookstore master file with the file containing the 22 new records, write:

```
MERGE  FIELDS=(110,5,A,1,75,A),FORMAT=CH,FILES=2
               ───────  ──────                  │
                  │       │                     └─► Number of files
                  │       │
                  │       └───────────────────────► Book title
                  │
                  └───────────────────────────────► Course department

RECORD  TYPE=F,LENGTH=173
             │        ─┬─
             │         └──────────────────────────► Length of record
             │
             └────────────────────────────────────► Fixed length
```

Figure 11 on page 17 shows the merged output.

Figure 11. Sample Bookstore File Merged with 22 New Records

Book Title		Course Department
1	75	110 114
ANOTHER ITALIAN DICTIONARY		
COMPLETE SPANISH DICTIONARY		
FRENCH TO ENGLISH DICTIONARY		
GUIDE TO COLLEGE LIFE		
GUNTHER'S GERMAN DICTIONARY		
LIVING WELL ON A SMALL BUDGET		
PICK'S POCKET DICTIONARY		
A SMALLER WORLD: MICROBES		**BIOL**
CELLS AND HOW THEY WORK		**BIOL**
DNA: BLUEPRINT FOR YOU		**BIOL**
INTRODUCTION TO BIOLOGY		BIOL
THE ANIMAL KINGDOM		**BIOL**
ANTICIPATING THE MARKET		**BUSIN**
KNOW YOUR CONSUMER		**BUSIN**
QUEUE THEORY		**BUSIN**
STRATEGIC MARKETING		BUSIN
SUPPLYING THE DEMAND		BUSIN
THE ART OF TAKEOVERS		**BUSIN**
ZEN BUSINESS		**BUSIN**
COMPUTER LANGUAGES		COMP
COMPUTERS: AN INTRODUCTION		COMP
DESIGNING APPLICATIONS		**COMP**
NUMBERING SYSTEMS		COMP
SYSTEM PROGRAMMING		COMP
THE TOY STORE TEST		**COMP**
VIDEO GAME DESIGN		COMP
EDITING SOFTWARE MANUALS		ENGL
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		ENGL
MODERN ANTHOLOGY OF WOMEN POETS		ENGL
NOVEL IDEAS		**ENGL**
SHORT STORIES AND TALL TALES		ENGL
THE COMPLETE PROOFREADER		ENGL
CIVILIZATION SINCE ROME FELL		**HIST**
CRISES OF THE MIDDLE AGES		HIST
EIGHTEENTH CENTURY EUROPE		HIST
POLITICS AND HISTORY		**HIST**
REBIRTH FROM ITALY		**HIST**
THE INDUSTRIAL REVOLUTION		HIST
ADVANCED TOPICS IN PSYCHOANALYSIS		PSYCH
FREUD'S THEORIES		**PSYCH**
INTRODUCTION TO PSYCHOLOGY		PSYCH
MAP OF THE HUMAN BRAIN		**PSYCH**

Merging Files with the JCL EXEC Statement

As in a sort, the JCL you need depends on whether you run DFSORT/VSE with the JCL EXEC statement or call it from a program. This chapter only discusses running DFSORT/VSE with the JCL EXEC statement.

The JCL needed for a merge is the same as that for a sort, with the following exceptions:

- You do *not* use the DLBL SORTWKn statement.

- Instead of the DLBL SORTIN1 statement, you use DLBL SORTINn statements to define the input files. You need one DLBL SORTINn statement for each file being merged. The value n in SORTINn is a number from 1 to 9 (minimum and maximum), indicating the number of files to be merged.

To merge the presorted bookstore file and the file containing the new records, code the following JCL statements for this example. The new file is called SORT.SAMPADD and the sorted version of the bookstore file is called MASTER.

```
// JOB EXAMP JOBA,PROGRAMMER
// ASSGN SYS001,X'287'       MERGE OUTPUT UNIT ADDRESS
// ASSGN SYS002,X'E5A'       MERGE INPUT1 UNIT ADDRESS
// DLBL VSESPUC,'VSESP.USER.CATALOG',,VSAM
// DLBL SORTIN1,'MASTER',,SD
// EXTENT SYS002,338000,,,580,20
// DLBL SORTIN2,'SORT.SAMPADD',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// DLBL SORTOUT,'SORT.SAMPOUT',,SD
// EXTENT SYS001,338000,,,600,20
// EXEC SORT
   MERGE  FIELDS=(110,5,A,1,75,A),FORMAT=CH
   RECORD  TYPE=F,LENGTH=173
/*
/&
```

In Chapter 6, “Calling DFSORT/VSE from a Program” on page 45, you learn how to merge files when calling DFSORT/VSE from a program.

So Far

So far in this chapter you covered how to write both the SORT and MERGE control statements and how to process those control statements using the JCL EXEC statement. Now you continue with the tutorial on copy statement.

Copying Files

With DFSORT/VSE you can copy files directly without performing a sort or merge.

You write a copy statement by specifying FIELDS=COPY on the SORT or MERGE statement.

You can use copy statement with all of the other DFSORT/VSE control statements except INPFIL and SUM. DFSORT/VSE can select and reformat the specific files you want to copy by using the control statements covered in later chapters.

Specifying COPY on the SORT or MERGE Statement

The SORT and MERGE statements change very little when you specify COPY. Just replace the information you usually put in parentheses with the word COPY:

```
 SORT FIELDS=COPY
```

```
 MERGE FIELDS=COPY
```

Both of these statements have identical results.

Copying Files with the JCL EXEC Statement

The JCL for a copy application is the same as for a sort, except that you do not use the DLBL SORTWKn statement.

This sample JCL will copy a file using the SORT FIELDS=COPY statement:

```
// JOB EXAMP JOBA,PROGRAMMER
// ASSGN SYS001,X'287'       SORT OUTPUT UNIT ADDRESS
// DLBL VSESPUC,'VSESP.USER.CATALOG',,VSAM
// DLBL SORTIN1,'SORT.SAMPIN',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// DLBL SORTOUT,'SORT.SAMPOUT',,SD
// EXTENT SYS001,338000,,,600,20
// EXEC SORT
    SORT  FIELDS=COPY
    RECORD  TYPE=F,LENGTH=173
/*
/&
```

You can use MERGE FIELDS=COPY instead of SORT FIELDS=COPY to produce the same results.

Summary

In this chapter of *Getting Started with DFSORT/VSE* you have covered the following concepts:

- Writing the SORT, COPY, or MERGE control statement
- Using the JCL EXEC to process your sort, copy, or merge

As you continue with the tutorials, you will cover two methods of tailoring your input file: using the INCLUDE statement and using the OMIT statement. Chapter 3, “Tailoring the Input File with INCLUDE or OMIT” on page 21 covers padding and truncation rules, allowable comparison operators for INCLUDE and OMIT, and formats for writing constants and strings.

Chapter 3. Tailoring the Input File with INCLUDE or OMIT

Often, you need only a subset of the records in a file for an application. This chapter explains how to tailor the input file by selecting specific records.

By tailoring the file, you can increase the speed of the sort, merge, or copy. The fewer the records, the less time it takes to process them.

You tailor an input file by:

- Using an INCLUDE control statement to collect wanted records
- Using an OMIT control statement to exclude unwanted records

Your choice of INCLUDE or OMIT depends on which is easier and more efficient to write for a given application. *You cannot use both statements together.*

You select the records you want included or omitted by comparing the contents of a field with either:

Another field	For example, you can select records for which the author's last name is the same as the instructor's last name.
A constant	The constant can be a character string, a decimal number, or a hexadecimal string. For example, you can select records that have the character string "HIST" in the department field.

You can also combine two conditions with logical ANDs and ORs. For example, you can select records that have either "HIST" or "PSYCH" in the department field.

INCLUDE and OMIT also offer a powerful substring search capability which allows you to select records based on the result of bit logic tests using bit or hexadecimal masks or bit constants. Examples of these feature are not shown in this book but details can be found in *Application Programming Guide*.

Writing the INCLUDE Statement

Suppose it is the end of the year and you want to sort, by title, only the books that you need to reorder for the coming year. If the number of copies sold this year for a particular book is greater than the number in stock, you can assume you need to order more copies.

To write an INCLUDE statement that selects only the books you need to order:

Figure 12. Steps to Create the INCLUDE Statement for Books You Need to Order

Step	Action
1	Leave at least one blank and type **INCLUDE**
2	Leave at least one blank and type **COND=**
3	Type, in parentheses, and separated by commas: 1. The location, length, and data format of the number sold field 2. The comparison operator **GT** (comparison operators are shown in Figure 13) for greater than 3. The location, length, and data format of the number-in-stock field. You can use **FORMAT=** when fields have the same data format.

You can select from the following comparison operators:

Comparison Operator	Meaning
EQ	Equal to
NE	Not equal to
GT	Greater than
GE	Greater than or equal to
LT	Less than
LE	Less than or equal to

Figure 13. Comparison Operators

You can place the SORT statement either before or after the INCLUDE statement. Control statements do not have to be in any specific order. However, it is good documentation practice to code them in the order in which they are processed. For a flowchart showing the order in which all the control statements are processed, see Appendix C, "Processing Order of Control Statements" on page 103.

```
INCLUDE COND=(166,4,GT,162,4),FORMAT=BI
              ─────     ─────
                │         └──────────► Number in stock
                │
                └────────────────────► Number sold


SORT FIELDS=(1,75,CH,A)
RECORD TYPE=F,LENGTH=173
```

This sorts the tailored file by title in ascending order by using the SORT statement. Figure 14 on page 23 shows the sorted file.

Figure 14. Books for Which Number Sold Is Greater Than Number in Stock

Book Title	Number In Stock	Number Sold
1 75	162 165	166 169
ADVANCED TOPICS IN PSYCHOANALYSIS	1	12
COMPUTER LANGUAGES	5	29
COMPUTERS: AN INTRODUCTION	20	26
CRISES OF THE MIDDLE AGES	**14**	**17**
EDITING SOFTWARE MANUALS	13	32
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	**2**	**32**
INTRODUCTION TO BIOLOGY	6	11
MODERN ANTHOLOGY OF WOMEN POETS	**1**	**26**
NUMBERING SYSTEMS	6	27
STRATEGIC MARKETING	3	35
SUPPLYING THE DEMAND	**0**	**32**
SYSTEM PROGRAMMING	4	23
THE COMPLETE PROOFREADER	7	19

Suppose you want to tailor the input file even further, to sort only the books you need to order from COR publishers. In this case, two conditions must be true:

- The number sold is greater than the number in stock.
- The book is published by COR.

To add the second condition, expand the INCLUDE statement by adding a logical AND, and compare the contents of the publisher field to the character string “COR” (see “Writing Constants” on page 26 for details how to specify constants). Because the publisher field is 4 bytes long, “COR” will be padded on the right with one blank.

```
INCLUDE COND=(166,4,BI,GT,162,4,BI,AND,106,4,CH,EQ,C'COR')
SORT FIELDS=(1,75,CH,A)
RECORD TYPE=F,LENGTH=173
```

Figure 15 shows the result.

Figure 15. COR Books for Which Number Sold Is Greater Than Number in Stock

Book Title	Publisher	Number In Stock	Number Sold
1 75	106 109	162 165	166 169
CRISES OF THE MIDDLE AGES	COR	14	17
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	COR	2	32
MODERN ANTHOLOGY OF WOMEN POETS	COR	1	26
SUPPLYING THE DEMAND	COR	0	32

As another example, you might sort only the books for courses 00032 and 10347 by writing the INCLUDE and SORT statements as follows:

```
INCLUDE COND=(115,5,CH,EQ,C'00032',OR,115,5,CH,EQ,C'10347')
SORT FIELDS=(115,5,CH,A)
RECORD TYPE=F,LENGTH=173
```

Note: In the previous example, you cannot substitute C'32' for C'00032', because character constants are padded on the right with blanks. DFSORT/VSE uses the following rules for padding and truncation:

Padding adds zeros or blanks in data

Truncation deletes or omits a leading or trailing portion of a string

In comparisons, the following rules apply:

- In a field-to-field comparison, the shorter field is padded as appropriate (with blanks or zeros).
- In a field-to-constant comparison, the constant is padded or truncated to the length of the field. Decimal constants are padded or truncated on the left. Character and hexadecimal constants are padded or truncated on the right.

Writing the OMIT Statement

Suppose that you want to sort, by title, all the books used for courses but not those for general reading. In this case, you can use an OMIT statement that excludes records containing a blank in the course department field.

The format of the OMIT statement is the same as that of the INCLUDE statement. To exclude the general reading books, write:

```
 OMIT COND=(110,5,CH,EQ,C' ')
 SORT FIELDS=(1,75,CH,A)
 RECORD TYPE=F,LENGTH=173
```

Figure 16 shows the sorted file.

Figure 16. Sorted File without Books Not Required for Classes

Book Title		**Course Department**
1	75	110 114
ADVANCED TOPICS IN PSYCHOANALYSIS		PSYCH
COMPUTER LANGUAGES		COMP
COMPUTERS: AN INTRODUCTION		COMP
CRISES OF THE MIDDLE AGES		HIST
EDITING SOFTWARE MANUALS		ENGL
EIGHTEENTH CENTURY EUROPE		HIST
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		ENGL
INTRODUCTION TO BIOLOGY		BIOL
INTRODUCTION TO PSYCHOLOGY		PSYCH
MODERN ANTHOLOGY OF WOMEN POETS		ENGL
NUMBERING SYSTEMS		COMP
SHORT STORIES AND TALL TALES		ENGL
STRATEGIC MARKETING		BUSIN
SUPPLYING THE DEMAND		BUSIN
SYSTEM PROGRAMMING		COMP
THE COMPLETE PROOFREADER		ENGL
THE INDUSTRIAL REVOLUTION		HIST
VIDEO GAME DESIGN		COMP

Allowable Comparisons for INCLUDE and OMIT

Figure 17 and Figure 18 show the allowable field-to-field and field-to-constant comparisons for INCLUDE and OMIT.

Figure 17. Allowable Field-to-Field Comparisons

Field Format	BI	CH	ZD	PD
BI	√	√		
CH	√	√		
ZD			√	√
PD			√	√

Figure 18. Allowable Field-to-Constant Comparisons

Field Format	Character String	Hexadecimal String	Decimal Number
BI	√	√	
CH	√	√	
ZD			√
PD			√

For example, if you want to sort by author's name and include only those books whose author's last name begins with "M," you can compare the contents of byte 76 (the first byte of the author's last name), which is in character format, with either a character or hexadecimal string:

```
INCLUDE COND=(76,1,CH,EQ,C'M')
SORT FIELDS=(76,15,CH,A)
RECORD TYPE=F,LENGTH=173
```

or

```
INCLUDE COND=(76,1,CH,EQ,X'D4')
SORT FIELDS=(76,15,CH,A)
RECORD TYPE=F,LENGTH=173
```

Also, if you want to sort by number in stock only the books for which the number in stock is less than 10, you can compare the contents of the number-in-stock field, which is in binary format, to a hexadecimal string:

```
INCLUDE COND=(162,4,BI,LT,X'0000000A')
SORT FIELDS=(162,4,BI,A)
RECORD TYPE=F,LENGTH=173
```

Again, remember the padding and truncation rules. If you specify X'0A', the string is padded on the right instead of the left.

Writing Constants

The formats for writing character strings, hexadecimal strings, and decimal numbers are shown below.

Character Strings

The format for writing a character string is:

```
C'x...x'
```

where *x* is an EBCDIC character. For example, C'FERN'.

If you want to include a single apostrophe in the string, you must specify it as two single apostrophes. For example, *O'NEILL* must be specified as C'O''NEILL'.

Hexadecimal Strings

The format for writing a hexadecimal string is:

```
X'yy...yy'
```

where *yy* is a pair of hexadecimal digits. For example, X'7FB0'.

Decimal Numbers

The format for writing a decimal number is:

```
n...n    or    ±n...n
```

where *n...n* is a decimal digit. Examples are 24, +24, and -24.

Summary

This chapter covered two ways to tailor the input file to make processing more efficient. You wrote INCLUDE and OMIT statements and read about allowable comparison operators and the formats for writing constants in the control statement.

Chapter 4. Summing Records

Suppose that the English department wants to know the total price of books for all its courses. You can tailor the file to include only records for the English department by using the INCLUDE statement, and add the book prices together by using the SORT and SUM statements.

On the SUM control statement, you specify one or more numeric fields that are to be summed whenever records have equal control fields (control fields are specified on the SORT statement). The numeric fields can be in binary, packed decimal, or zoned decimal format.

To sum the prices for all the records for the English department, specify the price field on the SUM statement and the department field on the SORT statement. By the time SUM and SORT are processed, INCLUDE has already tailored the file to contain only the records for the English department, making the department field equal for all the records, and allowing the prices to be summed. (For a flowchart showing the order in which the INCLUDE, SUM, and SORT statements are processed, see Appendix C, "Processing Order of Control Statements" on page 103.)

When you sum records, keep in mind that two types of fields are involved:

Control fields	specified on the SORT statement
Summary fields	specified on the SUM statement

The contents of the summary fields are summed only when the contents of the control fields are of the same data type. See Figure 2 on page 2.

Writing the SUM Statement

To write a SUM statement that sums the prices for the English department:

Figure 19. Steps to Create the SUM Statement for Prices

Step	Action
1	Leave at least one blank and type **SUM**
2	Leave at least one blank and type **FIELDS=**
3	Write, in parentheses and separated by commas, the location, length, and data format of the price field.

The INCLUDE, SORT, and SUM statements are shown below:

```
INCLUDE COND=(110,5,CH,EQ,C'ENGL')
SORT FIELDS=(110,5,CH,A)
RECORD TYPE=F,LENGTH=173
SUM FIELDS=(170,4,BI)
            ────────
                └──────► Price
```

When the prices are summed, the final sum appears in the price field of one record, and the other records are deleted. Therefore, the result (shown in Figure 20 on page 28) is only one record, containing the sum. You can control which record appears if you specify that records keep their original order. For the examples, the default is for records with equal control fields to appear in their original order. When summing records keeping the original order, DFSORT/VSE chooses the first record to contain the sum.

Figure 20. Sum of Prices for English Department

Book Title		**Course Department**	**Price**
1	75	110 114	170 173
INKLINGS: AN ANTHOLOGY OF YOUNG POETS[1]		ENGL[2]	4640[3]

Note:

[1] Some of the fields in your summation record might not be meaningful, such as the book title field in Figure 20. You could use the OUTREC statement to omit this field. In the next chapter, you will learn two ways to leave out fields that are not meaningful.

[2] Specified as a control field.

[3] Specified as a summary field.

Suppose now that the English department wants to know the total price of books for *each* of its courses. In this case, you still select only the English department's records using INCLUDE, and specify the price field on the SUM statement, but you specify the *course number* on the SORT statement.

```
INCLUDE COND=(110,5,CH,EQ,C'ENGL')
SORT FIELDS=(115,5,CH,A)
RECORD TYPE=F,LENGTH=173
SUM FIELDS=(170,4,BI)
            ────┬────
                └──────────→ Price
```

Figure 21 shows the result, one record per course.

Figure 21. Sum of Prices for English Department

Book Title		**Course Number**	**Price**
1	75	115 119	170 173
SHORT STORIES AND TALL TALES		10054	1520
EDITING SOFTWARE MANUALS		10347	2075
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		10856	1045

For an example using two summary fields, assume that for inventory purposes you want to sum separately the number of books in stock and the number sold for each of the publishers.

For this application, specify the publisher as the control field on the SORT statement and the number in stock and number sold as summary fields on the SUM statement.

```
SORT FIELDS=(106,4,CH,A)
RECORD TYPE=F,LENGTH=173
SUM FIELDS=(162,4,166,4),FORMAT=BI
            ──┬──  ──┬──
              │      └──► Number sold
              │
              └─────────► Number in stock
```

Figure 22 shows the result, one record per publisher.

Figure 22. Sum of Number in Stock and Number Sold for Each Publisher

Book Title		Publisher	Number In Stock	Number Sold
1	75	106 109	162 165	166 169
LIVING WELL ON A SMALL BUDGET		COR	103	161
COMPUTER LANGUAGES		FERN	19	87
VIDEO GAME DESIGN		VALD	42	97
COMPUTERS: AN INTRODUCTION		WETH	62	79

Suppressing Records with Duplicate Control Fields

Apart from summing values, you can also use SUM to delete records with duplicate control fields.

For example, you might want to list the publishers in ascending order, with each publisher appearing only once. If you use only the SORT statement, COR appears seven times (because seven books in the file are published by COR), FERN appears four times, VALD five times, and WETH four times.

By specifying FIELDS=NONE on the SUM statement as shown below, DFSORT/VSE writes only one record per publisher:

```
SORT FIELDS=(106,4,CH,A)
RECORD TYPE=F,LENGTH=173
SUM FIELDS=NONE
```

Figure 23 shows the result.

Figure 23. List of Publishers, Deleting Duplicates

Book Title		Publisher
1	75	106 109
LIVING WELL ON A SMALL BUDGET		COR
COMPUTER LANGUAGES		FERN
VIDEO GAME DESIGN		VALD
COMPUTERS: AN INTRODUCTION		WETH

Handling Overflow

When a sum becomes larger than the space available for it, *overflow* occurs. For example, if a 2-byte binary field (unsigned) contains X'FFFF' and you add X'0001' to it, overflow occurs, because the sum requires more than two bytes.

```
 FFFF
 0001
10000
```

If overflow occurs, the two records involved are not added together. That is, the contents of the records are left untouched, neither record is deleted, and the records are still available to be summed. Overflow does not prevent further summary.

In some cases, you can correct overflow by padding the summary fields with zeros, using the INREC control statement. “Preventing Overflow When Summing Values” on page 43 shows you how to do this.

