

1 INTRODUCTION: FILES AND FILE STRUCTURES

1.1 Primary and Secondary Storage 2

1.2 No Free Lunch 3

1.3 Files 4

1.4 File Structures versus Data Structures 5

1.5 A Conceptual Toolkit 6

Summary 7

Key Terms 8

2 FUNDAMENTAL FILE PROCESSING OPERATIONS

2.1 Physical Files and Logical Files 10

2.2 Opening Files and Creating Files 11

2.3 Closing Files 15

2.4 Reading and Writing 15

2.5 Detecting End of File 19

2.6 Seeking 19

2.6.1 Seeking in C 20

2.6.2 Seeking in Pascal 22

CONTENTS

2.7 Unexpected Characters in Files	23
Summary	24
Key Terms	25
Exercises	27
Further Readings	28
The append Program in Pascal	29

**3****SECONDARY STORAGE DEVICES AND SYSTEM SOFTWARE: PERFORMANCE CONSIDERATIONS**

3.1 Disks	33
3.1.1 The organization of disks	33
3.1.2 Estimating capacities and space needs	36
3.1.3 Sector organization	37
3.1.4 Block organization	42
3.1.5 Nondata overhead	43
3.1.6 The cost of a disk access	45
3.2 Magnetic Tape	48
3.2.1 Organization of data on tapes	49
3.2.2 Estimating tape length requirements	50
3.2.3 Estimating data transmission times	52
3.2.4 Tape applications	53
3.3 Other Kinds of Storage	54
3.4 Storage as a Hierarchy	56
3.5 A Journey of a Byte	56
3.5.1 The file manager	58
3.5.2 The I/O buffer	60
3.5.3 The byte leaves RAM—the I/O processor	61
3.5.4 The byte is put on the disk—the disk controller	63
3.6 Buffer Management	64
Summary	68
Key Terms	69
Exercises	73
Further Readings	76

**4****FUNDAMENTAL FILE STRUCTURE CONCEPTS**

- 4.1 A Stream File 80**
- 4.2 Field Structures 82**
 - 4.2.1 Method 1: Fixing the length of fields 82
 - 4.2.2 Method 2: Beginning each field with a length indicator 83
 - 4.2.3 Method 3: Separating the fields with delimiters 84
- 4.3 Reading a Stream of Fields 84**
- 4