Summary

This chapter covered summing records in your file. It explained how to use the SUM statement to sum records with equal control fields, and how to suppress any records with duplicate control fields. Now, you continue with tutorials about using OUTREC and INREC to reformat your files.

Chapter 5. Reformatting Records

You can reformat records in your files by using the OUTREC and INREC control statements. With OUTREC and INREC, you can:

- Delete fields
- Reorder fields
- Insert separators (blanks, zeros, or constants)

The difference between the two DFSORT/VSE control statements is that OUTREC reformats records *after* they are sorted, copied, or merged, whereas INREC reformats records *before* they are sorted. This has an effect on other control statements. See “Using Other Statements with INREC” on page 42 for information about how INREC affects other control statements.

INREC and OUTREC perform the same functions for the sort application. When deciding which to use, remember their processing order. In general:

- If you are deleting fields, try to use INREC because shorter records take less time to sort (INREC reformats the records before they are sorted).
- If you are going to insert separators, use OUTREC because OUTREC inserts the separators into the records after they are sorted. OUTREC can insert blanks, zeros, and constants whereas INREC can only insert blanks and zeros.
- If you are reordering fields, you can use either control statement because reordering fields does not affect the record length.

Reformatting Records After Sorting

In the last chapter, you used the SUM statement to sum the price of the books in stock and the books sold for each publisher. Now, using the OUTREC statement, you can delete all the fields that are not needed for the application; in other words, fields whose contents are not meaningful in a summation record. Only the publisher, number-in-stock, and number-sold fields are written, reducing the output record length to 12 bytes.

To write the OUTREC statement:

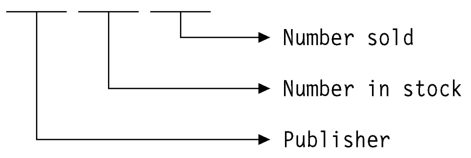
Figure 24. Steps to Create the OUTREC Statement for Reformatting Records

Step	Action
1	Leave at least one blank, and type **OUTREC**
2	Leave at least one blank, and type **FIELDS=**
3	Type, in parentheses, and separated by commas: 1. The location and length of the publisher field 2. The location and length of the number-in-stock field 3. The location and length of the number-sold field.

Because the number-in-stock and number-sold fields are next to each other, you can also specify them together as one field. They do not need to have the same data format.

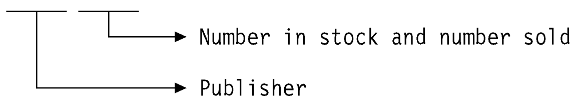
Note that on the OUTREC statement you do *not* specify the data format.

```
SORT FIELDS=(106,4,CH,A)
RECORD TYPE=F,LENGTH=173
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,162,4,166,4)
```



Or:

```
SORT FIELDS=(106,4,CH,A)
RECORD TYPE=F,LENGTH=173
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,162,8)
```



Note: If you use INREC or OUTREC to change the record length, you must be aware of the change in record size and layout of the resulting reformatted output records. The final length is either:

- The INREC length if you are using just INREC
- The OUTREC length if you are using just OUTREC or both INREC and OUTREC

Figure 25 shows the output.

Figure 25. Writing Only Publisher, Number-in-Stock, and Number-Sold Fields

Publisher	**Number In Stock**	**Number Sold**
1 4	5 8	9 12
COR	103	161
FERN	19	87
VALD	42	97
WETH	62	79

Reordering Fields to Reserve Space

The fields always appear in the order in which you specify them. Therefore, if you want the number sold to appear before the number in stock, as shown in Figure 26 on page 33, you reverse their order on the OUTREC statement.

```
SORT FIELDS=(106,4,CH,A)
RECORD TYPE=F,LENGTH=173
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,166,4,162,4)
```

Figure 26. Reordering the Fields

Publisher	Number Sold	Number In Stock
1 4	5 8	9 12
COR	161	103
FERN	87	19
VALD	97	42
WETH	79	62

Suppose you want to select and reformat specific records from the file shown in Figure 27:

Figure 27. Bookstore File as a Source for a Copy Application

Book Title		Course Department	Price
1	75	110 114	170 173
LIVING WELL ON A SMALL BUDGET			9900
PICK'S POCKET DICTIONARY			295
INTRODUCTION TO BIOLOGY		BIOL	2350
SUPPLYING THE DEMAND		BUSIN	1925
STRATEGIC MARKETING		BUSIN	2350
COMPUTER LANGUAGES		COMP	2600
COMPUTERS: AN INTRODUCTION		COMP	1899
NUMBERING SYSTEMS		COMP	360
SYSTEM PROGRAMMING		COMP	3195
VIDEO GAME DESIGN		COMP	2199
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		ENGL	595
EDITING SOFTWARE MANUALS		ENGL	1450
MODERN ANTHOLOGY OF WOMEN POETS		ENGL	450
THE COMPLETE PROOFREADER		ENGL	625
SHORT STORIES AND TALL TALES		ENGL	1520
THE INDUSTRIAL REVOLUTION		HIST	795
EIGHTEENTH CENTURY EUROPE		HIST	1790
CRISES OF THE MIDDLE AGES		HIST	1200
INTRODUCTION TO PSYCHOLOGY		PSYCH	2200
ADVANCED TOPICS IN PSYCHOANALYSIS		PSYCH	2600

From the complete file, you want a copy of just the reading list (without the prices) for the computer department.

Use the INCLUDE statement to select only departments equal to "COMP", add the OUTREC statement to include only the title and department fields, and use the SORT statement to specify the copy function. The statements look like this:

```
INCLUDE COND=(110,5,CH,EQ,C'COMP')
SORT FIELDS=COPY
RECORD TYPE=F,LENGTH=173
OUTREC FIELDS=(1,114)
```

Figure 28 on page 34 shows the copy of the file.

Figure 28. List of Computer Texts Copied from Bookstore File

Book Title		Course Department
1	75	110 114
COMPUTER LANGUAGES		COMP
COMPUTERS: AN INTRODUCTION		COMP
NUMBERING SYSTEMS		COMP
SYSTEM PROGRAMMING		COMP
VIDEO GAME DESIGN		COMP

Inserting Binary Zeros

Building on the last example, assume you want to reformat the records to include a new 4-byte binary field after the number in stock (beginning at byte 13). In this case, you can insert binary zeros as place holders for the new field (to be filled in with data at a later date). You can use **Z** or **1Z** to specify a single binary zero.

To insert the zeros, write 4Z after the last field:

```
SORT FIELDS=(106,4,CH,A)
RECORD TYPE=F,LENGTH=173
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,166,4,162,4,4Z)
```

Figure 29 shows the result.

Figure 29. Inserting Binary Zeros

Publisher	Number Sold	Number In Stock	X'0...0'
1 4	5 8	9 12	13 16
COR	161	103	0...0
FERN	87	19	0...0
VALD	97	42	0...0
WETH	79	62	0...0

Inserting Blanks

You can make DFSORT/VSE output more legible by using the OUTREC statement to separate the fields with blanks and to create margins. You can insert blanks before, between, or after fields. You can use **X** or **1X** to specify a single blank.

For example, assume you want to print just the publisher and title fields, with the publisher field appearing first. Because most of the publishers' names fill up the entire 4-byte publisher field, the publishers' names will run into the titles if you do not separate the two fields with blanks. Also, without a margin, the publishers' names will begin at the edge of the paper.

The printout can be made more legible by separating the fields with 10 blanks and creating a margin of 20 blanks.

To insert the blanks, specify 10X between the two fields, and 20X before the first field. The SORT statement sorts the records by title in ascending order (remember that SORT or MERGE is always required).

```
SORT FIELDS=(1,75,CH,A)
RECORD TYPE=F,LENGTH=173
OUTREC FIELDS=(20X,106,4,10X,1,75)
```

Figure 30 shows the result.

Figure 30. Output After Inserting Blanks

```
              Publisher        Book Title
1         20  21  24    25  34 35                                             109
(20 Blanks)             (10 Blanks)
              FERN             ADVANCED TOPICS IN PSYCHOANALYSIS
              FERN             COMPUTER LANGUAGES
              WETH             COMPUTERS: AN INTRODUCTION
              COR              CRISES OF THE MIDDLE AGES
              VALD             EDITING SOFTWARE MANUALS
              WETH             EIGHTEENTH CENTURY EUROPE
              COR              INKLINGS: AN ANTHOLOGY OF YOUNG POETS
              VALD             INTRODUCTION TO BIOLOGY
              COR              INTRODUCTION TO PSYCHOLOGY
              COR              LIVING WELL ON A SMALL BUDGET
              COR              MODERN ANTHOLOGY OF WOMEN POETS
              FERN             NUMBERING SYSTEMS
              COR              PICK'S POCKET DICTIONARY
              VALD             SHORT STORIES AND TALL TALES
              VALD             STRATEGIC MARKETING
              COR              SUPPLYING THE DEMAND
              WETH             SYSTEM PROGRAMMING
              FERN             THE COMPLETE PROOFREADER
              WETH             THE INDUSTRIAL REVOLUTION
              VALD             VIDEO GAME DESIGN
```

Inserting Constants

In addition to making the printout more legible, OUTREC can also be used to set up a very basic report format by inserting constants. ICETOOL's DISPLAY operator can be used to create complex reports as you will see in a later example. The formats for writing constants are shown below.

Character Strings

The format for writing a character string is:

```
C'x...x'
```

where *x* is an EBCDIC character. For example, C'FERN'.

The format for writing character string repetition is:

```
nC'x...x'
```

where n can be from 1 to 4095; n repetitions of the character string constant (C'x...x') are inserted into the reformatted input records. If n is omitted, 1 is used instead.

If you want to include a single apostrophe in the string, you must specify it as two single apostrophes. For example, *O'NEILL* must be specified as C'O''NEILL'.

Hexadecimal Strings

The format for writing a hexadecimal string is:

```
X'yy...yy'
```

where *yy* is a pair of hexadecimal digits. For example, X'7FB0'.

The format for writing a hexadecimal string is:

```
nX'yy...yy'
```

where n can be from 1 to 4095. n repetitions of the hexadecimal string constant (X'yy..yy') are inserted in inserted in the reformatted input records. If n is omitted, 1 is used.

Setting Up the Report Format

To produce a very basic report of the publisher's names and author's names from the bookstore file, you can OUTREC to put in "Publisher is" and "Author is" as character separators.

To write the OUTREC statement:

Figure 31. Steps to Write the OUTREC Statement

Step	Action
1	Leave at least one blank and type **OUTREC**
2	Leave at least one blank and type **FIELDS=**
3	Type, in parenthesis: 1. The repetition (**10**) and the letter **X**, for ten blanks, followed by a comma. 2. The letter **C**. 3. The term **Publisher is** in single quotes and followed by a comma. Make sure that there is one space after the **is** and before the single quote. Otherwise, the first name will look like a continuation of the word **is** (Alternatively, you can use **X** for the space.) 4. The location (**106**) and length (**4**) of the publisher field, each followed by a comma. 5. The repetition (**3**) and the letter **X**, for three blanks, followed by a comma. 6. The term **Author is** in single quotes (with an extra space), followed by a comma. 7. The location (**91**) and length (**15**) of the author's-first-name field, each followed by a comma. 8. The letter **X**, for one blank,followed by a comma. 9. The location of the author's-last-name field (**76**), followed by a comma, and length of the field (**15**).

The statements look like this:

```
SORT FIELDS=COPY
RECORD TYPE=F,LENGTH=173
OUTREC FIELDS=(10X,C'PUBLISHER IS ',106,4,3X,
       C'Author is ',91,15,X,76,15)
```

The result is shown in Figure 32.

Figure 32. Output of a Report

		Publisher		**Author's First Name**	**Author's Last Name**
1 10	11 22	24 27	31 39	41 55	57 71
(10 blanks)	Publisher is	FERN	Author is	ROBERT	MURRAY
	Publisher is	COR	Author is	FRANK	DEWAN
	Publisher is	COR	Author is	TOM	MILLER
	Publisher is	VALD	Author is	LORI	RASMUSSEN
	Publisher is	COR	Author is	KAREN	WILDE
	Publisher is	WETH	Author is	JOKHI	DINSHAW
	Publisher is	COR	Author is	CAROL	GUSTLIN
	Publisher is	VALD	Author is	VICTOR	OJALVO
	Publisher is	FERN	Author is	WILLIAM	BAYLESS
	Publisher is	VALD	Author is	MARK	YAEGER
	Publisher is	WETH	Author is	DON	GROSS
	Publisher is	COR	Author is	PETER	COWARD
	Publisher is	COR	Author is	LINDA	DUZET
	Publisher is	FERN	Author is	ANN	GREEN
	Publisher is	WETH	Author is	RAUL	CAUDILLO
	Publisher is	VALD	Author is	LILIANA	AVRIL
	Publisher is	VALD	Author is	CHIEN	WU
	Publisher is	FERN	Author is	DIANNE	OSTOICH
	Publisher is	WETH	Author is	ALICE	MUNGER
	Publisher is	COR	Author is	GREG	BENDER

Using Edit and Lookup Features

OUTREC offers other features that can be used to make a printout of your output file more readable: edit masks, lookup and change, date conversion and hexadecimal conversion. This section shows a a simple example using an edit mask and a lookup and change table. For complete information on all of OUTREC's features, see *Application Programming Guide*.

Edit masks tell DFSORT/VSE to change the appearance of your numeric fields (binary, fixed point, packed decimal or zoned decimal) according to a specific pattern. For example, you can insert a comma as a thousands separator and a period as a decimal point. Twenty-six pre-defined edit masks are available. You can also create your own edit masks.

Lookup and change tells DFSORT/VSE to look at the value of a field in your input record and match it to the values you have set up in a table. When a match is found, the associated string in the table is substituted for the original value in the output record. This makes it easy to substitute meaningful words or phrases for cryptic values (for example, "FERN" can be changed to "FERNALL BROS.").

Here's the complete JCL and control statements for a job that uses an edit mask (M18) for the Price field and a lookup and change table for the Publisher field.

```
// JOB EXAMP JOBA,PROGRAMMER
// DLBL VSESPUC,'VSESP.USER.CATALOG',,VSAM
// DLBL SORTIN1,'SORT.SAMPIN',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// DLBL SORTIN2,'SORT.SAMPADD',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// EXEC SORT
 OPTION SORTOUT=LST
 SORT FIELDS=(1,50,CH,A),FILES=2
 RECORD TYPE=F,LENGTH=173
 OUTREC FIELDS=(5:1,50,
       60:106,4,
         CHANGE=(15,
           C'FERN',C'FERNALL BROS.',
           C'COR',C'CORNISH LTD.',
           C'VALD',C'VALDERN AND CO.',
           C'WETH',C'WETHMAN INC.'),
         NOMATCH=(C'UNKNOWN'),
       80:170,4,BI,M18)
/*
/&
```

To write the OUTREC statement:

Figure 33. Steps to Write the OUTREC Statement for the Report

Step	Action
1	Leave at least one blank, and type **OUTREC**
2	Leave at least one blank, and type **FIELDS=**
3	Type, in parentheses, and separated by commas, the following: 1. The column you want the Book Title to start in, which is 5, followed by a colon. 2. The location (1) and length (50) of the Book Title field. Although the full Book Title field is 75 characters, we are only using the first 50 characters of the field here. 3. On the next line, the column you want the expanded Publisher field to start in, which is 60, followed by a colon. 4. The location (106) and length (4) of the Publisher field. 5. On the next line, the subparameter CHANGE= which describes your lookup and change table. OUTREC's lookup and change feature can be used in many ways in output records and reports to substitute meaningful words and phrases for cryptic character, hexadecimal and bit values. 6. Type, in parentheses, and separated by commas, the following: a. The length for the expanded Publisher field, which is 15. b. On the next line, the lookup table to convert the Publisher field to the expanded Publisher field, which consists of a character constant for each expected input field value (for example, C'FERN') followed by a character constant for the associated output field value (for example, C'FERNALL BROS.'). 7. On the next line, the subparameter NOMATCH= which indicates the action to be taken for an unexpected input value. Use the NOMATCH feature to identify invalid or unexpected values in your report. If you do not specify NOMATCH, DFSORT/VSE issues a message and terminates for an unexpected value. 8. Type, in parentheses, the character constant for an unexpected input field value, which is C'UNKNOWN'. 9. On the next line, the column you want the edited Price field to start in, which is 80, followed by a colon. 10. The location (170), length (4), format (BI) and edit mask (M18) for the Price field.

The result is shown in Figure 34 on page 41.

Figure 34. Edited Report

Book Title	Publisher	Price ($)
1 4 5	60	80
A SMALLER WORLD: MICROBES	FERNALL BROS.	19.95
ADVANCED TOPICS IN PSYCHOANALYSIS	FERNALL BROS.	26.00
ANOTHER ITALIAN DICTIONARY	CORNISH LTD.	9.25
ANTICIPATING THE MARKET	WETHMAN INC.	20.00
CELLS AND HOW THEY WORK	VALDERN AND CO.	24.95
CIVILIZATION SINCE ROME FELL	WETHMAN INC.	13.50
COMPLETE SPANISH DICTIONARY	VALDERN AND CO.	6.50
COMPUTER LANGUAGES	FERNALL BROS.	26.00
COMPUTERS: AN INTRODUCTION	WETHMAN INC.	18.99
CRISES OF THE MIDDLE AGES	CORNISH LTD.	12.00
DESIGNING APPLICATIONS	CORNISH LTD.	14.35
DNA: BLUEPRINT FOR YOU	FERNALL BROS.	21.95
EDITING SOFTWARE MANUALS	VALDERN AND CO.	14.50
EIGHTEENTH CENTURY EUROPE	WETHMAN INC.	17.90
FRENCH TO ENGLISH DICTIONARY	FERNALL BROS.	11.00
FREUD'S THEORIES	VALDERN AND CO.	12.50
GUIDE TO COLLEGE LIFE	WETHMAN INC.	20.00
GUNTHER'S GERMAN DICTIONARY	WETHMAN INC.	10.88
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	CORNISH LTD.	5.95
INTRODUCTION TO BIOLOGY	VALDERN AND CO.	23.50
INTRODUCTION TO PSYCHOLOGY	CORNISH LTD.	22.00
KNOW YOUR CONSUMER	CORNISH LTD.	45.00
LIVING WELL ON A SMALL BUDGET	CORNISH LTD.	99.00
MAP OF THE HUMAN BRAIN	CORNISH LTD.	8.95
MODERN ANTHOLOGY OF WOMEN POETS	CORNISH LTD.	4.50
NOVEL IDEAS	VALDERN AND CO.	24.50
NUMBERING SYSTEMS	FERNALL BROS.	3.60
PICK'S POCKET DICTIONARY	CORNISH LTD.	2.95
POLITICS AND HISTORY	FERNALL BROS.	9.95
QUEUE THEORY	FERNALL BROS.	15.00
REBIRTH FROM ITALY	WETHMAN INC.	25.60
SHORT STORIES AND TALL TALES	VALDERN AND CO.	15.20
STRATEGIC MARKETING	VALDERN AND CO.	23.50
SUPPLYING THE DEMAND	CORNISH LTD.	19.25
SYSTEM PROGRAMMING	WETHMAN INC.	31.95
THE ANIMAL KINGDOM	CORNISH LTD.	30.00
THE ART OF TAKEOVERS	FERNALL BROS.	6.15
THE COMPLETE PROOFREADER	FERNALL BROS.	6.25
THE INDUSTRIAL REVOLUTION	WETHMAN INC.	7.95
THE TOY STORE TEST	CORNISH LTD.	26.00
VIDEO GAME DESIGN	VALDERN AND CO.	21.99
ZEN BUSINESS	VALDERN AND CO.	12.00

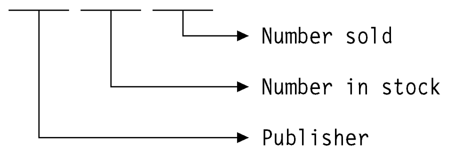
So Far

So far this chapter has covered how the OUTREC statement can define only certain fields to go in the records of your output file. This chapter has also covered inserting binary zeroes as place holders, and using blanks, constants, edit masks and lookup and change tables to make a printout of the output file more readable.

Reformatting Records Before Sorting

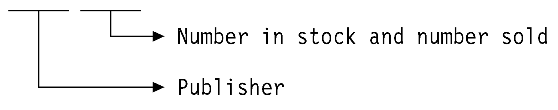
The INREC statement has some of the same capabilities as the OUTREC statement. Therefore, in the first example of “Reformatting Records After Sorting” on page 31, where you used OUTREC to write only the publisher, number-in-stock, and number-sold fields, you could use INREC instead, as shown below.

```
INREC FIELDS=(106,4,162,4,166,4)
```



Or:

```
INREC FIELDS=(106,4,162,8)
```



Using Other Statements with INREC

Because INREC reformats the records *before* they are sorted, the SORT and SUM statements must refer to the *reformatted* records as they will appear in the output file.

Thus, after INREC, the input records for the control statement in the previous section are 12 bytes long (see Figure 25 on page 32 for an example).

You write the SORT and SUM statements to process the byte positions in the output file:

```
INREC FIELDS=(106,4,162,8)
SORT FIELDS=(1,4,CH,A)
RECORD TYPE=F,LENGTH=173
SUM FIELDS=(5,4,BI,9,4,BI)
```

Figure 35 shows the result.

Figure 35. Using INREC to Write Only Publisher, Number in Stock, and Number Sold

Publisher	**Number In Stock**	**Number Sold**
1 4	5 8	9 12
COR	103	161
FERN	19	87
VALD	42	97
WETH	62	79

As the flowchart in Appendix C, “Processing Order of Control Statements” on page 103 shows, DFSORT/VSE processes the INREC statement *before* SORT, SUM, and OUTREC, but *after* INCLUDE or OMIT. Therefore, when used with the INREC statement, SORT, SUM, and OUTREC must refer to the *reformatted* records, and INCLUDE or OMIT must refer to the *original* records.

Preventing Overflow When Summing Values

In some cases, you can prevent overflow by using INREC to pad summary fields with zeros. However, this method cannot be used for negative fixed-point binary data, because padding with zeros rather than with ones would change the sign.

If the summary fields in Figure 35 on page 42 were overflowing, you could pad each of them on the *left* with 4 bytes (binary fields must be 2, 4, or 8 bytes long), as shown in Figure 36.

Figure 36. Padding Summary Fields

Publisher	**X'0...0'**	**Number In Stock**	**X'0...0'**	**Number Sold**
1 4	5 8	9 12	13 16	17 20
COR		103		161
FERN		19		87
VALD		42		97
WETH		62		79

```
INREC FIELDS=(106,4,4Z,162,4,4Z,166,4)
SORT FIELDS=(1,4,CH,A)
RECORD TYPE=F,LENGTH=173
SUM FIELDS=(5,8,BI,13,8,BI)
           ────── ──────
              │      └─────► New number sold field
              │
              └────────────► New number in stock field
```

Figure 37 shows the output records, each 20 bytes long.

Figure 37. Padding Summary Fields

Publisher	**Number In Stock**	**Number Sold**
1 4	5 12	13 20
COR	103	161
FERN	19	87
VALD	42	97
WETH	62	79

Note: You cannot use the OUTREC statement to prevent overflow, because it is processed *after* summarization.

Summary

This chapter covered using INREC and OUTREC to reformat files. You can delete fields, insert blanks or zeros, and reorder fields with both of these control statements. These two control statements help improve the appearance of DFSORT/VSE output.

Chapter 6. Calling DFSORT/VSE from a Program

In addition to processing your DFSORT/VSE control statements with a JCL EXEC statement, you can call DFSORT/VSE from programs written in COBOL, PL/I, or assembler language. In this chapter you will concentrate on sorting and merging using COBOL. The examples in this chapter assume that the COBOL environment is available.

For information on restrictions when using these languages and on calling DFSORT/VSE from an assembler, see *Application Programming Guide*.

Passing Control Statements

When using COBOL/VSE, you can pass the INCLUDE, OMIT, SUM, INREC, and OUTREC control statements to DFSORT/VSE by using the SYSIPT input or a VSE Librarian member. For example, you can assign the value "SYSIPT" (if the DFSORT/VSE control statements are to be read from the SYSIPT input) or the name of a C-type VSE Librarian member to the SORT-CONTROL special register to pass the INCLUDE control statement that selects only the English department books:

```
//JOB    EXAMP JOBA,PROGRAMMER
  .
  .
  .
/*
    INCLUDE COND=(110,5,CH,EQ,C'ENGL')
/*
/&
```

When using COBOL/VSE, you need to understand the use of the SORT-CONTROL and SORT-RETURN special registers. For full information, see the COBOL Programmer's Guide describing the compiler version available at your site.

Calling DFSORT/VSE from a COBOL Program

To call DFSORT/VSE from a COBOL program, use the COBOL statements SORT and MERGE. This section shows sample programs that use the COBOL SORT and MERGE statements. For complete information, see the COBOL Programmer's Guide describing the compiler version available at your site.

Sorting Records

The sample COBOL program in Figure 38 on page 46 calls DFSORT/VSE to sort the bookstore master file (MASTER-FILE) by title in ascending order. The sorted master file is written to SORTED-MASTER-FILE.

Below is the JCL that calls the sample COBOL program:

```
// JOB EXAMP JOBA,PROGRAMMER
// DLBL VSESPUC,'VSE.USER.CATALOG',,VSAM
// DLBL MASTIN,'SORT.SAMPIN',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// DLBL MASTOUT,'SORT.MASTER',,VSAM,CAT=VSESPUC,RECORDS=100,          C
               RECSIZE=173,DISP=(NEW,KEEP)
// EXEC IGYCRCTL,SIZE=IGYCRCTL,GO,PARM='FASTSRT'
              .
              .
              .
      <COBOL program>
              .
              .
              .
/*
      <INCLUDE control statement>
/*
/&
```

In contrast to the JCL for executing DFSORT/VSE with the JCL EXEC statement (see “Sorting Files with the JCL EXEC Statement” on page 13) the above JCL has these differences:

- The program name on the EXEC statement is that of the COBOL program.
- The name of the DLBL statement for the input file need not be SORTIN.
- The name of the DLBL statement for the output file need not be SORTOUT.

Notice that the control field and order of the sort are specified in the COBOL program itself rather than with a SORT control statement. Figure 38 shows the sample COBOL program.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.
     COBOLPGM.
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT SD-FILE ASSIGN TO
     DUMMYNM.
     SELECT MASTER-FILE ASSIGN TO
     MASTIN.
     SELECT SORTED-MASTER-FILE ASSIGN TO
     MASTOUT.
 DATA DIVISION.
 FILE SECTION.
 SD  SD-FILE
     DATA RECORD IS SD-RECORD.
 01  SD-RECORD.
  05 TITLE-IN   PICTURE X(75).
  05 AUTH-LN-IN   PICTURE X(15).
  05 AUTH-FN-IN   PICTURE X(15).
  05 PUB-IN    PICTURE X(4).
  05 COUR-DEPT-IN  PICTURE X(5).
  05 COUR-NO-IN   PICTURE X(5).
  05 COUR-NAM-IN   PICTURE X(25).
  05 INST-LN-IN   PICTURE X(15).
  05 INST-INIT-IN   PICTURE X(2).
  05 NO-STOCK-IN   PICTURE 9(8) COMP.
  05 NO-SOLD-IN   PICTURE 9(8) COMP.
  05 PRICE-IN   PICTURE 9(8) COMP.
```

Figure 38 (Part 1 of 2). Sample COBOL Program with SORT Commands

```
 FD  MASTER-FILE
     DATA RECORD IS MASTER-RECORD.
 01  MASTER-RECORD.
  05 FILLER    PICTURE X(173).

 FD  SORTED-MASTER-FILE
     DATA RECORD IS SORTED-MASTER-RECORD.
 01  SORTED-MASTER-RECORD.
  05 FILLER    PICTURE X(173).
  .
  .
  .
 PROCEDURE DIVISION.
  .
  .
  .
 SORT-ROUTINE SECTION.
     MOVE "SYSIPT" TO SORT-CONTROL.
     SORT SD-FILE
     ASCENDING KEY TITLE-IN
     USING MASTER-FILE
     GIVING SORTED-MASTER-FILE.
     IF SORT-RETURN > 0
     DISPLAY "SORT FAILED".
     .
     .
     .
SORT-REPORT SECTION.
     print a report on PRINT-FILE using SORTED-MASTER-FILE.
     .
     .
     .
     STOP RUN.
```

Figure 38 (Part 2 of 2). Sample COBOL Program with SORT Commands

Merging Records

The sample COBOL program in Figure 39 on page 48 calls DFSORT/VSE to merge the presorted bookstore master file (MASTER-FILE) with another presorted file (NEW-BOOKS-FILE) to create a new master file (MERGED-FILE).

The JCL for the program is as follows:

```
// JOB EXAMP JOBA,PROGRAMMER
// DLBL VSESPUC,'VSESP.USER.CATALOG',,VSAM
// DLBL BKINS,'SORT.MASTER',,VSAM,CAT=VSESPUC,DISP=(OLD,DELETE)
// DLBL BKADD,'SORT.NEWBKS',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// DLBL MERGOUT,'SORT.SORTOUT',,VSAM,CAT=VSESPUC,RECORDS=100,         C
               RECSIZE=173,DISP=(NEW,KEEP)
// EXEC IGYCRCTL,SIZE=IGYCRCTL,GO,PARM='FASTSRT'
              .
              .
              .
      <COBOL program>
              .
              .
              .
/*
/&
```

Figure 39 shows the sample COBOL program.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.
    COBOLPGM.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT SD-FILE ASSIGN TO
    DUMMYNM.
    SELECT MASTER-FILE ASSIGN TO
    BKINS.
    SELECT NEW-BOOKS-FILE ASSIGN TO
    BKADD.
    SELECT MERGED-FILE ASSIGN TO
    MERGOUT.
DATA DIVISION.
FILE SECTION.
SD  SD-FILE
    DATA RECORD IS SD-RECORD.
01  SD-RECORD.
 05 TITLE-KEY  PICTURE X(75).
 05 FILLER     PICTURE X(98).

FD  MASTER-FILE
    DATA RECORD IS MASTER-RECORD.
01  MASTER-RECORD.
 05 FILLER     PICTURE X(173).

FD  NEW-BOOKS-FILE
    DATA RECORD IS NEW-BOOKS-RECORD.
01  NEW-BOOKS-RECORD.
 05 FILLER    PICTURE X(173).

FD  MERGED-FILE
    DATA RECORD IS MERGED-RECORD.
01  MERGED-RECORD.
 05 FILLER    PICTURE X(173).
  .
  .
  .

PROCEDURE DIVISION.
  .
  .
  .

MERGE-ROUTINE SECTION.
    MERGE SD-FILE
    ASCENDING KEY TITLE-KEY
    USING MASTER-FILE NEW-BOOKS-FILE
    GIVING MERGED-FILE.
    IF SORT-RETURN > 0
    DISPLAY "MERGE FAILED".
    STOP RUN.
```

Figure 39. Sample COBOL Program with MERGE Commands

Sorting with COBOL/VSE FASTSRT

If you compile the COBOL program in Figure 39 on page 48 for sorting records with COBOL/VSE, the input (from MASTER-FILE) and the output (to SORTED-MASTER-FILE) would qualify for the COBOL/VSE FASTSRT option. With this compile-time FASTSRT option, your sort runs faster, because DFSORT/VSE rather than COBOL does the input and output processing. For full information on FASTSRT, refer to *Application Programming Guide* and the COBOL Programmer's Guide that describes the compiler version available at your site.

Note: COBOL evaluates sort input and output independently to see if it qualifies for FASTSRT. If either the input or the output of your sort does not qualify because of the presence of an input or output procedure, you might be able to replace such a procedure and use DFSORT/VSE control statements to accomplish the same thing. For example, you can use a control statement (OUTREC) to indicate how records will be reformatted before being written to the output file.

Summary

This chapter covered methods of calling DFSORT/VSE from COBOL/VSE.

Chapter 7. Overriding Installation Defaults

IBM provides DFSORT/VSE with preset defaults. During installation, your system programmer can change these defaults. For example, one IBM-supplied default is to route the DFSORT/VSE messages to SYSLST. However, at your site, the default might be to route DFSORT/VSE messages to the system console. You can temporarily override some of the installation defaults using a DFSORT/VSE OPTION control statement. When calling DFSORT/VSE from an assembler, you can pass a DFSORT/VSE OPTION control statement in the parameter list.

In this chapter, you will learn how to override some of the many available defaults, concentrating on the OPTION control statement. For a list of all the possible defaults, and information on how to code the assembler parameter list, see *Application Programming Guide*. See also Chapter 9, “Using the ICETOOL Utility” on page 59 on using the ICETOOL DEFAULTS operator to print a list of your installation defaults.

Writing an OPTION Control Statement

Whether you execute DFSORT/VSE with the JCL EXEC statement or call it from a program, you can use the OPTION statement to override certain defaults. To do so, place the OPTION statement among the other DFSORT/VSE control statements.

A particular default that can be overridden with the OPTION statement is one that specifies whether equally collating records are to be written in their original order.

The IBM default is for DFSORT/VSE to use no GETVIS space as sort work space. If your site has kept this default and you want to temporarily override it (so that *getvis sorting* could be provided to improve performance), you can specify GVSIZE=MAX on the OPTION statement:

```
 OPTION  GVSIZE=MAX
```

Or, if your site has established GVSIZE=MAX as the default and you want to temporarily override it (so that GETVIS area size for sorting would be *limited* by 10 MB), you can specify GVSIZE=10M on the OPTION statement:

```
 OPTION  GVSIZE=10M
```

Summary

This chapter covered methods of overriding the installation defaults using a DFSORT/VSE OPTION control statement.

Chapter 8. Using DFSORT/VSE Efficiently

You will get the best performance from DFSORT/VSE if you follow these guidelines:

- Be generous with virtual storage.
- Enable dataspace sorting and getvis sorting.
- Use high-speed disks.
- Eliminate unnecessary fields with INREC.
- Eliminate unnecessary records with INCLUDE or OMIT.
- Reduce file size with STOPAFT and SKIPREC.
- Consolidate records with SUM.
- Run DFSORT/VSE with the JCL EXEC statement.
- Use FASTSRT with COBOL/VSE.
- Avoid options that might degrade performance.

Additional suggestions can be found in *Application Programming Guide* and *Installation and Tuning Guide*.

Be Generous with Virtual Storage

In general, the more partition virtual storage available to DFSORT/VSE, the better the performance. This is especially true when the input file is larger than available virtual storage.

In your installation, VSE/ESA partitions can differ in size. Use an appropriate partition to execute your sort application.

Enable Dataspace Sorting and Getvis Sorting

When the partition size is large enough (usually, more than 16 MB), you can enable getvis sorting to sort large files more efficiently.

When data space is available, you can enable dataspace sorting to sort large files more efficiently.

DFSORT/VSE is shipped with dataspace sorting and getvis sorting disabled. To enable dataspace sorting, specify DSPSIZE=n, where n is the amount of data space, in megabytes, you want DFSORT/VSE to use. To enable getvis sorting, specify GVSIZE=n, where n is the number of bytes of GETVIS area you want DFSORT/VSE to use.

Use High–Speed Disks

Using high-speed disks, such as the IBM 3390, for work files offers the best performance.

Eliminate Unnecessary Fields with INREC

If you need to reformat your records, using INREC to significantly shorten them can result in faster sort processing.

Remember that INREC reformats records *before* they are sorted, and OUTREC reformats them *after* they are sorted. Therefore, you should use INREC to shorten records and OUTREC to lengthen records. For a summary of the control statements and the corresponding record positions to refer to when using INREC, see Figure 40.

Figure 40. Control Statement and Corresponding Records with INREC

Control Statement	Original Records	Reformatted Records
SORT		√
SUM		√
OUTREC		√
INCLUDE	√	
OMIT	√	

Eliminate Unnecessary Records with INCLUDE or OMIT

Naturally, the size of the input file(s) also affects the amount of time processing will take. The fewer the records, the faster the DFSORT/VSE application. You can improve performance by using INCLUDE or OMIT whenever possible to select only the records pertaining to your application.

Reduce File Size with STOPAFT and SKIPREC

You can also use the STOPAFT and SKIPREC options to reduce the size of the input file for the sort or copy application.

- Use STOPAFT to specify the maximum number of records that should be accepted for sorting or copying.
- Use SKIPREC to skip a specified number of records at the beginning of the input file being sorted or copied.

For information on how to use these options, see *Application Programming Guide*.

Consolidate Records with SUM

You can improve performance by using the SUM statement, if appropriate for your job, to either:

- Add the contents of fields whenever two records with equal control fields are found. DFSORT/VSE places the result in one record and deletes the other, reducing the number of records to be sorted or merged.
- Delete records with duplicate control fields by specifying FIELDS=NONE on a SUM statement.

For details on these methods, see Chapter 4, “Summing Records” on page 27.

Run DFSORT/VSE with JCL

As a rule, DFSORT/VSE is more efficient when executed with the JCL EXEC statement than when called from a program.

Although calling DFSORT/VSE from a program might be convenient if the program modifies the files before or after DFSORT/VSE (for example, if DFSORT/VSE sums numbers and the program calculates their average), be aware of the possible trade-off in performance.

Use FASTSRT with COBOL/VSE

With COBOL/VSE, using the FASTSRT compile-time option enhances DFSORT/VSE performance. With FASTSRT, DFSORT/VSE rather than COBOL does the input and output processing. For more information on this option, see the COBOL Programmer's Guide describing the compiler version available at your site.

Avoid Options That Might Degrade Performance

The options listed below might adversely affect DFSORT/VSE performance. Use them only when necessary. For a full description of what these options are and how they affect performance, see *Application Programming Guide* and *Installation and Tuning Guide*.

- EQUALS option
- CKPT option
- User exit routines
- Small values for STORAGE, DSPSIZE, or GVSIZE options
- LOCALE option

Summary

This chapter covered methods of improving DFSORT/VSE performance, including specification before processing and options to use or avoid.

Part 2. Learning to Use ICETOOL

The chapter in this section explains how to use the multipurpose DFSORT/VSE utility ICETOOL.

Chapter 9. Using the ICETOOL Utility 59

Chapter 9. Using the ICETOOL Utility

ICETOOL is a multipurpose DFSORT/VSE utility that uses the capabilities of DFSORT/VSE to perform multiple operations on one or more files in a single step.

This chapter will show you how to write a simple ICETOOL job using several of the ICETOOL operators. To fully use the capabilities of ICETOOL, you should become familiar with all of its operators, operands, methods of invocation, and its messages and operator return codes, as described in *Application Programming Guide*.

ICETOOL Operators

The thirteen ICETOOL operators listed below can be used to perform a variety of functions. By using various combinations of the thirteen ICETOOL operators, you can easily create applications that perform many complex tasks.

COPY Copies one or more input files to one or more output files.

COUNT Prints a message containing the count of records in one or more input files.

DEFAULTS Prints the DFSORT/VSE installation defaults.

DEFINE Specifies input or output file characteristics for COUNT, COPY, DISPLAY, OCCUR, RANGE, SELECT, SORT, STATS, UNIQUE, and VERIFY operations or where the DFSORT/VSE messages are to be routed for a group of operations.

DISPLAY Prints the values or characters of specified numeric or character fields. Simple, tailored, or sectioned reports can be produced.

MODE Three modes are available which can be set or reset for groups of operators:

- STOP mode (the default) stops subsequent operations if an error is detected
- CONTINUE mode continues with subsequent operations if an error is detected
- SCAN mode allows ICETOOL statement checking without actually performing any operations.

OCCUR Prints each unique value for specified numeric or character fields and how many times it occurs. Simple or tailored reports can be produced. The values printed can be limited to those for which the value count meets specified criteria (for example, only duplicate values or only nonduplicate values).

RANGE Prints a message containing the count of values in a specified range for a specified numeric field in one or more input files.

SELECT Selects records from one or more input files for inclusion in an output file based on meeting criteria for the number of times specified numeric or character field values occur (for example, only duplicate values or only nonduplicate values).

SORT	Sorts one or more input files to one or more output files.
STATS	Prints messages containing the minimum, maximum, average, and total for specified numeric fields in a file.
UNIQUE	Prints a message containing the count of unique values for a specified numeric or character field.
VERIFY	Examines specified decimal fields in a file and prints a message identifying each invalid value found for each field.

Input Files

Each ICETOOL operation requires an input file. The input file used by one operator can be the same or different from the input file used by another operator. Thus, ICETOOL can process many files in a single step.

This chapter uses as input files the branch office file named SORT.BRANCH, the sample bookstore file named SORT.SAMPIN, and the additional bookstore file named SORT.SAMPADD. See "Creating Your Sample Input Files" on page 5 for additional information about these sample files. Two temporary files created by ICETOOL from SORT.BRANCH are also used as input.

Note: Some of the examples use files other than SORT.BRANCH. You can either create files from scratch to match the ones used in the text, or else perform a similar exercise on files you already have.

Figure 41 shows the length and format of the fields in the branch office file (SORT.BRANCH).

Figure 41. Field Lengths and Formats for SORT.BRANCH

Field	**Length**	**Data Format**
City	15	CH
State	2	CH
Employees	4	ZD
Revenue	6	PD
Profit	6	PD

Figure 42 on page 61 shows the records in the branch office file.

Figure 42. Branch Office File Records

City		State	Employees		Revenue		Profit	
1	15	16 17	18	21	22	27	28	33
Los Angeles		CA		32		22530		–4278
San Francisco		CA		35		42820		6832
Fort Collins		CO		22		12300		–2863
Sacramento		CA		29		42726		8276
Sunnyvale		CA		18		16152		–978
Denver		CO		33		31876		6288
Boulder		CO		32		33866		7351
Morgan Hill		CA		15		18200		3271
Vail		CO		19		23202		5027
San Jose		CA		21		27225		8264
San Diego		CA		22		32940		8275
Aspen		CO		20		25800		5200

Appendix A, “Using the DFSORT/VSE Sample Files” on page 97 shows the length, format and contents of the fields in the records of the bookstore files (SORT.SAMPIN and SORT.SAMPADD).

Creating an ICETOOL Job

An ICETOOL job consists of:

1. The JCL statements that are required for every ICETOOL job
2. The JCL statements that are required as a result of the specified operator statements
3. The operator statements indicating the operations to be performed by the ICETOOL job

Writing Required JCL Statements

The first step in creating any ICETOOL job is to write the JCL that is always required. Here is the required JCL for the job in this chapter:

```
// JOB EXAMP JOBA,PROGRAMMER
// DLBL VSESPUC,'VSESP.USER.CATALOG',,VSAM
<Additional JCL statements go here>
// EXEC ICETOOL,SIZE=100K
  <ICETOOL statements go here>
/*
/&
```

- The JOB statement signals the beginning of the job.
- The EXEC statement signals the beginning of the job step and tells the operating system to run the ICETOOL program. SIZE=100K is recommended.
- The DLBL statement defines the VSE/VSAM user catalog for sample files. If the ICETOOL files are not SAM ESDS, you can omit this statement.
- The ICETOOL statements (comment, blank, and operator statements) and DFSORT/VSE sections you write must appear in SYSIPT. The additional JCL statements you write can appear before the EXEC statement.

Notes:

1. The OCCUR, SORT, SELECT, and UNIQUE operators use the SORTWK1 file. Usually, VSE/ESA installations provide the SORTWK1 file automatically. Otherwise, you should add the SORTWK1 DLBL statement to the ICETOOL job.
2. ICETOOL prints its messages to SYSLST.

ICETOOL Comment and Blank Statements

Comment statements and blank statements can be placed anywhere among the ICETOOL operator statements.

- Comment statements start with an asterisk (*) in column 1 and are printed along with the ICETOOL operator statements.
- Blank statements contain blanks in columns 1-72 and are ignored since ICETOOL prints blank lines where appropriate.

To write a blank statement and a comment statement for our example:

Figure 43. Steps to Create a Blank Statement and a Comment Statement

Step	Action
1	After the EXEC ICETOOL statement, skip one line.
2	Type an asterisk (*) in column 1 followed by the comment.

When complete, the ICETOOL statements may look like this:

```

* Statistics from all branches
```

For this ICETOOL job, a comment statement will be placed before each operator to describe its function. Although not required, this is a good practice to follow.

DEFINE Operator Statement

You must write the DEFINE operator statements for this job to specify:

- Record type and record length for the first file name specified in the FROM operand of the ICETOOL's operator. For this job ALL, BKIN, BKADD, CADASD, CODASD, and DAPUBS file names are used as input files. Figure 44 on page 64 shows an example of creating the DEFINE operator statement for the ALL file name.
- Logical unit name for each tape file name. For this job CATAPE and COTAPE file names are used. Figure 50 on page 70 shows an example of creating the DEFINE control statement for the CATAPE tape file name.
- Output printer name to route the DFSORT/VSE messages. For example, assume that two printers with FEE and FEF addresses have been defined for your partition and you want to use the SYS012 logical unit name for the FEF printer to route DFSORT/VSE messages.

You must write the following ASSGN JCL statement and DEFINE operator statement:

```
// ASSGN SYS012,FEF
       . . .
DEFINE ROUTE(012)
```

Notes:

1. The DEFINE operator statement must be placed before the corresponding ICETOOL statements.
2. VSE/POWER JECL statements can be used for this job to separate SYSLST and SYS012 outputs as follows:

```
* $$ JOB JNM=PGMA
* $$ LST CLASS=E
      . . .
  <JCL statements>
      . . .
// ASSGN SYS012,FEF
* $$ LST CLASS=D,JNM=DFSMSG,LST=SYS012
      . . .
DEFINE ROUTE(012)
  <ICETOOL statements>
      . . .
/*
/&
* $$ EOJ
```

As a result, class E job output queue will contain PGMA output for ICETOOL messages and class D job output queue will contain DFSMSG output for DFSORT/VSE messages.

So Far

So far, you have been introduced to the basics of the ICETOOL utility. Now, using the following tutorials, you can learn about some of the operators. Each of the following sections contains a part of the same ICETOOL job, so that by the end of the chapter, you will have created a complete ICETOOL job. At the end of the chapter, there is a section that contains the complete job and its resulting messages.

Printing Statistics For Numeric Fields

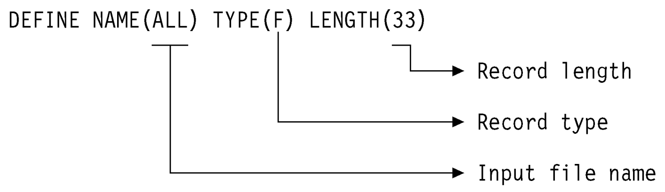
When working with files containing numeric fields, you may want statistical information about one or more of those fields. You can use the STATS operator to find the minimum, maximum, average, and total values of up to 10 specific numeric fields. The STATS operator uses input file and you must describe its characteristics using a DEFINE operator statement.

To write a DEFINE operator statement for ALL input file:

Figure 44. Steps to Create the DEFINE Operator for ALL Input File

Step	Action
1	Type **DEFINE** (you can leave one or more blanks before DEFINE if you like).
2	Leave at least one blank and type **NAME(ALL)** NAME specifies the file name (that is, the file name of the FROM operand on the STATS operator statement) for the input file from which you want to print statistics.
3	Leave at least one blank and type **TYPE(F)** TYPE specifies the type of record. In this case F (fixed length) is the record type.
4	Leave at least one blank and type **LENGTH(33)** LENGTH specifies the record length. In this case 33 bytes is the record length. Make sure that the statement is coded between columns 1 and 72.

When complete, the DEFINE operator statement looks like:



```
DEFINE NAME(ALL) TYPE(F) LENGTH(33)
```

To write a STATS operator statement that prints statistics for the employees, profit, and revenue fields of the branch office file:

Figure 45 (Page 1 of 2). Steps to Create the STATS Operator

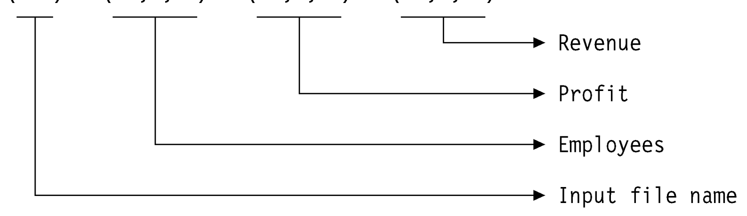
Step	Action
1	Type **STATS** after the comment statement (you can leave one or more blanks before STATS if you like).
2	Leave at least one blank and type **FROM(ALL)** FROM specifies the file name (that is, the file name on the DLBL statement) for the input file from which you want to print statistics.
3	Leave at least one blank and type **ON** ON defines a field for which you want to print statistics.
4	Type in parentheses, and separated by commas: 1. Where the employees field begins relative to the beginning of the input record (the first position is byte 1). The employees field begins at byte **18**. 2. The length of the employees field in bytes. The employees field is **4** bytes long. 3. A code for the data format. The employees field contains zoned decimal data which you specify as **ZD**.
5	Leave at least one blank and type **ON** ON defines another field for which you want to print statistics. You can print statistics for up to 10 fields with one STATS operator statement. Specify the ON fields in the same order in which you want their statistics to be printed.

Figure 45 (Page 2 of 2). Steps to Create the STATS Operator

Step	Action
6	Type in parentheses, and separated by commas the location (**28**), length (**6**), and format (**PD** for packed decimal) of the profit field.
7	Leave at least one blank and type **ON**. Type in parentheses and separated by commas, the location (**22**), length (**6**), and format (**PD**) of the revenue field. Make sure that the statement is coded between columns 1 and 72.

When complete, the STATS operator statement looks like:

```
  STATS FROM(ALL) ON(18,4,ZD) ON(28,6,PD) ON(22,6,PD)
```



You must also write a DLBL JCL statement for the SORT.BRANCH file using the file name ALL and place it before the EXEC statement:

```
   // DLBL ALL,'SORT.BRANCH',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
```

When complete, the ICETOOL statements and DLBL statement look like:

```
// DLBL ALL,'SORT.BRANCH',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
      . . .
DEFINE NAME(ALL) TYPE(F) LENGTH(33)

* Statistics from all branches
STATS FROM(ALL) ON(18,4,ZD) ON(28,6,PD) ON(22,6,PD)
```

When this STATS operation is run, the results are printed to SYSLST. If you ran the ICETOOL job you created so far, the output would look like:

```
ILU600I T0 ICETOOL UTILITY RUN STARTED
ILU630I T0 MODE IN EFFECT:  STOP

ILU632I T0 SOURCE FOR ICETOOL STATEMENTS:  SYSIPT

           * All input files characteristics
           DEFINE NAME(ALL) TYPE(F) LENGTH(33)
ILU602I T0 OPERATION RETURN CODE:  00
```

```
          * Statistics from all branches
          STATS FROM(ALL) ON(18,4,ZD) ON(28,6,PD) ON(22,6,PD)
ILU627I T0 DFSORT/VSE CALL 0001 FOR COPY FROM ALL     TO E35 EXIT COMPLETED
ILU628I T0 RECORD COUNT:  000000000000012
ILU607I T0 STATISTICS FOR (18,4,ZD)     :
ILU608I T0   MINIMUM:  +000000000000015, MAXIMUM:  +000000000000035
ILU609I T0   AVERAGE:  +000000000000024, TOTAL  :  +000000000000298
ILU607I T0 STATISTICS FOR (28,6,PD)     :
ILU608I T0   MINIMUM:  -000000000004278, MAXIMUM:  +000000000008276
ILU609I T0   AVERAGE:  +000000000004222, TOTAL  :  +000000000050665
ILU607I T0 STATISTICS FOR (22,6,PD)     :
ILU608I T0   MINIMUM:  +000000000012300, MAXIMUM:  +000000000042820
ILU609I T0   AVERAGE:  +000000000027469, TOTAL  :  +000000000329637
ILU602I T0 OPERATION RETURN CODE:  00


ILU601I T0 DFSORT/VSE ICETOOL UTILITY RUN ENDED - RETURN CODE:  00
```

Looking at the output, you will notice that:

- Message ILU628I gives the count of records processed.
- Messages ILU607I, ILU608I, and ILU609I give the numerical statistics for each ON field specified in the order in which they were specified.
- A return code for each operator is given in message ILU602I and the highest operator return code is given in message ILU601I.

Continuing an Operator Statement

If you cannot fit your STATS operator (or any other ICETOOL operator statement) between columns 1 and 72 of a single line, you can continue it across multiple lines. If you end a line with a dash (-) after the operator or any operand, the next line is treated as a continuation. Any characters specified after the dash are ignored.

Note that the operator and each operand must be completely specified on one line (between columns 1 and 72).

For example:

```
STATS        - this is the operator
     FROM(ALL) - ALL is the file name for SORT.BRANCH
        ON(18,4,ZD)-
        ON(28,6,PD)-
        ON(22,6,PD)
```

Statistics For Record Lengths

When working with variable length record files, you can use the STATS operator to easily obtain the following information:

- The shortest record in the file (minimum)
- The longest record in the file (maximum)
- The average length of records in the file (average)
- The total number of bytes in the file (total)

ICETOOL provides the special ON(VLEN) field for printing these statistics. Specify ON(VLEN) as you would any other ON field.

So Far

Now you know how to print statistics for numeric fields and record lengths. In the next section, you will learn how to work with the ICETOOL SORT operator.

Creating Identical Sorted Files

You can use ICETOOL's SORT operator to create sorted output files. A single SORT operator can be used to create one output file or up to 10 identical output files. Using INCLUDE or OMIT statements, you can select a subset of the input records. Using INREC or OUTREC statements, you can rearrange the fields of the input records.

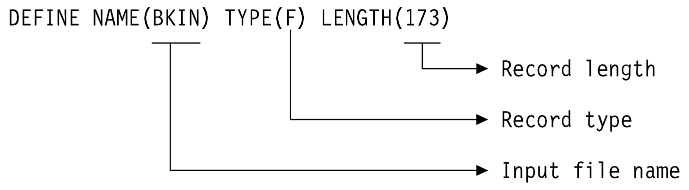
For this example, we will use both the sample bookstore file (SORT.SAMPIN) and the additional bookstore file (SORT.SAMPADD) as input.

To write a DEFINE operator statement for BKIN input file name:

Figure 46. Steps to Create the DEFINE Operator for BKIN Input File

Step	Action
1	Type **DEFINE** (you can leave one or more blanks before DEFINE if you like).
2	Leave at least one blank and type **NAME(BKIN)** NAME specifies the file name (that is, the first file name of the FROM operand on the SORT operator statement) for the input file from which you want to print statistics.
3	Leave at least one blank and type **TYPE(F)** TYPE specifies the type of record. In this case F (fixed length) is the record type.
4	Leave at least one blank and type **LENGTH(173)** LENGTH specifies the record length. In this case 173 bytes is the record length. Make sure that the statement is coded between columns 1 and 72.

When complete, the DEFINE operator statement looks like:



```
DEFINE NAME(BKIN) TYPE(F) LENGTH(173)
```

Since the input record type and input record length must be supplied with the DEFINE operator statement for the first input file name only, you do not need to add a DEFINE operator statement for BKADD input file name to the ICETOOL job.

To write a SORT operator that selects the books from publishers VALD and WETH, sorts them by publisher and title, and writes them to DASD file:

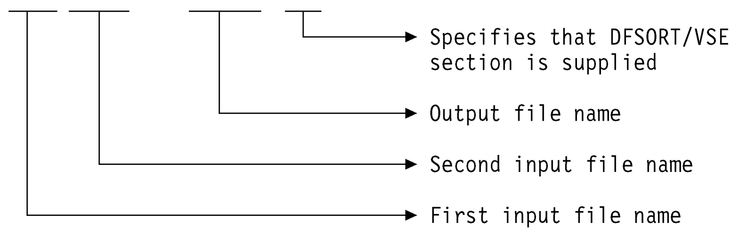
Figure 47. Steps to Create the SORT Operator

Step	Action
1	Write a comment statement (optional): `* Books from VALD and WETH`
2	Type **SORT** after the comment statement
3	Leave at least one blank and type **FROM(BKIN,BKADD)** BKIN and BKADD specify the file names for the input files you want to sort.
4	Leave at least one blank and type **TO(DAPUBS)** TO specifies the file name for the output file to contain the sorted subset of records. You can create up to 10 identical output files of any type that DFSORT/VSE allows. In this case, DAPUBS is the file name chosen for the temporary DASD file. You can use any valid 1-7 character file name you like.
5	Leave at least one blank and type **USE** USE specifies that the DFSORT/VSE section is supplied after the SORT operator statement. For the SORT operator, you must specify a DFSORT/VSE SORT control statement in the DFSORT/VSE section in order to tell DFSORT/VSE how to sort the input files. You can also specify additional DFSORT/VSE control statements, like INCLUDE, OMIT, INREC, and OUTREC, as appropriate.

6	Write the DFSORT/VSE section containing the USTART delimiter statement, the SORT control statement to sort the input files by publisher and title, the INCLUDE statement to select only the books by publishers VALD and WETH, and the UEND delimiter statement and place them after the SORT operator statement:
	`USTART`
	`   SORT FIELDS=(106,4,A,1,75,A),FORMAT=CH`
	`   INCLUDE COND=(106,4,EQ,C'VALD',OR,106,4,EQ,C'WETH'),`
	`           FORMAT=CH`
	`UEND`

When complete, the SORT operator statement looks like:

```
SORT FROM(BKIN,BKADD) TO(DAPUBS) USE
```



To write the JCL statements that go with the SORT operator:

Figure 48. Steps to Create JCL Statements for the SORT Operator

Step	Action

1	Write DLBL statements for the input files:
	`// DLBL BKIN,'SORT.SAMPIN',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)`
	`// DLBL BKADD,'SORT.SAMPADD',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)`

2	Write DLBL statement for the DASD file: `//DLBL DAPUBS,'%%SORT',,VSAM,RECORDS=100,RECSIZE=173,CAT=VSESPUC`

When complete, the ICETOOL statements and DLBL statements for this SORT operator look like:

```
// DLBL BKIN,'SORT.SAMPIN',,VSAM,CAT=VSESPUC
// DLBL BKADD,'SORT.SAMPADD',,VSAM,CAT=VSESPUC
// DLBL DAPUBS,'%%SORT',,VSAM,RECORDS=100,RECSIZE=173,CAT=VSESPUC
        . . .
 DEFINE NAME(BKIN) TYPE(F) LENGTH(173)
* Books from VALD and WETH
SORT FROM(BKIN,BKADD) TO(DAPUBS) USE
 USTART
    SORT FIELDS=(106,4,A,1,75,A),FORMAT=CH
    INCLUDE COND=(106,4,EQ,C'VALD',OR,106,4,EQ,C'WETH'),FORMAT=CH
 UEND
```

Figure 49 shows the Book Title and Publisher fields for the records as they would appear in the resulting output files. The actual records contain all of the fields.

```
Book
Title                                           Publisher
---------------------------------------------------------------------
1                                       75      106   109
---------------------------------------------------------------------
CELLS AND HOW THEY WORK                         VALD
COMPLETE SPANISH DICTIONARY                     VALD
EDITING SOFTWARE MANUALS                        VALD
FREUD'S THEORIES                                VALD
INTRODUCTION TO BIOLOGY                         VALD
NOVEL IDEAS                                     VALD
SHORT STORIES AND TALL TALES                    VALD
STRATEGIC MARKETING                             VALD
VIDEO GAME DESIGN                               VALD
ZEN BUSINESS                                    VALD
ANTICIPATING THE MARKET                         WETH
CIVILIZATION SINCE ROME FELL                    WETH
COMPUTERS: AN INTRODUCTION                      WETH
EIGHTEENTH CENTURY EUROPE                       WETH
GUIDE TO COLLEGE LIFE                           WETH
GUNTHER'S GERMAN DICTIONARY                     WETH
REBIRTH FROM ITALY                              WETH
SYSTEM PROGRAMMING                              WETH
THE INDUSTRIAL REVOLUTION                       WETH
```

Figure 49. Books from Publishers VALD and WETH

In the following sample, ICETOOL's SORT operator uses an input file and an output tape file; you must describe their characteristics using DEFINE operators. For output disk files, the DEFINE operator statement is not required.

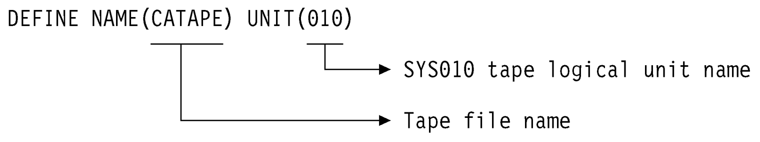
To write a DEFINE operator statement for ALL input file, see the STATS operator example.

To write a DEFINE operator statement for CATAPE output tape file:

Figure 50. Steps to Create the DEFINE Operator for CATAPE Output File

Step	Action
1	Type **DEFINE** (you can leave one or more blanks before DEFINE if you like).
2	Leave at least one blank and type **NAME(CATAPE)** NAME specifies the file name (that is, the tape file name of the TO operand on the SORT operator statement) for the output file you want to have sorted data.
3	Leave at least one blank and type **UNIT(010)** UNIT specifies the logical unit name for output tape. In this case SYS010 is the logical unit name for this tape. Make sure that the statement is coded between columns 1 and 72.

When complete, the DEFINE operator statement looks like:



```
DEFINE NAME(CATAPE) UNIT(010)
```

To write SORT operators that create separate DASD and tape files for the branch offices in California, and those in Colorado, sorted by city:

Figure 51 (Page 1 of 2). Steps to Create the SORT Operator

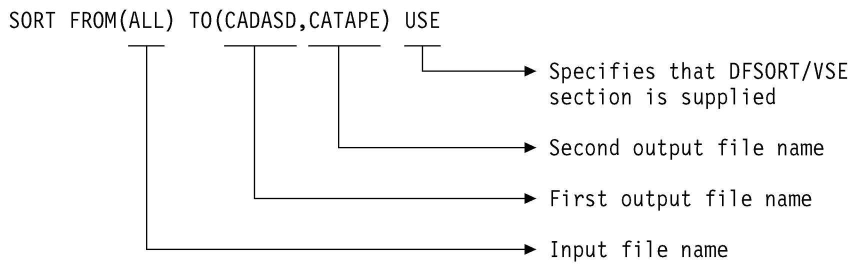
Step	Action
1	Write a comment statement (optional): `* Separate output for California and Colorado branches`
2	Type **SORT** after the comment statement
3	Leave at least one blank and type **FROM(ALL)** ALL specifies the file name for the input file you want to sort. You can use the same file name that you used for SORT.BRANCH in the STATS operator.
4	Leave at least one blank and type **TO(CADASD,CATAPE)** TO specifies the file names for the output files you want to create to contain the records for the California branches. You can create up to 10 identical output files of any type that DFSORT/VSE allows (permanent, temporary, DASD, tape). In this case, CADASD is the file name chosen for the temporary DASD file and CATAPE is the file name chosen for the tape file. You can use any valid 1-7 character file names you like. ICETOOL will create the first file and then copy it to the second and subsequent files.

Figure 51 (Page 2 of 2). Steps to Create the SORT Operator

Step	Action
5	Leave at least one blank and type **USE** USE specifies that DFSORT/VSE section is supplied after the SORT operator statement. For the SORT operator, you must specify a DFSORT/VSE SORT control statement in the DFSORT/VSE section in order to tell DFSORT/VSE how to sort the input file. You can also specify additional DFSORT/VSE control statements, like INCLUDE, OMIT, INREC, and OUTREC as appropriate.

6	Write the DFSORT/VSE section containing the USTART delimiter statement, the SORT control statement to sort the input file by city, the INCLUDE statement to select only the California branches, and the UEND delimiter statement and place them after the ICETOOL's SORT operator statement:
	
	`USTART`
	`   SORT FIELDS=(1,15,CH,A)`
	`   INCLUDE COND=(16,2,CH,EQ,C'CA')`
	`UEND`

When complete, the SORT operator statement looks like:



```
SORT FROM(ALL) TO(CADASD,CATAPE) USE
```

To write the JCL statements that go with the SORT operator:

Figure 52. Steps to Create the JCL Statements for SORT Operator

Step	Action

1	Write DLBL, TLBL, and ASSGN statements for the DASD and tape output files:
	
	`// DLBL CADASD,'%%CA',,VSAM,RECORDS=20,RECSIZE=33,CAT=VSESPUC`
	`// ASSGN SYS010,181`
	`// TLBL CATAPE,'CA.BRANCH',,111111,,1`

2	Write a DLBL statement for ALL input file: `// DLBL ALL,'SORT.BRANCH',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)`
3	Repeat steps 2 through 6 of the first procedure and 1 through 2 of the second procedure for the Colorado branches using unique names for the TO (such as CODASD and COTAPE) fields, and appropriate DFSORT/VSE control statements.

When complete, the DLBL and TLBL JCL statements and ICETOOL statements for these SORT operators look like:

```
// DLBL CADASD,'%%CA',,VSAM,RECORDS=20,RECSIZE=33,CAT=VSESPUC
// ASSGN SYS010,181
// TLBL CATAPE,'CA.BRANCH',,111111,,1
// DLBL CODASD,'%%CO',,VSAM,RECORDS=20,RECSIZE=33,CAT=VSESPUC
// ASSGN SYS011,181
// TLBL COTAPE,'CO.BRANCH',,111111,,2
      . . .
DEFINE NAME(CATAPE) UNIT(010)
DEFINE NAME(COTAPE) UNIT(011)
* Separate output for California and Colorado branches
SORT FROM(ALL) TO(CADASD,CATAPE) USE
USTART
   SORT FIELDS=(1,15,CH,A)
   INCLUDE COND=(16,2,CH,EQ,C'CA')
UEND
SORT FROM(ALL) TO(CODASD,COTAPE) USE
USTART
   SORT FIELDS=(1,15,CH,A)
   INCLUDE COND=(16,2,CH,EQ,C'CO')
UEND
```

Notes:

1. DFSORT/VSE's RECORD statement is not needed for the ICETOOL utility.
2. Since the ALL input file was previously defined, you do not need to add a new JCL statement and a DEFINE operator to the ICETOOL job.

Figure 53 shows the records as they would appear in the CADASD file (%%CA) and the CATAPE file (CA.BRANCH) as a result of using the first SORT operator.

Figure 53. Records for California Sorted by City

City	State	Employees	Revenue	Profit
1 15	16 17	18 21	22 27	28 33
Los Angeles	CA	32	22530	–4278
Morgan Hill	CA	15	18200	3271
Sacramento	CA	29	42726	8276
San Diego	CA	22	32940	8275
San Francisco	CA	35	42820	6832
San Jose	CA	21	27225	8264
Sunnyvale	CA	18	16152	–978

Figure 54 shows the records as they would appear in the CODASD file (%%CO) and the COTAPE file (CO.BRANCH) as a result of using the second SORT operator.

Figure 54. Records for Colorado Sorted by City

City	State	Employees	Revenue	Profit
1 15	16 17	18 21	22 27	28 33
Aspen	CO	20	25800	5200
Boulder	CO	32	33866	7351
Denver	CO	33	31876	6288
Fort Collins	CO	22	12300	–2863
Vail	CO	19	23202	5027

Figure 58 on page 89 shows the complete output.

So Far

So far in this tutorial, you have learned how to print statistics for numeric fields using the ICETOOL STATS operator, and how to sort an input file and create multiple sorted output files using the ICETOOL SORT operator. Next, you will learn about the COPY operator.

Creating Multiple Unsorted Files

If you want to make unsorted copies of an input file, you can use the COPY operator. The COPY operator does not require any DFSORT/VSE statements. However, you can supply DFSORT/VSE statements (for example, INCLUDE, OMIT) if appropriate. You must provide the DEFINE operator statement for each COPY's input file and output tape file.

Here are a couple of examples of COPY operator statements with their accompanying JCL statements. Since the ALL input file was previously defined, you do not need to add a new JCL statement and DEFINE operator statement to the ICETOOL job.

```
         . . .
// DLBL D1,'SORT.COPY1',,VSAM,RECORDS=50,RECSIZE=33,CAT=VSESPUC
// DLBL D2,'SORT.COPY2',,VSAM,RECORDS=50,RECSIZE=33,CAT=VSESPUC
// DLBL D3,'SORT.COPY3',,VSAM,RECORDS=50,RECSIZE=33,CAT=VSESPUC
// DLBL D4,'SORT.COPY4',,VSAM,RECORDS=50,RECSIZE=33,CAT=VSESPUC
         . . .
  COPY FROM(ALL) TO(D1,D2,D3)
  COPY FROM(ALL) TO(D4) USE
USTART
   INCLUDE COND=(16,2,CH,EQ,C'CA')
UEND
```

The first COPY operator creates identical copies of SORT.BRANCH file in SORT.COPY1, SORT.COPY2, and SORT.COPY3 files.

The second COPY operator creates the SORT.BRANCH records for the branches in California in SORT.COPY4 file.

Since copying is more efficient than sorting, you should use the COPY operator rather than the SORT operator when possible.

So Far

So far in this chapter you have learned about STATS, SORT, and COPY, three important ICETOOL operators. The next tutorial shows you how to use the RANGE operator.

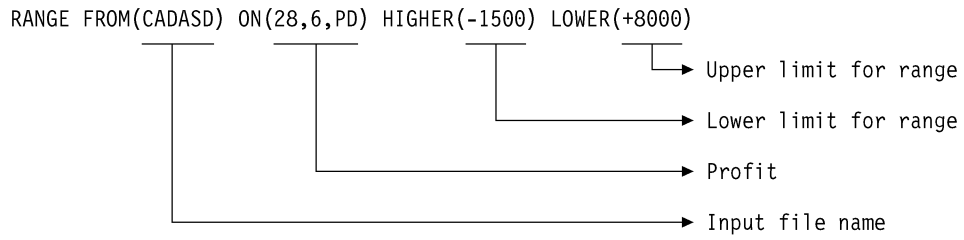
Counting Values in a Range

You can use ICETOOL's RANGE operator to count the number of values for a particular numeric field that fall within a range you define. The range can be defined with:

Operand	**Comparison**
EQUAL	Equal to a value
NOTEQUAL	Not equal to a value
LOWER	Less than a value
HIGHER	Greater than a value
HIGHER and LOWER	Greater than a value, but less than another value

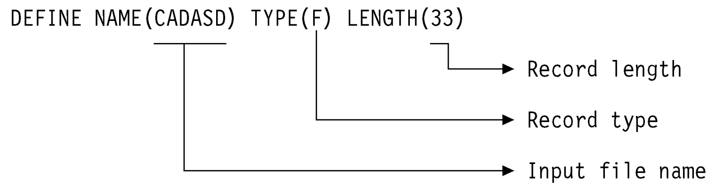
To print a count of the number of California branches with profit greater than −1500, but less than +8000, write the following RANGE operator statement:

```
* California branches profit analysis
```



```
 RANGE FROM(CADASD) ON(28,6,PD) HIGHER(-1500) LOWER(+8000)
```

You must add the DEFINE operator statement for the CADASD input file:



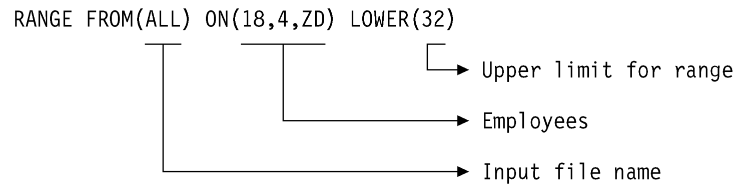
```
 DEFINE NAME(CADASD) TYPE(F) LENGTH(33)
```

This DEFINE operator statement must be placed before the RANGE operator.

The input file defined by CADASD is the same file you created earlier (with the SORT operator) for the California branches. HIGHER(−1500) indicates that you want to count values in the profit field that are greater than −1500, while LOWER(+8000) indicates that you want to count values in the profit field that are less than +8000. For a negative limit, you must specify a minus (−) sign before the number. For a positive limit, you either specify a plus (+) sign before the number or leave it out, therefore, HIGHER(8000) is the same as HIGHER(+8000).

To print a count of the number of branches (in California and Colorado) with less than 32 employees, write the following RANGE operator statement:

```
* Branches with less than 32 employees

 RANGE FROM(ALL) ON(18,4,ZD) LOWER(32)
```



Since CADASD and ALL were previously defined, you do not need to add any new JCL statements and DEFINE operators to the ICETOOL job.

When these RANGE operators are run, the results are printed to SYSLST. The ICETOOL output produced for these RANGE operators would look like:

```
           * California branches profit analysis
           RANGE FROM(CADASD) ON(28,6,PD) HIGHER(-1500) LOWER(+8000)
ILU627I T0 DFSORT/VSE CALL 0011 FOR COPY FROM CADASD  TO E35 EXIT COMPLETED
ILU628I T0 RECORD COUNT:  000000000000007
ILU631I T0 NUMBER OF VALUES IN RANGE FOR (28,6,PD)     :  000000000000003
ILU602I T0 OPERATION RETURN CODE:  00

           * Branches with less than 32 employees
           RANGE FROM(ALL) ON(18,4,ZD) LOWER(32)
ILU627I T0 DFSORT/VSE CALL 0012 FOR COPY FROM ALL     TO E35 EXIT COMPLETED
ILU628I T0 RECORD COUNT:  000000000000012
ILU631I T0 NUMBER OF VALUES IN RANGE FOR (18,4,ZD)     :  000000000000008
ILU602I T0 OPERATION RETURN CODE:  00
```

Looking at the output, you will notice that:

- Message ILU628I gives the count of records processed.
- Message ILU631I gives the count of values in the specified range. There were 3 California branches with profit greater than −1500, but less than +8000, and 8 branches in all with less than 32 employees.
- A return code for each operator is given in message ILU602I.

So Far

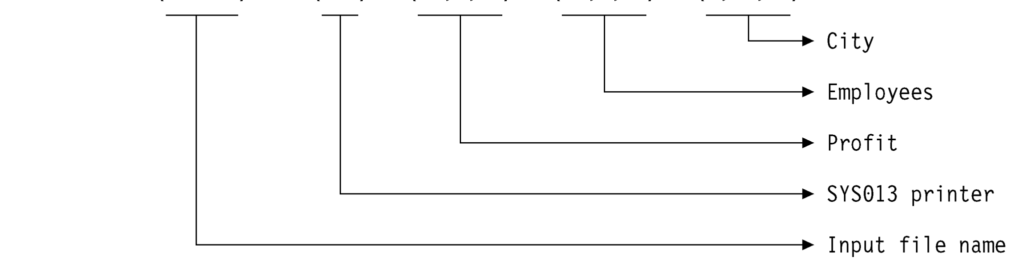
You have now learned how to count the number of values in a range for a particular field using the ICETOOL RANGE operator. Next, you will learn about the DISPLAY operator.

Printing Simple Reports

Numeric fields are often in a format (binary, fixed-point, or decimal) that is not readable when printed. You can use ICETOOL's DISPLAY operator to print numeric and character fields from an input file in readable form. The specified fields are printed to the printer or to SYSLST that you define. For numeric fields, appropriate plus (+) and minus (−) signs are sent to the printer along with the decimal value of each number.

To print a report showing the profit, employees, and city fields for the Colorado branches, write the following DISPLAY operator statement:

```
* Print profit, employees, and city for each Colorado branch

 DISPLAY FROM(CODASD) LIST(013) ON(28,6,PD) ON(18,4,ZD) ON(1,15,CH)
```



The input file defined by CODASD is the same file you created earlier (with the SORT operator) for the Colorado branches. LIST specifies the SYS013 printer you want the fields to be printed on. Specify the ON fields in the same order in which you want their values to be printed.

Since SYS013 has not been defined previously, you must add an ASSGN JCL statement for it looks like:

```
 // ASSGN SYS013,SYSLST
```

You must add the DEFINE operator statement for the CODASD file name of the DISPLAY operator.

```
DEFINE NAME(CODASD) TYPE(F) LENGTH(33)
```

This DEFINE operator statement must be placed before the DISPLAY operator.

When this DISPLAY operator is run, the report looks like:

```
 (28,6,PD)          (18,4,ZD)          (1,15,CH)
 +000000000005200   +000000000000020   Aspen
 +000000000007351   +000000000000032   Boulder
 +000000000006288   +000000000000033   Denver
 -000000000002863   +000000000000022   Fort Collins
 +000000000005027   +000000000000019   Vail
```

The values for profit, employees, and city are printed in separate columns across the page with a header for each column at the top. If more than one page is printed, DISPLAY puts the header at the top of each page. If you do not want the header printed, you can use DISPLAY's NOHEADER operand to suppress it.

DISPLAY also has two special ON fields you can use:

- **ON(VLEN)** can be used for variable length record files to print the length of each record.
- **ON(NUM)** can be used to print a relative record number for each record (starting with 1).

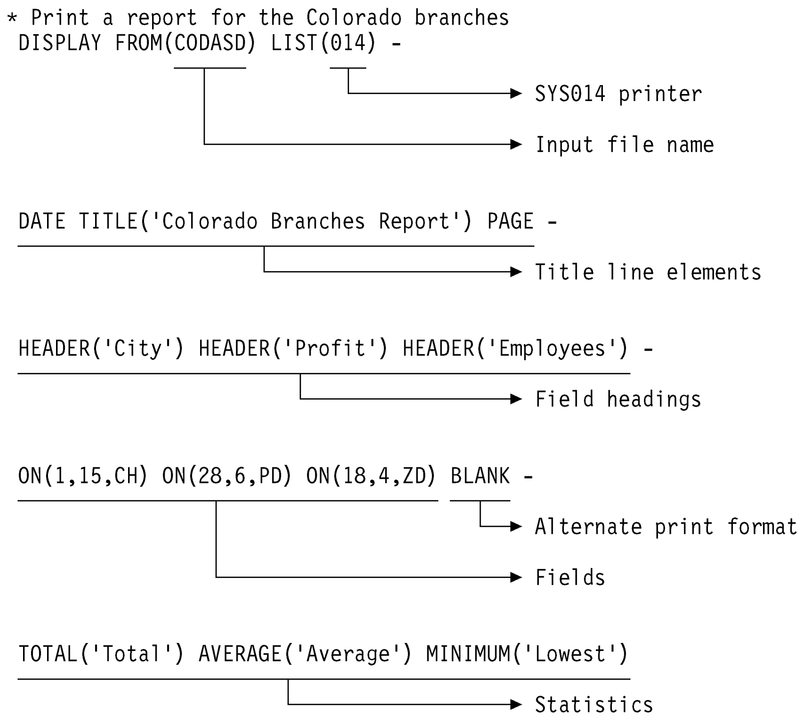
Use the ON(VLEN) or ON(NUM) field just as you would any other ON field.

Printing Tailored Reports

The previous tutorial showed you how to print a simple listing of numeric and character fields using the DISPLAY operator. By using additional operands of DISPLAY, you can create list files showing character and numeric fields in a variety of report formats. You can specify:

- Title elements (TITLE, DATE, PAGE, and TIME operands)
- Field headings (HEADER operand)
- Field formats (BLANK and PLUS operands)
- Statistics (TOTAL, AVERAGE, MAXIMUM, and MINIMUM operands)
- Lines per page (LINES operand)

To print a report for the Colorado branches showing the city, profit and employee fields with a title line, field headings, totals, averages and minimums, write the following DISPLAY operator statement:



```
* Print a report for the Colorado branches
 DISPLAY FROM(CODASD) LIST(014) -
 DATE TITLE('Colorado Branches Report') PAGE -
 HEADER('City') HEADER('Profit') HEADER('Employees') -
 ON(1,15,CH) ON(28,6,PD) ON(18,4,ZD) BLANK -
 TOTAL('Total') AVERAGE('Average') MINIMUM('Lowest')
```

CODASD is the file name for the previously created Colorado branches file. LIST specifies the SYS014 printer you want the report to be printed.

DATE, TITLE and PAGE indicate the elements to be included in the title line and their placement. Specify these operands in the same order in which you want them to be printed.

Each HEADER indicates a heading to be used for the corresponding field. Specify the ON fields and their corresponding HEADER strings in the same order in which you want their values to be printed. BLANK specifies that numeric values are printed with blank for plus sign, - for minus sign and no leading zeros. HEADER and BLANK also change the justification and column width for the fields to produce a more report-like format.

TOTAL, AVERAGE and MINIMUM cause the indicated statistics to be produced at the end of the report, identified by the specified strings.

Since SYS014 has not been defined previously, you must add an ASSGN JCL statement for it looks like:

```
 // ASSGN SYS014,SYSLST
```

When this DISPLAY operator is run, the report looks like:

```
06/01/98       Colorado Branches Report       - 1 -

City                         Profit         Employees
---------------   ----------------   ----------------
Aspen                          5200                20
Boulder                        7351                32
Denver                         6288                33
Fort Collins                  -2863                22
Vail                           5027                19

Total                         21003               126

Average                        4200                25

Lowest                        -2863                19
```

The title line and heading line are printed at the top of each page. The character data is left-justified and the numeric data is right-justified with zeros suppressed. The statistics are printed after the columns of data.

Using Formatting Items

The previous tutorial used the BLANK operand to change the way all numeric values in the report are displayed. You can use formatting items to change the appearance of individual numeric fields and their related statistics in the report, with respect to separators, decimal point, decimal places, signs, division, and leading, floating and trailing strings. Formatting items are written as part of the ON operand, separated by commas, as follows: ON(p,m,f,formatting).

Edit Masks

You can select from thirty-three predefined edit masks. The table below describes the available masks and shows how the values 12345678 and -1234567 would be printed for each mask. In the pattern:

- **d** is used to represent a decimal digit (0-9).
- **w** is used to represent a leading sign that will be blank for a positive value or - for a negative value.
- **x** is used to represent a trailing sign that will be blank for a positive value or - for a negative value.
- **y** is used to represent a leading sign that will be blank for a positive value or (for a negative value.
- **z** is used to represent a trailing sign that will be blank for a positive value or) for a negative value.

Figure 55. Edit Mask Patterns

Mask	Pattern	12345678	-1234567
A0	wdddddddddddddddd	12345678	-1234567
A1	wddd,ddd,ddd,ddd,ddd	12,345,678	-1,234,567
A2	wddd.ddd.ddd.ddd.ddd	12.345.678	-1.234.567
A3	wddd ddd ddd ddd ddd	12 345 678	-1 234 567
A4	wddd'ddd'ddd'ddd'ddd	12'345'678	-1'234'567
A5	ddd ddd ddd ddd dddx	12 345 678	1 234 567-
B1	wdd,ddd,ddd,ddd,ddd.d	1,234,567.8	-123,456.7
B2	wdd.ddd.ddd.ddd.ddd,d	1.234.567,8	-123.456,7
B3	wdd ddd ddd ddd ddd,d	1 234 567,8	-123 456,7
B4	wdd'ddd'ddd'ddd'ddd.d	1'234'567.8	-123'456.7
B5	wdd'ddd'ddd'ddd'ddd,d	1'234'567,8	-123'456,7
B6	dd ddd ddd ddd ddd,dx	1 234 567,8	123 456,7-
C1	wd,ddd,ddd,ddd,ddd.dd	123,456.78	-12,345.67
C2	wd.ddd.ddd.ddd.ddd,dd	123.456,78	-12.345,67
C3	wd ddd ddd ddd ddd,dd	123 456,78	-12 345,67
C4	wd'ddd'ddd'ddd'ddd.dd	123'456.78	-12'345.67
C5	wd'ddd'ddd'ddd'ddd,dd	123'456,78	-12'345,67
C6	d ddd ddd ddd ddd,ddx	123 456,78	12 345,67-
D1	wddd,ddd,ddd,ddd.ddd	12,345.678	-1,234.567
D2	wddd.ddd.ddd.ddd,ddd	12.345,678	-1.234,567
D3	wddd ddd ddd ddd,ddd	12 345,678	-1 234,567
D4	wddd'ddd'ddd'ddd.ddd	12'345.678	-1'234.567
D5	wddd'ddd'ddd'ddd,ddd	12'345,678	-1'234,567
D6	ddd ddd ddd ddd,dddx	12 345,678	1 234,567-
E1	yddd,ddd,ddd,ddd,dddz	12,345,678	(1,234,567)
E2	yddd.ddd.ddd.ddd.dddz	12.345.678	(1.234.567)
E3	yddd ddd ddd ddd dddz	12 345 678	(1 234 567)
E4	yddd'ddd'ddd'ddd'dddz	12'345'678	(1'234'567)
F1	yd,ddd,ddd,ddd,ddd.ddz	123,456.78	(12,345.67)
F2	yd.ddd.ddd.ddd.ddd,ddz	123.456,78	(12.345,67)
F3	yd ddd ddd ddd ddd,ddz	123 456,78	(12 345,67)
F4	yd'ddd'ddd'ddd'ddd.ddz	123'456.78	(12'345.67)
F5	yd'ddd'ddd'ddd'ddd,ddz	123'456,78	(12'345,67)

To use the E1 edit mask for the Profit field in the previous report, just change ON(28,6,PD) to ON(28,6,PD,**E1**). The resulting report then looks like:

```
06/01/98       Colorado Branches Report       - 1 -

City                           Profit         Employees
---------------   ---------------------   ----------------
Aspen                           5,200                20
Boulder                         7,351                32
Denver                          6,288                33
Fort Collins                   (2,863)               22
Vail                            5,027                19

Total                          21,003               126

Average                         4,200                25

Lowest                         (2,863)               19
```

Division

You can select from six division items as follows:

- **/K** - divide by 1000
- **/M** - divide by 1000000 (1000*1000)
- **/G** - divide by 1000000000 (1000*1000*1000)
- **/KB** - divide by 1024
- **/MB** - divide by 1048576 (1024*1024)
- **/GB** - divide by 1073741824 (1024*1024*1024)

The Profit values from SORT.BRANCH would look as follows with HEADER('Profit/(Loss) in K$') and ON(28,6,PD,E1,**/K**):

```
Profit/(Loss) in K$
-------------------
               (4)
                6
               (2)
                8
                0
                6
                7
                3
                5
                8
                8
                5
```

Leading, Floating and Trailing Characters

You can add floating characters to your numeric fields and add leading and trailing characters to your numeric and character fields as follows:

- **F'string'** - a floating string to appear to the left of the first nonblank character of the formatted numeric data.
- **L'string'** - a leading string to appear at the beginning of the character or numeric data column.

- **T'string'** - a trailing string to appear at the end of the character or numeric data column.

The Profit values from SORT.BRANCH would look as follows with HEADER('Profit') and ON(28,6,PD,A1,**F'$',T'**'**):

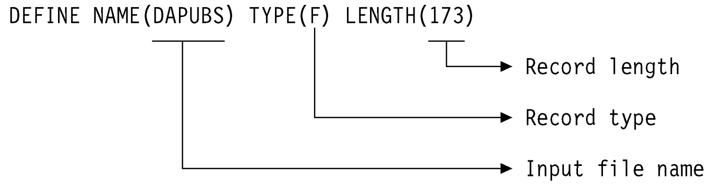
```
           Profit
------------------
        $-4,278**
         $6,832**
        $-2,863**
         $8,276**
          $-978**
         $6,288**
         $7,351**
         $3,271**
         $5,027**
         $8,264**
         $8,275**
         $5,200**
```

Printing Sectioned Reports

The previous tutorial showed you how to print tailored reports using the DISPLAY operator. By using the BREAK operand of DISPLAY, you can create reports divided into sections by a character or numeric break field on which you have previously sorted. You can also specify a string for the break title (BTITLE operand) and statistics for the individual sections (BTOTAL, BAVERAGE, BMAXIMUM, and BMINIMUM operands).

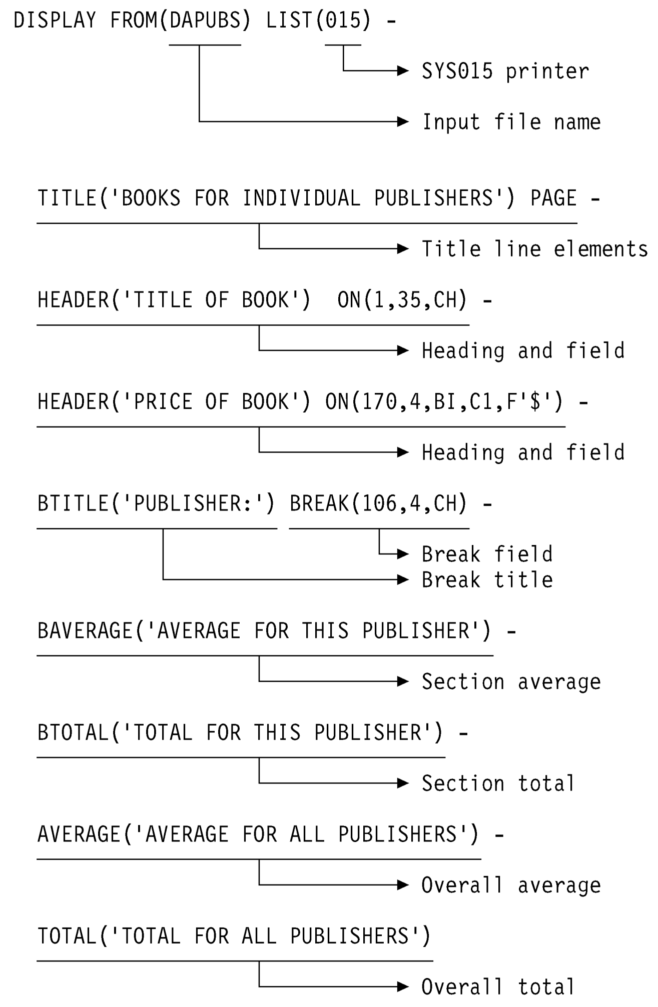
For this example, we will use the file with books from publishers VALD and WETH, sorted by publisher and title, that we created previously. To define input file characteristics for the DISPLAY operator write the following DEFINE operator statement:

```
DEFINE NAME(DAPUBS) TYPE(F) LENGTH(173)
```



To print a report with sections by publisher showing the title and price fields with a title line, field headings, break title, break averages and totals, and overall averages and totals, write the following DISPLAY operator statement:

```
* Print a report of books for individual publishers
```



```
 DISPLAY FROM(DAPUBS) LIST(015) -
   TITLE('BOOKS FOR INDIVIDUAL PUBLISHERS') PAGE -
   HEADER('TITLE OF BOOK')  ON(1,35,CH) -
   HEADER('PRICE OF BOOK') ON(170,4,BI,C1,F'$') -
   BTITLE('PUBLISHER:') BREAK(106,4,CH) -
   BAVERAGE('AVERAGE FOR THIS PUBLISHER') -
   BTOTAL('TOTAL FOR THIS PUBLISHER') -
   AVERAGE('AVERAGE FOR ALL PUBLISHERS') -
   TOTAL('TOTAL FOR ALL PUBLISHERS')
```

DAPUBS is the file name for the previously created VALD and WETH file. LIST specifies the SYS015 printer you want the report to be printed.

TITLE and PAGE indicate the elements to be included in the title line and their placement.

Each HEADER and ON pair indicate a field to be included in the report and the heading to be used for it.

BTITLE indicates a string to be used for the break title and its placement (before or after the break field). BREAK indicates the break field to be used to create sections. BAVERAGE and BTOTAL indicate section statistics to be produced at the end of each section.

AVERAGE and TOTAL indicate overall statistics to be produced at the end of the report.

Since SYS015 has not been defined previously, you must add an ASSGN JCL statement for it looks like:

```
// ASSGN SYS015,SYSLST
```

When this DISPLAY operator is run, the SYSLST results in a three-page report that looks as follows:

```
BOOKS FOR INDIVIDUAL PUBLISHERS       - 1 -

PUBLISHER:  VALD

TITLE OF BOOK                                  PRICE OF BOOK
-----------------------------------   ----------------------
CELLS AND HOW THEY WORK                               $24.95
COMPLETE SPANISH DICTIONARY                            $6.50
EDITING SOFTWARE MANUALS                              $14.50
FREUD'S THEORIES                                      $12.50
INTRODUCTION TO BIOLOGY                               $23.50
NOVEL IDEAS                                           $24.50
SHORT STORIES AND TALL TALES                          $15.20
STRATEGIC MARKETING                                   $23.50
VIDEO GAME DESIGN                                     $21.99
ZEN BUSINESS                                          $12.00

AVERAGE FOR THIS PUBLISHER                            $17.91

TOTAL FOR THIS PUBLISHER                             $179.14


BOOKS FOR INDIVIDUAL PUBLISHERS       - 2 -

PUBLISHER:  WETH

TITLE OF BOOK                                  PRICE OF BOOK
-----------------------------------   ----------------------
ANTICIPATING THE MARKET                               $20.00
CIVILIZATION SINCE ROME FELL                          $13.50
COMPUTERS: AN INTRODUCTION                            $18.99
EIGHTEENTH CENTURY EUROPE                             $17.90
GUIDE TO COLLEGE LIFE                                 $20.00
GUNTHER'S GERMAN DICTIONARY                           $10.88
REBIRTH FROM ITALY                                    $25.60
SYSTEM PROGRAMMING                                    $31.95
THE INDUSTRIAL REVOLUTION                              $7.95

AVERAGE FOR THIS PUBLISHER                            $18.53

TOTAL FOR THIS PUBLISHER                             $166.77
```

```
BOOKS FOR INDIVIDUAL PUBLISHERS       - 3 -

TITLE OF BOOK                                   PRICE OF BOOK
-----------------------------------   ----------------------

AVERAGE FOR ALL PUBLISHERS                             $18.20

TOTAL FOR ALL PUBLISHERS                              $345.91
```

So Far

You have now learned how to print simple, tailored and sectioned reports using the ICETOOL DISPLAY operator. Next, you will learn about the OCCUR operator.

Printing How Many Times Fields Occur

You can use ICETOOL's OCCUR operator to print a simple or tailored report showing how many times different field values occur. You can list all of the values in the file or limit the listing to those values that occur:

- More than once, that is, duplicate values (ALLDUPS operand)
- Only once, that is, nonduplicate values (NODUPS operand)
- A specified number of times (EQUAL operand)
- More than a specified number of times (HIGHER operand)
- Less than a specified number of times (LOWER operand)

For this example, we will use the sample bookstore file as input. To print a report showing the number of different books in use from each publisher, write the following OCCUR operator statement:

```
* Print the count of books in use from each publisher
 OCCUR    FROM(BKIN) LIST(016) BLANK -
                               └──────► Alternate print format
                         └────────────► SYS016 printer
               └──────────────────────► Input file name

 TITLE('Books from Publishers') DATE(DMY.) -
                     └────────────────► Title line elements

 HEADER('Publisher') HEADER('Books in Use') -
                     └────────────────► Field headings

 ON(106,4,CH) ON(VALCNT)
             └────────────────────────► Publisher and Count
```

BKIN is the file name for the sample bookstore file. LIST specifies the SYS016 printer you want the report to be printed. BLANK specifies the field format to be used.

TITLE indicates the title string to appear in the title line. DATE indicates the format in which the date is to appear in the title line (dd.mm.yy where dd is the two-digit day, mm is the two-digit month and yy is the last two digits of the year).

The HEADER strings correspond to the ON fields. ON(VALCNT) is a special ON field used with OCCUR to print the count of occurrences.

Write appropriate ASSGN statement for the SYS016 printer and place it into the job:

```
// ASSGN SYS016,SYSLST
```

Since BKIN input file was previously defined, you do not need to add any new JCL statements and DEFINE operator to the ICETOOL job.

When this OCCUR operator is run, the resulting report looks like:

```
Books from Publishers       01.06.98

Publisher      Books in Use
---------   ---------------
COR                       7
FERN                      4
VALD                      5
WETH                      4
```

The name of each publisher is printed along with the number of times that publisher appeared in the sample bookstore file which is equivalent to the number of different books from that publisher.

So Far

You have now learned how to print a simple or tailored report showing how many times different field values occur. Next, you will learn how to use the SELECT operator.

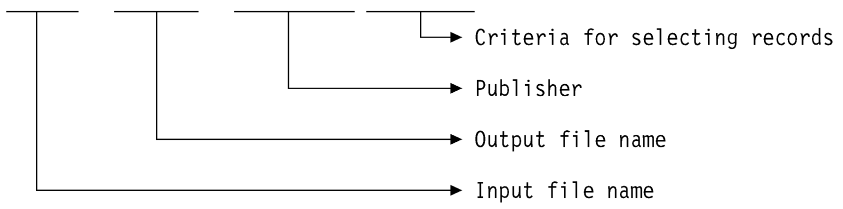
Selecting Records by Field Occurrences

You can use ICETOOL's SELECT operator to create an output file with records selected according to how many times different field values occur. You can keep only the first record for each value (FIRST operand), only the last record for each value (LAST operand), or only those records with values that occur:

- More than once, that is, duplicate values (ALLDUPS operand)
- Only once, that is, nonduplicate values (NODUPS operand)
- A specified number of times (EQUAL operand)
- More than a specified number of times (HIGHER operand)
- Less than a specified number of times (LOWER operand)

To create an output file containing records for publishers with more than four different books in use, write the following SELECT operator statement (since BKIN input file was previously defined, you do not need to add any new JCL statements and DEFINE operator to the ICETOOL job):

```
* Separate output containing records for publishers
* with more than 4 books in use
 SELECT FROM(BKIN) TO(BKOUT) ON(106,4,CH) HIGHER(4)
```



BKIN is the file name for the sample bookstore file. BKOUT is the file name of the output file that will contain the records for each publisher field value that occurs more than 4 times (all of the records for COR and VALD in this case).

Write appropriate DLBL statement for the SORT.BOOKS1 file and place it into the job:

```
// DLBL BKOUT,'SORT.BOOKS1',,VSAM,RECORDS=100,RECSIZE=173,CAT=VSESPUC
```

Figure 56 shows the Book Title and Publisher fields for the records in the resulting report. The actual records contain all of the fields.

```
Book
Title                                       Publisher
----------------------------------------------------------------------
1                                       75      106   109
----------------------------------------------------------------------
LIVING WELL ON A SMALL BUDGET               COR
SUPPLYING THE DEMAND                        COR
INKLINGS: AN ANTHOLOGY OF YOUNG POETS       COR
PICK'S POCKET DICTIONARY                    COR
MODERN ANTHOLOGY OF WOMEN POETS             COR
INTRODUCTION TO PSYCHOLOGY                  COR
CRISES OF THE MIDDLE AGES                   COR
VIDEO GAME DESIGN                           VALD
EDITING SOFTWARE MANUALS                    VALD
STRATEGIC MARKETING                         VALD
SHORT STORIES AND TALL TALES                VALD
INTRODUCTION TO BIOLOGY                     VALD
```

Figure 56. Books from Publishers with More Than Four Books in Use

So Far

So far in this chapter you have learned how to print statistics for numeric fields, create sorted and unsorted files, obtain a count of numeric fields in a range for a particular field, print fields from an input file, print reports, print a count of field occurrences and select output records based on field occurrences. The last part of this chapter shows the complete ICETOOL job and its resulting output.

Complete ICETOOL Job and its Output

Here is the complete ICETOOL job you created in this chapter:

```
* $$ JOB JNM=PGMA
* $$ LST CLASS=E
// JOB   PGMA JOBA,PROGRAMMER
// ASSGN SYS012,FEF
* $$ LST CLASS=E,JNM=DFSMSG,LST=SYS012
// OPTION NOSYSDMP
// ASSGN SYS013,SYSLST
// ASSGN SYS014,SYSLST
// ASSGN SYS015,SYSLST
// ASSGN SYS016,SYSLST
// DLBL VSESPUC,'VSESP.USER.CATALOG',,VSAM
// DLBL ALL,'SORT.BRANCH',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// DLBL CADASD,'%%CA',,VSAM,RECORDS=20,RECSIZE=33,CAT=VSESPUC
// DLBL CODASD,'%%CO',,VSAM,RECORDS=20,RECSIZE=33,CAT=VSESPUC
// DLBL D1,'SORT.COPY1',,VSAM,RECORDS=20,RECSIZE=33,CAT=VSESPUC
// DLBL D2,'SORT.COPY2',,VSAM,RECORDS=20,RECSIZE=33,CAT=VSESPUC
// DLBL D3,'SORT.COPY3',,VSAM,RECORDS=20,RECSIZE=33,CAT=VSESPUC
// DLBL D4,'SORT.COPY4',,VSAM,RECORDS=20,RECSIZE=33,CAT=VSESPUC
// ASSGN SYS010,181
// TLBL CATAPE,'CA.BRANCH',,111111,,1
// ASSGN SYS011,181
// TLBL COTAPE,'CO.BRANCH',,111111,,2
// DLBL BKIN,'SORT.SAMPIN',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// DLBL BKADD,'SORT.SAMPADD',,VSAM,CAT=VSESPUC,DISP=(OLD,KEEP)
// DLBL DAPUBS,'%%SORT',,VSAM,RECORDS=100,RECSIZE=173,CAT=VSESPUC
// DLBL BKOUT,'SORT.BOOKS1',,VSAM,RECORDS=100,RECSIZE=173,CAT=VSESPUC
// EXEC ICETOOL,SIZE=100K

* Input files characteristics for ICETOOL operations
DEFINE NAME(ALL) TYPE(F) LENGTH(33)
DEFINE NAME(CODASD) TYPE(F) LENGTH(33)
DEFINE NAME(CADASD) TYPE(F) LENGTH(33)
DEFINE NAME(BKADD) TYPE(F) LENGTH(173)
DEFINE NAME(BKIN) TYPE(F) LENGTH(173)
DEFINE NAME(BKOUT) TYPE(F) LENGTH(173)
DEFINE NAME(DAPUBS) TYPE(F) LENGTH(173)

* Tape output files characteristics
DEFINE NAME(COTAPE) UNIT(011)
DEFINE NAME(CATAPE) UNIT(010)

* DFSORT/VSE messages printer queue
DEFINE ROUTE(012)
```

Figure 57 (Part 1 of 2). Complete ICETOOL Job.

```
* Statistics from all branches
STATS FROM(ALL) ON(18,4,ZD) ON(28,6,PD) ON(22,6,PD)
* Books from VALD and WETH
SORT FROM(BKIN,BKADD) TO(DAPUBS) USE
USTART
   SORT FIELDS=(106,4,A,1,75,A),FORMAT=CH
   INCLUDE COND=(106,4,EQ,C'VALD',OR,106,4,EQ,C'WETH'),
           FORMAT=CH
UEND
* Separate output for California and Colorado branches
SORT FROM(ALL) TO(CADASD,CATAPE) USE
USTART
  SORT FIELDS=(1,15,CH,A)
  INCLUDE COND=(16,2,CH,EQ,C'CA')
UEND
SORT FROM(ALL) TO(CODASD,COTAPE) USE
USTART
   SORT FIELDS=(1,15,CH,A)
   INCLUDE COND=(16,2,CH,EQ,C'CO')
UEND
* Copy multiple files
COPY FROM(ALL) TO(D1,D2,D3)
COPY FROM(ALL) TO(D4) USE
USTART
   INCLUDE COND=(16,2,CH,EQ,C'CA')
UEND
* California branches profit analysis
RANGE FROM(CADASD) ON(28,6,PD) HIGHER(-1500) LOWER(+8000)
* Branches with less than 32 employees
RANGE FROM(ALL) ON(18,4,ZD) LOWER(32)
* Print profit, employees, and city for each Colorado branch
DISPLAY FROM(CODASD) LIST(013) ON(28,6,PD) ON(18,4,ZD) ON(1,15,CH)
* Print a report for the Colorado branches
DISPLAY FROM(CODASD) LIST(014) -
    DATE TITLE('Colorado Branches Report') PAGE -
    HEADER('City') HEADER('Profit') HEADER('Employees') -
    ON(1,15,CH) ON(28,6,PD) ON(18,4,ZD) BLANK -
    TOTAL('Total') AVERAGE('Average') MINIMUM('Lowest')
DISPLAY FROM(DAPUBS) LIST(015) -
    TITLE('Books for individual publishers') PAGE -
    HEADER('Title of book') ON(1,35,CH) -
    HEADER('Price of book') ON(170,4,BI,C1,F'$') -
    BTITLE('Publisher:') BREAK(106,4,CH) -
    BAVERAGE('Average for this publisher') -
    BTOTAL('Total for this publisher') -
    AVERAGE('Average for all publishers') -
    TOTAL('Total for all publishers')
* Print the count of books in use from each publisher
OCCUR FROM(BKIN) LIST(016) BLANK -
    TITLE('Books from Publishers') DATE(DMY.) -
    HEADER('Publisher') HEADER('Books in use') -
    ON(106,4,CH) ON(VALCNT)
* Separate output containing records for publishers
* with more than 4 books in use
SELECT FROM(BKIN) TO(BKOUT) ON(106,4,CH) HIGHER(4)
/*
/&
* $$ EOJ
```

Figure 57 (Part 2 of 2). Complete ICETOOL Job.

Here is the complete ICETOOL output that results from running this job:

```
ILU600I T0 ICETOOL UTILITY RUN STARTED
ILU630I T0 MODE IN EFFECT:  STOP

ILU632I T0 SOURCE FOR ICETOOL STATEMENTS:  SYSIPT

           * All input files characteristics
           DEFINE NAME(ALL) TYPE(F) LENGTH(33)
ILU602I T0 OPERATION RETURN CODE:  00

           DEFINE NAME(CODASD) TYPE(F) LENGTH(33)
ILU602I T0 OPERATION RETURN CODE:  00

           DEFINE NAME(CADASD) TYPE(F) LENGTH(33)
ILU602I T0 OPERATION RETURN CODE:  00

           DEFINE NAME(BKIN) TYPE(F) LENGTH(173)
ILU602I T0 OPERATION RETURN CODE:  00

           DEFINE NAME(DAPUBS) TYPE(F) LENGTH(173)
ILU602I T0 OPERATION RETURN CODE:  00

           * Tape output files characteristics
           DEFINE NAME(COTAPE) UNIT(011)
ILU602I T0 OPERATION RETURN CODE:  00

           DEFINE NAME(CATAPE) UNIT(010)
ILU602I T0 OPERATION RETURN CODE:  00

           * DFSORT/VSE messages printer queue
           DEFINE ROUTE(012)
ILU602I T0 OPERATION RETURN CODE:  00

           * Statistics from all branches
           STATS FROM(ALL) ON(18,4,ZD) ON(28,6,PD) ON(22,6,PD)
ILU627I T0 DFSORT/VSE CALL 0001 FOR COPY FROM ALL     TO E35 EXIT COMPLETED
ILU628I T0 RECORD COUNT:  000000000000012
ILU607I T0 STATISTICS FOR (18,4,ZD)     :
ILU608I T0   MINIMUM:  +000000000000015, MAXIMUM:  +000000000000035
ILU609I T0   AVERAGE:  +000000000000024, TOTAL  :  +000000000000298
ILU607I T0 STATISTICS FOR (28,6,PD)     :
ILU608I T0   MINIMUM:  -000000000004278, MAXIMUM:  +000000000008276
ILU609I T0   AVERAGE:  +000000000004222, TOTAL  :  +000000000050665
ILU607I T0 STATISTICS FOR (22,6,PD)     :
ILU608I T0   MINIMUM:  +000000000012300, MAXIMUM:  +000000000042820
ILU609I T0   AVERAGE:  +000000000027469, TOTAL  :  +000000000329637
ILU602I T0 OPERATION RETURN CODE:  00

           * Books from VALD and WETH
           SORT FROM(BKIN,BKADD) TO(DAPUBS) USE
           USTART
ILU661I T0 01    SORT FIELDS=(106,4,A,1,75,A),FORMAT=CH
ILU661I T0 02    INCLUDE COND=(106,4,EQ,C'VALD',OR,106,4,EQ,C'WETH'),
ILU661I T0 03            FORMAT=CH
           UEND
ILU627I T0 DFSORT/VSE CALL 0002 FOR SORT FROM BKIN    TO DAPUBS   COMPLETED
ILU602I T0 OPERATION RETURN CODE:  00
```

Figure 58 (Part 1 of 4). Complete ICETOOL Output.

```
           * Separate output for California and Colorado branches
           SORT FROM(ALL) TO(CADASD,CATAPE) USE
           USTART
ILU661I T0 01   SORT FIELDS=(1,15,CH,A)
ILU661I T0 02   INCLUDE COND=(16,2,CH,EQ,C'CA')
           UEND
ILU627I T0 DFSORT/VSE CALL 0003 FOR SORT FROM ALL     TO CADASD   COMPLETED
ILU627I T0 DFSORT/VSE CALL 0004 FOR COPY FROM CADASD  TO CATAPE   COMPLETED
ILU602I T0 OPERATION RETURN CODE:  00

           SORT FROM(ALL) TO(CODASD,COTAPE) USE
           USTART
ILU661I T0 01      SORT FIELDS=(1,15,CH,A)
ILU661I T0 02      INCLUDE COND=(16,2,CH,EQ,C'CO')
           UEND
ILU627I T0 DFSORT/VSE CALL 0005 FOR SORT FROM ALL     TO CODASD   COMPLETED
ILU627I T0 DFSORT/VSE CALL 0006 FOR COPY FROM CODASD  TO COTAPE   COMPLETED
ILU602I T0 OPERATION RETURN CODE:  00

           * Copy multiple files
           COPY FROM(ALL) TO(D1,D2,D3)
ILU627I T0 DFSORT/VSE CALL 0007 FOR COPY FROM ALL     TO D1       COMPLETED
ILU627I T0 DFSORT/VSE CALL 0008 FOR COPY FROM D1      TO D2       COMPLETED
ILU627I T0 DFSORT/VSE CALL 0009 FOR COPY FROM D1      TO D3       COMPLETED
ILU602I T0 OPERATION RETURN CODE:  00

           COPY FROM(ALL) TO(D4) USE
           USTART
ILU661I T0 01      INCLUDE COND=(16,2,CH,EQ,C'CA')
           UEND
ILU627I T0 DFSORT/VSE CALL 0010 FOR COPY FROM ALL     TO D4       COMPLETED
ILU602I T0 OPERATION RETURN CODE:  00

           * California branches profit analysis
           RANGE FROM(CADASD) ON(28,6,PD) HIGHER(-1500) LOWER(+8000)
ILU627I T0 DFSORT/VSE CALL 0011 FOR COPY FROM CADASD  TO E35 EXIT COMPLETED
ILU628I T0 RECORD COUNT:  000000000000007
ILU631I T0 NUMBER OF VALUES IN RANGE FOR (28,6,PD)     :  000000000000003
ILU602I T0 OPERATION RETURN CODE:  00

           * Branches with less than 32 employees
           RANGE FROM(ALL) ON(18,4,ZD) LOWER(32)
ILU627I T0 DFSORT/VSE CALL 0012 FOR COPY FROM ALL     TO E35 EXIT COMPLETED
ILU628I T0 RECORD COUNT:  000000000000012
ILU631I T0 NUMBER OF VALUES IN RANGE FOR (18,4,ZD)     :  000000000000008
ILU602I T0 OPERATION RETURN CODE:  00

           * Print profit, employees, and city for each Colorado branch
           DISPLAY FROM(CODASD) LIST(013) ON(28,6,PD) ON(18,4,ZD) ON(1,15,CH)
(28,6,PD)          (18,4,ZD)          (1,15,CH)
+000000000005200   +000000000000020   Aspen
+000000000007351   +000000000000032   Boulder
+000000000006288   +000000000000033   Denver
-000000000002863   +000000000000022   Fort Collins
+000000000005027   +000000000000019   Vail
ILU627I T0 DFSORT/VSE CALL 0013 FOR COPY FROM CODASD  TO E35 EXIT COMPLETED
ILU603I T0 INFORMATION PRINTED ON SYS013
ILU628I T0 RECORD COUNT:  000000000000005
ILU602I T0 OPERATION RETURN CODE:  00
```

Figure 58 (Part 2 of 4). Complete ICETOOL Output.

```
      * Print a report for the Colorado branches
      DISPLAY FROM(CODASD) LIST(014) -
          DATE TITLE('Colorado Branches Report') PAGE -
          HEADER('City') HEADER('Profit') HEADER('Employees') -
          ON(1,15,CH) ON(28,6,PD) ON(18,4,ZD) BLANK -
          TOTAL('Total') AVERAGE('Average') MINIMUM('Lowest')
06/01/98        Colorado Branches Report        - 1 -

City                       Profit         Employees
---------------   ----------------   ----------------
Aspen                        5200                20
Boulder                      7351                32
Denver                       6288                33
Fort Collins                -2863                22
Vail                         5027                19

Total                       21003               126

Average                      4200                25

Lowest                      -2863                19
ILU627I T0 DFSORT/VSE CALL 0014 FOR COPY FROM CODASD  TO E35 EXIT COMPLETED
ILU603I T0 INFORMATION PRINTED ON SYS014
ILU628I T0 RECORD COUNT:  000000000000005
ILU602I T0 OPERATION RETURN CODE:  00

           DISPLAY FROM(DAPUBS) LIST(015) -
               TITLE('Books for individual publishers') PAGE -
               HEADER('Title of book') ON(1,35,CH) -
               HEADER('Price of book') ON(170,4,BI,C1,F'$') -
               BTITLE('Publisher:') BREAK(106,4,CH) -
               BAVERAGE('Average for this publisher') -
               BTOTAL('Total for this publisher') -
               AVERAGE('Average for all publishers') -
               TOTAL('Total for all publishers')
Books for individual publishers        - 1 -

Publisher:  VALD

Title of book                                  Price of book
-----------------------------------   ----------------------
EDITING SOFTWARE MANUALS                              $14.50
INTRODUCTION TO BIOLOGY                               $23.50
SHORT STORIES AND TALL TALES                          $15.20
STRATEGIC MARKETING                                   $23.50
VIDEO GAME DESIGN                                     $21.99

Average for this publisher                            $19.73

Total for this publisher                              $98.69
Books for individual publishers        - 2 -

Publisher:  WETH
```

Figure 58 (Part 3 of 4). Complete ICETOOL Output.

```
Title of book                                 Price of book
-----------------------------------   ----------------------
COMPUTERS: AN INTRODUCTION                           $18.99
EIGHTEENTH CENTURY EUROPE                            $17.90
SYSTEM PROGRAMMING                                   $31.94
THE INDUSTRIAL REVOLUTION                             $7.95

Average for this publisher                           $19.19

Total for this publisher                             $76.78
Books for individual publishers        - 3 -

Title of book                                 Price of book
-----------------------------------   ----------------------
Average for all publishers                           $19.49

Total for all publishers                            $175.47
ILU627I T0 DFSORT/VSE CALL 0015 FOR COPY FROM DAPUBS  TO E35 EXIT COMPLETED
ILU603I T0 INFORMATION PRINTED ON SYS015
ILU628I T0 RECORD COUNT:  000000000000009
ILU602I T0 OPERATION RETURN CODE:  00

           * Print the count of books in use from each publisher
           OCCUR FROM(BKIN) LIST(016) BLANK -
               TITLE('Books from Publishers') DATE(DMY.) -
               HEADER('Publisher') HEADER('Books in use') -
               ON(106,4,CH) ON(VALCNT)
Books from Publishers        01.06.98

Publisher      Books in use
---------   ---------------
COR                       7
FERN                      4
VALD                      5
WETH                      4
ILU627I T0 DFSORT/VSE CALL 0016 FOR SORT FROM BKIN    TO E35 EXIT COMPLETED
ILU603I T0 INFORMATION PRINTED ON SYS016
ILU628I T0 RECORD COUNT:  000000000000020
ILU638I T0 NUMBER OF RECORDS RESULTING FROM CRITERIA:  000000000000004
ILU602I T0 OPERATION RETURN CODE:  00

           * Separate output containing records for publishers
           * with more than 4 books in use
           SELECT FROM(BKIN) TO(BKOUT) ON(106,4,CH) HIGHER(4)
ILU656I T0 0004K EXTRA STORAGE REQUESTED, 0004K AVAILABLE
ILU627I T0 DFSORT/VSE CALL 0017 FOR SORT FROM BKIN    TO BKOUT    COMPLETED
ILU628I T0 RECORD COUNT:  000000000000020
ILU638I T0 NUMBER OF RECORDS RESULTING FROM CRITERIA:  000000000000012
ILU602I T0 OPERATION RETURN CODE:  00


ILU601I T0 ICETOOL UTILITY RUN ENDED - RETURN CODE:  00
1S55I  LAST RETURN CODE WAS 0000
EOJ PGMA     MAX.RETURN CODE=0000       DATE 06/01/98,CLOCK 09/11/54,DURATION 00/03/23
```

Figure 58 (Part 4 of 4). Complete ICETOOL Output.

Summary

This chapter covered several of the thirteen ICETOOL operators and their uses. This completes the *Getting Started with DFSORT/VSE* tutorials. For more information on the ICETOOL operators, please refer to *Application Programming Guide*.

The following appendixes contain valuable information on DFSORT/VSE that is also related to this publication.

Part 3. Appendixes

Appendix A. Using the DFSORT/VSE Sample Files

The DFSORT/VSE product tape includes a sample job, ILUDATA, which creates the files SORT.SAMPIN, SORT.SAMPADD, and SORT.BRANCH, to use with the examples in this book. Make sure these files are available at your site.

Many of the examples in this book use SORT.SAMPIN, SORT.SAMPADD, and SORT.BRANCH as the input files.

Appendix B. The Sample Bookstore Files

Assume that the sample file, SORT.SAMPIN, is used at a college bookstore to keep information about the books it sells. Each horizontal line represents a record, and each column represents a record field. For the sake of illustration, the file has only 20 records, each 173 bytes long. The file has also been arranged with headings and numbers to show the byte positions of each field. Neither of these appear in the actual file. The fields which are in binary format may not appear on your terminal. The methods used to arrange and present that file as you see it here are explained in the chapters detailing the DFSORT/VSE functions.

The first nine fields of each record contain character data and the last three fields contain binary data (binary data is shown in its character representation). Note that because binary data cannot contain decimal points, the prices are shown in cents rather than dollars.

Blanks in the course field indicate that the book is not required for any class.

The additional sample file, SORT.SAMPADD, has 22 records, each 173 bytes long, with the same fields as the sample file, SORT.SAMPIN.

For your quick reference, the table below shows the length and data format of each field.

Field	Length	Data Format
Title	75	CH
Author's Last Name	15	CH
Author's First Name	15	CH
Publisher	4	CH
Course Department	5	CH
Course Number	5	CH
Course Name	25	CH
Instructor's Last Name	15	CH
Instructor's Initials	2	CH
Number In Stock	4	BI
Number Sold Y-to-D	4	BI
Price	4	BI

Sample File - SORT.SAMPIN

Book Title	Author's Last Name	Author's First Name	Publisher	Course Department
1 75	76 90	91 105	106 109	110 114
COMPUTER LANGUAGES	MURRAY	ROBERT	FERN	COMP
LIVING WELL ON A SMALL BUDGET	DEWAN	FRANK	COR	
SUPPLYING THE DEMAND	MILLER	TOM	COR	BUSIN
VIDEO GAME DESIGN	RASMUSSEN	LORI	VALD	COMP
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	WILDE	KAREN	COR	ENGL
COMPUTERS: AN INTRODUCTION	DINSHAW	JOKHI	WETH	COMP
PICK'S POCKET DICTIONARY	GUSTLIN	CAROL	COR	
EDITING SOFTWARE MANUALS	OJALVO	VICTOR	VALD	ENGL
NUMBERING SYSTEMS	BAYLESS	WILLIAM	FERN	COMP
STRATEGIC MARKETING	YAEGER	MARK	VALD	BUSIN
THE INDUSTRIAL REVOLUTION	GROSS	DON	WETH	HIST
MODERN ANTHOLOGY OF WOMEN POETS	COWARD	PETER	COR	ENGL
INTRODUCTION TO PSYCHOLOGY	DUZET	LINDA	COR	PSYCH
THE COMPLETE PROOFREADER	GREEN	ANN	FERN	ENGL
SYSTEM PROGRAMMING	CAUDILLO	RAUL	WETH	COMP
SHORT STORIES AND TALL TALES	AVRIL	LILIANA	VALD	ENGL
INTRODUCTION TO BIOLOGY	WU	CHIEN	VALD	BIOL
ADVANCED TOPICS IN PSYCHOANALYSIS	OSTOICH	DIANNE	FERN	PSYCH
EIGHTEENTH CENTURY EUROPE	MUNGER	ALICE	WETH	HIST
CRISES OF THE MIDDLE AGES	BENDER	GREG	COR	HIST

Sample File - SORT.SAMPADD

Book Title	Author's Last Name	Author's First Name	Publisher	Course Department
1 75	76 90	91 105	106 109	110 114
ANOTHER ITALIAN DICTIONARY	UNDER	JOAN	COR	
COMPLETE SPANISH DICTIONARY	ROBERTS	ANGEL	VALD	
FRENCH TO ENGLISH DICTIONARY	JONES	JACK	FERN	
GUIDE TO COLLEGE LIFE	LAMB	CHARLENE	WETH	
GUNTHER'S GERMAN DICTIONARY	WILLIS	GUNTHER	WETH	
A SMALLER WORLD: MICROBES	BEESLY	GEORGE	FERN	BIOL
CELLS AND HOW THEY WORK	JETTS	PETER	VALD	BIOL
DNA: BLUEPRINT FOR YOU	HAVERS	ILSE	FERN	BIOL
THE ANIMAL KINGDOM	YOUNG	KEVIN	COR	BIOL
ANTICIPATING THE MARKET	ALLEN	CLYDE	WETH	BUSIN
KNOW YOUR CONSUMER	ZANE	JENNIFER	COR	BUSIN
QUEUE THEORY	FOX	THAD	FERN	BUSIN
THE ART OF TAKEOVERS	HUNT	ROBERT	FERN	BUSIN
ZEN BUSINESS	WILLIAMS	KATIE	VALD	BUSIN
DESIGNING APPLICATIONS	STEVENS	NOAH	COR	COMP
THE TOY STORE TEST	LITTLE	MARIE	COR	COMP
NOVEL IDEAS	PETERS	SETH	VALD	ENGL
CIVILIZATION SINCE ROME FELL	PIERCE	NICOLE	WETH	HIST
POLITICS AND HISTORY	TOMPSOM	KEN	FERN	HIST
REBIRTH FROM ITALY	FISH	JOHN	WETH	HIST
FREUD'S THEORIES	GOOLE	APRIL	VALD	PSYCH
MAP OF THE HUMAN BRAIN	WINTER	POLLY	COR	PSYCH

Course Number	Course Name	Instructor's Last Name	Instructor's Initials	Number In Stock	Number Sold Year-to-Date	Price
115 119	120 144	145 159	160 161	162 165	166 169	170 173
00032	INTRO TO COMPUTERS	CHATTERJEE	CL	5	29	2600
				14	1	9900
70251	MARKETING	MAXWELL	RF	0	32	1925
00205	VIDEO GAMES	NEUMANN	LB	10	10	2199
10856	MODERN POETRY	FRIEDMAN	KR	2	32	595
00032	INTRO TO COMPUTERS	CHATTERJEE	CL	20	26	1899
				46	38	295
10347	TECHNICAL EDITING	MADRID	MM	13	32	1450
00032	INTRO TO COMPUTERS	CHATTERJEE	AN	6	27	360
70124	ADVANCED MARKETING	LORCH	MH	3	35	2350
50420	WORLD HISTORY	GOODGOLD	ST	15	9	795
10856	MODERN POETRY	FRIEDMAN	KR	1	26	450
30016	PSYCHOLOGY I	ZABOSKI	RL	26	15	2200
10347	TECHNICAL EDITING	MADRID	MM	7	19	625
00103	DATA MANAGEMENT	SMITH	DC	4	23	3195
10054	FICTION WRITING	BUCK	GR	10	9	1520
80521	BIOLOGY I	GREENBERG	HC	6	11	2350
30975	PSYCHOANALYSIS	NAKATSU	FL	1	12	2600
50632	EUROPEAN HISTORY	BISCARDI	HR	23	21	1790
50521	WORLD HISTORY	WILLERTON	DW	14	17	1200

Course Number	Course Name	Instructor's Last Name	Instructor's Initials	Number In Stock	Number Sold Year-to-Date	Price
115 119	120 144	145 159	160 161	162 165	166 169	170 173
				26	6	925
				8	4	650
				7	17	1100
				20	1	2000
				3	16	1088
80522	BIOLOGY II	HAROLD	LM	9	20	1995
80523	INTRO TO GENETICS	ABRAHAM	NG	8	46	2495
80523	INTRO TO GENETICS	ABRAHAM	NG	15	21	2195
80522	BIOLOGY II	HAROLD	LM	2	30	3000
70124	ADVANCED MARKETING	LORCH	MH	20	25	2000
70251	MARKETING	MAXWELL	RF	16	3	4500
70255	BUSINESS THEORY	SCHOFFE	KN	2	20	1500
70255	BUSINESS THEORY	SCHOFFE	KN	17	15	615
70255	BUSINESS THEORY	SCHOFFE	KN	12	12	1200
00103	DATA MANAGEMENT	SMITH	DC	7	15	1435
00205	VIDEO GAMES	NEUMANN	LB	15	12	2600
10054	FICTION WRITING	BUCK	GR	11	25	2450
50420	WORLD HISTORY	GOODGOLD	ST	6	15	1350
50521	WORLD HISTORY	WILLERTON	DW	3	17	995
50632	EUROPEAN HISTORY	BISCARDI	HR	10	20	2560
30975	PSYCHOANALYSIS	NAKATSU	FL	12	15	1250
30016	PSYCHOLOGY I	ZABOSKI	RL	6	9	895

Appendix C. Processing Order of Control Statements

The flowchart below shows the order in which control statements are processed. (SUM is processed at the same time as SORT or MERGE. It is not used with a copy application.)

Although you can write the statements in any order, DFSORT/VSE always processes the statements in the order shown below.

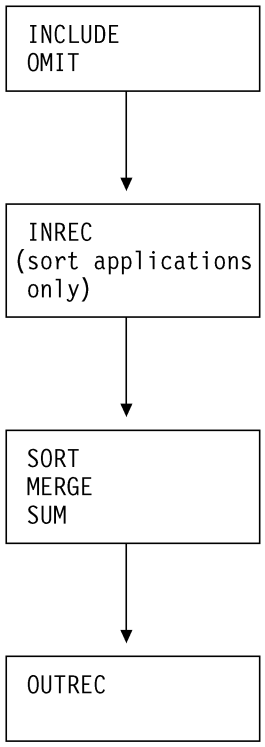


Figure 59. Processing Order of Control Statements

Summary of Changes for Previous Releases of DFSORT/VSE

Second Edition, February 1997

Programming Support for Release 3

National Language Support

Cultural Sort and Merge: DFSORT/VSE will allow the selection of an active locale at installation or run time and will produce sorted or merged records for output according to the collating rules defined in the active locale. This provides sorting and merging for single- or multi-byte character data based on defined collating rules which retain the cultural and local characteristics of a language.

Cultural Include and Omit: DFSORT/VSE will allow the selection of an active locale at installation or run time and will include or omit records for output according to the collating rules defined in the active locale. This provides inclusion or omission for single- or multi-byte character data based on defined collating rules which retain the cultural and local characteristics of a language.

DFSORT/VSE Messages Translation: DFSORT/VSE provides the ability for easy translation of messages into different languages. Flexible positioning of variables within a message is supported.

ICETOOL Reports: ICETOOL's DISPLAY operator allows date, time, and numeric values in reports to be formatted in many of the notations used throughout the world. (Note that this support was first made available in DFSORT/VSE Version 3 Release 2.)

Productivity

Year 2000 Features: DFSORT/VSE's Year 2000 support will help you prepare for the turn of the century by correctly processing your two-digit year data. New Y2C, Y2Z, Y2P, and Y2D formats, in conjunction with a new Y2PAST installation and run-time option, allow you to handle two-digit year data in the following ways:

- Set the appropriate century window for your applications (for example, 1915-2014 or 1950-2049).
- Order two-digit character, zoned decimal, packed decimal or decimal year data according to the century window using SORT or MERGE (for example, order 96 representing 1996 before 00 representing 2000 in ascending sequence, or order 00 before 96 in descending sequence).
- Transform two-digit character, zoned decimal, packed decimal or decimal year data to four-digit character (or zoned decimal) year data according to the century window using OUTREC (for example, transform 96 to 1996 and 00 to 2000).

A new PD0 format allows you to order parts of packed decimal fields, such as month and day in date fields, using SORT or MERGE.

New PZ, PSI and ZSI formats allow you to transform packed decimal and zoned decimal fields, such as month and day in date fields, to character fields using OUTREC.

New character and hexadecimal string separators allow you to insert literals in reformatted fields using OUTREC (for example, '/' in mm/dd/yyyy fields).

File Management Systems: New installation option FMS allows you to specify that DFSORT/VSE should attempt to take advantage of benefits provided by the File Management System installed at your site, such as:

- Dynamic logical and physical device assignment
- Dynamic primary and secondary extent allocation

INCLUDE and OMIT Enhancements: Do sophisticated filtering with new INCLUDE and OMIT features:

- Use the substring search capability to allow inclusion of records when:
  - A specified character or hexadecimal constant is found anywhere within a specified input field (that is, a constant is a substring within a field) or
  - A specified input value is found anywhere within a specified character or hexadecimal constant (that is, a field is a substring within a constant).
- Use bit level logic to allow inclusion of records based on:
  - Bit comparison tests of a binary field against a bit string constant. The bit string constant allows you to specify which bits of the input field must be on, which must be off and which can be ignored.
  - Bit operator tests of a binary field against a bit or hexadecimal mask. The mask allows you to test many different possible bit combinations with a single operation, similar to what you can

do using the Test Under Mask (TM) machine instruction.

- Have unlimited levels of parentheses within logical expressions which allows you to create more complex conditions for inclusion of records.

SORT or MERGE Bit Control Fields: Use DFSORT/VSE's new bit control fields in conjunction with byte control fields to give you additional ways to sort or merge your data.

Input/Output Improvements: You can take advantage of the following input/output handling improvements:

- Copy up to nine input files to one output file.
- Process records and output them to a printer or punch device for sort, merge, or copy applications.
- Process input and output files on the IBM 3590 Tape Subsystems for sort, merge, and copy applications.

ICETOOL: ICETOOL is now more versatile as a result of enhancements to the existing operators. The improvements include:

- Allowing up to nine input files to be processed with the COPY, COUNT, DISPLAY, RANGE, STATS, and VERIFY operators.
- Allowing the active locale to be specified for the COPY, COUNT, and SORT operators, overriding the installation default for the active locale. Locale processing provides a way to SORT or COPY and INCLUDE or OMIT records according to the language and country rules defined in the active locale.

Performance: Performance enhancements for DFSORT/VSE Version 3 Release 3 include the following:

- Dataspace sorting, introduced in DFSORT/VSE Version 3 Release 1 for fixed-length record sort applications, is now available for variable-length record sort applications.
- Improved Extended Count-Key-Data (ECKD) disk device support for work files.
- Improved data processing methods for:
  - Incore and non-incore dataspace sorting
  - Incore getvis sorting
  - Copy and merge applications
  - Input and output tape file processing for sort, merge, and copy applications.

Additional Enhancements

EQUALS/NOEQUALS: Preserve the original sequence of records that collate identically from input to output for merge applications using the EQUALS option.

Virtual Storage Constraint Relief (VSCR): Reduction in the number of DFSORT/VSE modules that are loaded into the 24-bit Shared Virtual Area (SVA) and partition program area.

HDR2: HDR2 support for tape output files.

First Edition, October 1995

Programming Support for Release 2

DFSORT/VSE Release 2 continues the strategy of providing performance improvements and productivity features. These improvements and features are described in more detail in the subsections that follow.

Performance: With DFSORT/VSE Release 2, performance enhancements for dataspace sorting of fixed-length records and for getvis sorting of fixed-length and variable-length records include:

- Dynamic control of storage for dataspace sorting and getvis sorting
- Improved data processing methods for incore getvis sorting
- Improved data processing methods for non-incore getvis and dataspace sorting
- Reduced amount of work space required for SAM ESDS work files

Productivity: With DFSORT/VSE Release 2, your productivity is improved because you can:

- Create reports and analyze data using the new ICETOOL utility
- Select subsets of data using the STOPAFT and SKIPREC options
- Tailor DFSORT/VSE to suit your needs by using the new and modified ILUINST installation defaults and run-time options
- Access both introductory and reference information more quickly using a new DFSORT/VSE publication and a new DFSORT/VSE CD-ROM

ICETOOL: With ICETOOL, a versatile new DFSORT/VSE utility, you can do reporting and analysis of data at your site. ICETOOL allows you to perform multiple operations on one or more files in a single job step. This batch front-end utility uses the capabilities of DFSORT/VSE to perform the operations you request.

ICETOOL has thirteen operators: COPY, COUNT, DEFAULTS, DEFINE, DISPLAY, MODE, OCCUR, RANGE, SELECT, SORT, STATS, UNIQUE, and VERIFY. By using one operator or a combination of these operators, you can easily create applications that perform a variety of tasks including:

- Sorting input files to one or more output files
- Creating multiple copies of input files
- Creating output files containing subsets of input files based on various criteria
- Creating detailed reports allowing control of title, date, time, page numbers, headings, lines per page, field formats, and total, maximum, minimum, and average values
- Creating reports showing unique values for selected character and numeric fields and the number of times each occurs
- Creating reports or output files for records with: duplicate values, nonduplicate values, or values that occur n times, less than n times, or more than n times
- Creating a wide variety of reports using the preferred date, time, and numeric notations of individual countries
- Creating a report showing the DFSORT/VSE installation defaults in use
- Printing messages that give statistical information for selected numeric fields such as minimum, maximum, average, total, count of values, and count of unique values
- Printing messages that identify invalid decimal values
- Using three different modes, (stop, continue, and scan) to control error checking and actions after error detection for groups of operators

ICETOOL can be called directly or from a program. It also produces messages and return codes describing the results of each operation and any errors detected. Although you generally do not need to look at the DFSORT/VSE messages produced as a result of an ICETOOL run, they are available if you need them.

SKIPREC and STOPAFT Options: With DFSORT/VSE Release 2, there are two new options that allow you to select subsets of data for sorting or copying. These options are:

- The SKIPREC option which enables you to specify the number of records you want to skip before starting to sort or copy the input files.
- The STOPAFT option which enables you to specify the maximum number of records you want accepted for sorting or copying.

Installation Defaults and Run-time Options: Several ILUINST installation defaults have been added or changed providing you with more flexibility in using installation defaults and run-time options. These defaults include:

- GVSRLOW and GVSRANY which enable you to specify the reserved partition GETVIS area for user application requirements.
- WRKSEC which enables you to allow or suppress the dynamic secondary allocation for SAM ESDS work files.
- STORAGE which may be specified in megabytes.

To give you even more flexibility on a daily basis, several run-time options have been added or changed. These run-time options include:

- WRKSEC and NOWRKSEC which enable you to override the installation defaults.
- GVSRLOW and GVSRANY which enable you to override the installation defaults.
- STORAGE which may be specified in megabytes.
- CKPT which enables you to restart DFSORT/VSE when the DFSORT/VSE modules are placed in SVA.

Additional Enhancements: Additional changes that you will find helpful are:

- An extended DFSORT/VSE parameter list feature which enables you to supply the *end of parameter* list indicator.
- Tagging of the error position in an invalid parameter list control statement image.
- The END control statement which enables you to discontinue accepting control statements.
- Improved work file processing through internal enhancements to the SAM ESDS work files secondary allocation algorithm which allows you to use multivolume SAM ESDS work files.
- Improved system and application availability with additional program modules placed above 16 MB virtual.

With DFSORT/VSE, Release 2, a new book is included in the DFSORT/VSE library, *Getting Started with DFSORT/VSE*, SC26-7101. This book gives you all of the information and instructions you need to start using the features of DFSORT/VSE.

Also new with this release is the *DFSORT/VSE Online Product Library*, SK2T-8730. This CD-ROM contains all of the books in the DFSORT/VSE library except for the *DFSORT/VSE Reference Summary* and the *DFSORT/VSE Licensed Program Specifications*.

Index

A
allowable comparisons 25
ascending order
  example 1
  sorting 9
ASSGN statement 14

B
BI format 2
binary data 2
binary zeros 34
blanks 34

C
calling DFSORT/VSE
  COBOL program 45
CH format 2
character data 2
character strings 35
COBOL
  calling DFSORT/VSE 45
  MERGE statement 45
  passing DFSORT/VSE statements 45
  sample program 46, 47
  SORT statement 45
COBOL/VSE
  FASTSRT compile-time option 55
  passing DFSORT statements 49
comparison
  allowable 25
  field-to-constant 21, 24, 25
  field-to-field 21, 24, 25
  operators 22
constant
  format for writing 26
  inserting with the OUTREC Statement 35
continuing a statement 13
control fields
  combining 12
  deleting duplicate records 29
  equal 5
  general information 3
  multiple 11
  overlapping 11
  reordering 32
  summing 27
control statements
  ordering 22
  processing order flowchart 103
  writing 3
COPY operator (ICETOOL) 59, 73
COPY statement 18
copying records 18
COUNT operator (ICETOOL) 59
creating a DFSORT/VSE job 3

D
data format 2
data space 53
dataspace sorting 53
defaults
  order of equal records 5
  overriding 51
    using OPTION statement 51
DEFAULTS operator (ICETOOL) 59
DEFINE operator (ICETOOL) 59, 62
deleting fields 31
descending order
  example 1
  sorting 11
devices for efficient use 53
DFSORT/VSE home page x
DISPLAY operator (ICETOOL) 59, 75, 77, 78, 81
DLBL SORTINn statement 17
DLBL SORTWKn statement 17
DLBL statement 14
  SORTIN 14
  SORTOUT 14
  SORTWKn 14
duplicate fields
  OCCUR operator (ICETOOL) 59, 84
  SELECT operator (ICETOOL) 59, 85

E
EBCDIC data format code 2
edit masks
  ICETOOL 78
equal control fields 5
excluding records 24
EXEC statement 14
  merging 17
  sorting 13
EXTENT statement 14

F
FASTSRT
  COBOL 49
  compile-time option 55
field-to-constant comparison 21, 24, 25

field-to-field comparison 21, 24, 25
fields
  deleting unnecessary fields 31
  numeric and character, printing 75
  printing statistics for numeric 63
FIELDS, specifying on SUM statement 27
FIELDS=NONE, specifying on SUM statement 29
file
  copying
    definition 2
    increasing speed 21
    writing the COPY statement 18
  merging 15
    definition 2
    increasing speed 21
  sorting
    ascending order 9
    by multiple control fields 11
    definition 1
    descending order 11
    increasing speed 21
    writing the SORT statement 9
  tailoring with INCLUDE and OMIT 21
format
  BI 2
  CH 2
  data 2
  PD 2
  ZD 2
FORMAT= parameter 13
formats for writing constants
  character strings 26, 35
  decimal numbers 26
  hexadecimal strings 26, 36

G
GETVIS area 53
getvis sorting 53

H
handling overflow 29
hexadecimal strings 36

I
ICETOOL
  complete sample job 87
  continuing an operator statement 66
  COPY operator 59, 73
  COUNT operator 59
  creating a job 61
  DEFAULTS operator 59
  DEFINE operator 59, 62
  definition 59
ICETOOL *(continued)*
  DISPLAY operator 59, 75, 77, 78, 81
  examples
    complete ICETOOL job 87
    continuing an operator statement 66
    counting values in a range 74
    creating a job 61
    creating identical sorted files 67
    creating multiple unsorted files 73
    ICETOOL output 87, 89
    printing numeric and character fields 75
    printing statistics for numeric fields 63
    printing statistics for record lengths 66
    writing required JCL 61
  ICETOOL output, sample 89
  JCL statements 61
  job
    elements 61
    sample 87
  MODE operator 59
  OCCUR operator 59, 84
  operator summary 59
  printing
    fields 75
    record length 76
    relative record number 76
    statistics for numeric fields 63
  printing field value occurrences 84
  printing sectioned reports 81
  printing tailored reports 77
  RANGE operator 59, 74
  requirements
    input files 60
    JCL 61
  SELECT operator 59, 85
  selecting records by field occurrences 85
  SORT operator 60, 67
  statements
    blank 62
    comment 62
    operator, continuing 66
  STATS operator 60, 63, 66
  UNIQUE operator 60
  using formatting items 78
  VERIFY operator 60
ILUDATA (sample job) 5, 97
improving performance
  INCLUDE statement 54
  INREC statement 54
  JCL (job control language) 55
  OMIT statement 54
  SKIPREC option 54
  STOPAFT option 54
  SUM statement 54
INCLUDE statement
  allowable comparisons 25

INCLUDE statement *(continued)*
  improving performance 54
  using with OUTREC 33
  writing 21
increasing speed
  copying 21
  merging 21
  sorting 21
input file 2, 3, 5, 14, 17
  ICETOOL 60
  selecting 13
  SORT.BRANCH 60
  SORT.SAMPADD 60
  SORT.SAMPIN 60
  tailoring with INCLUDE and OMIT 21
INREC statement
  considerations when used with other statements 42
  deleting fields 31
  differences between INREC and OUTREC processing 31
  improving performance 54
  preventing overflow 43
  processing order flowchart 103
  reformatting records 31, 42
  writing 42
inserting constants 35
inserting separators 31
inserting with OUTREC
  blanks 34
  blanks or zeros 31
  character strings 35
  hexadecimal strings 36
  zeros 34
installation defaults, overriding 51
Internet x

J
JCL (job control language)
  calling DFSORT/VSE from a program 45, 47
  executing a copy 19
  executing a merge 17
  executing a sort 13
  general information 13
  ICETOOL 61
  improving performance 55
  statements 14
JOB statement 14

M
margins 34
MERGE statement
  specifying with COPY 18
  writing 16
MERGE statement, COBOL 45
merging records 15
MODE operator (ICETOOL) 59
multiple control fields 11

N
nonduplicate fields
  OCCUR operator (ICETOOL) 59, 84
  SELECT operator (ICETOOL) 59, 85
notices vii

O
OCCUR operator (ICETOOL) 59, 84
occurrences
  OCCUR operator (ICETOOL) 59, 84
  SELECT operator (ICETOOL) 59, 85
OMIT statement
  allowable comparisons 25
  improving performance 54
  writing 24
OPTION statement 51
  writing 51
order of control statements 22
ordering
  ascending order 1
  control statements 22
  descending order 1
  DFSORT/VSE defaults 5
output file 10, 14
output records, reformatting 31
OUTREC statement
  deleting fields 31
  differences between INREC and OUTREC processing 31
  inserting separators 31
  processing order flowchart 103
  reformatting records 31
  reordering fields 33
  using edit and lookup features 38
  using to
    insert binary zeros 34
    insert blanks 34
    insert character strings 35
    insert hexadecimal strings 36
  writing 31
overflow
  explanation 29
  preventing 43
overriding installation defaults 51

P
packed decimal data 2

padding
  description 24
  using INCLUDE and OMIT 24
padding summary
  fields 43
passing DFSORT statements
  from a COBOL/VSE program 49
passing DFSORT/VSE statements
  from a COBOL program 45
PD format 2
printing reports
  DISPLAY operator (ICETOOL) 78, 81
printing reports with DISPLAY operator (ICETOOL) 59, 75, 77
processing order
  differences between INREC and OUTREC 31
  flowchart 103
  special considerations for INREC 42

R

RANGE operator (ICETOOL) 59, 74
record length, changing 32
RECORD statement
  writing 9
records
  copying
    definition 2
    writing the COPY statement 18
  merging 15
    definition 2
  reordering and reformatting with OUTREC 32
  selecting by occurrences 85
  sorting
    ascending order 9
    by multiple control fields 11
    definition 1
    descending order 11
    writing the SORT statement 9
reformatting records 31
reordering fields 31, 32
reports
  See also printing reports with DISPLAY operator (ICETOOL)
  DISPLAY operator (ICETOOL) 78
  division items (ICETOOL) 80
  edit masks 78
  leading, floating and trailing characters (ICETOOL) 80
  sectioned (ICETOOL) 81

S

sample file 3, 5, 99, 100
  input file
    SORT.BRANCH 5, 97
    SORT.SAMPADD 3, 5, 100
sample file *(continued)*
  input file *(continued)*
    SORT.SAMPIN 3, 5, 97, 100
  output file
    SORT.SAMPOUT 97
  SORT.BRANCH 5, 97
  SORT.SAMPADD 5, 100
  SORT.SAMPIN 5, 97, 100
  SORT.SAMPOUT 97
sample job
  ILUDATA 5, 97
SELECT operator (ICETOOL) 59, 85
selecting records 21
SKIPREC option 54
SORT operator (ICETOOL) 60, 67
SORT statement
  specifying COPY 18
  using with OUTREC 33
  writing 9
SORT statement, COBOL 45
SORT-CONTROL special register 45
SORT.BRANCH 5, 60, 97
SORT.SAMPADD 3, 5, 60, 100
SORT.SAMPIN 3, 5, 60, 97, 100
SORT.SAMPOUT 97
sorting records
  ascending order 9
  by multiple control fields 11
  descending order 11
  FASTSRT option 49
  increasing speed 21
  writing the SORT statement 9
specifying COPY
  MERGE statement 18
  SORT statement 18, 19
statistics, printing for numeric fields 63
STATS operator (ICETOOL) 60, 63
STOPAFT option 54
SUM statement
  FIELDS 27
  FIELDS=NONE 29
  improving performance 54
  writing 27
summary fields 27
SYSIPT 9, 14
SYSLST 14

T

tailoring a file
  using INCLUDE and OMIT 21
truncation 24

U
UNIQUE operator (ICETOOL) 60
use high–speed disks 53
using the DFSORT/VSE sample files 97

V
VERIFY operator (ICETOOL) 60
virtual storage, allocating 53

W
work storage file 14
World Wide Web x

Z
ZD format 2
zeros 34
zoned decimal data 2

Communicating Your Comments to IBM

DFSORT/VSE
Getting Started with DFSORT/VSE
Version 3 Release 4

Publication No. SC26-7101-02

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
  - United States: 1-800-426-6209
  - Other countries: (+1)+408+256-7896
- If you prefer to send comments electronically, use this network ID:
  - IBMLink from U.S. and IBM Network: STARPUBS at SJEVM5
  - IBMLink from Canada: STARPUBS at TORIBM
  - IBM Mail Exchange: USIB3VVD at IBMMAIL
  - Internet: starpubs@vnet.ibm.com

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

DFSORT/VSE
Getting Started with DFSORT/VSE
Version 3 Release 4

Publication No. SC26-7101-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
SC26-7101-02

IBM®

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
RCF Processing Department
M86/050
5600 Cottle Road
SAN JOSE, CA 95193-0001

Fold and Tape | **Please do not staple** | Fold and Tape

SC26-7101-02

Cut or Fold
Along Line

IBM Corporation **Printer's Collating List**

Direct Questions to: Henry Caudillo (408) 256-1466

Title: DFSORT/VSE Getting Started with DFSORT/VSE V3R4

Order/Top Number: SC26-7101-02 Page Count: 130 + covers Sheet 1 of 3
Publication Part Number: N/A

item	Type	Folio
	Front Cover	
	Blank (reverse of cover)	
1	Title Page	
2	Ed. Notice	
3	TOC	iii
4	TOC	iv
5	TOC	v
6	TOC	vi
7	Notices	vii
8	Notices	viii
9	Preface	ix
10	Preface	x
11	Preface	xi
12	Preface	xii
13	Preface	xiii
14	Preface	xiv
15	Text	1
16	Text	2
17	Text	3
18	Text	4
19	Text	5
20	Text	6
21	Text	7
22	Text	8

item	Type	Folio
23	Text	9
24	Text	10
25	Text	11
26	Text	12
27	Text	13
28	Text	14
29	Text	15
30	Text	16
31	Text	17
32	Text	18
33	Text	19
34	Text	20
35	Text	21
36	Text	22
37	Text	23
38	Text	24
39	Text	25
40	Text	26
41	Text	27
42	Text	28
43	Text	29
44	Text	30
45	Text	31
46	Text	32

IBM Corporation **Printer's Collating List**

Direct Questions to: Henry Caudillo (408) 256-1466

Title: DFSORT/VSE Getting Started with DFSORT/VSE V3R4

Order/Top Number: SC26-7101-02 Page Count: 130 + covers Sheet 2 of 3
Publication Part Number: N/A

item	Type	Folio
47	Text	33
48	Text	34
49	Text	35
50	Text	36
51	Text	37
52	Text	38
53	Text	39
54	Text	40
55	Text	41
56	Text	42
57	Text	43
58	Text	44
59	Text	45
60	Text	46
61	Text	47
62	Text	48
63	Text	49
64	Text	50
65	Text	51
66	Text	52
67	Text	53
68	Text	54
69	Text	55
70	Text	56
71	Text	57
72	Text	58
73	Text	59
74	Text	60
75	Text	61
76	Text	62
77	Text	63
78	Text	64
79	Text	65
80	Text	66
81	Text	67
82	Text	68
83	Text	69
84	Text	70
85	Text	71
86	Text	72
87	Text	73
88	Text	74
89	Text	75
90	Text	76
91	Text	77
92	Text	78
93	Text	79
94	Text	80

IBM Corporation **Printer's Collating List**

Direct Questions to: Henry Caudillo (408) 256-1466

Title: DFSORT/VSE Getting Started with DFSORT/VSE V3R4

Order/Top Number: SC26-7101-02 Page Count: 130 + covers Sheet 3 of 3
Publication Part Number: N/A

item	Type	Folio
95	Text	81
96	Text	82
97	Text	83
98	Text	84
99	Text	85
100	Text	86
101	Text	87
102	Text	88
103	Text	89
104	Text	90
105	Text	91
106	Text	92
107	Text	93
108	Text	94
109	Text	95
110	Text	96
111	Text	97
112	Text	98
113	Text	99
114	Text	100
115	Text	101
116	Text	102
117	Text	103
118	Text	104
119	Text	105
120	Text	106
121	Text	107
122	Text	108
123	Text	109
124	Text	110
125	Text	111
126	Text	112
127	Text	113
128	Blank	
129	RCF	
130	RCF Mailer	
	Blank (reverse of cover)	
	Back Cover	

SPECIAL INSTRUCTIONS

1. This book prints with spine text.

Program Number: 5746-SM3

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC26-7101-02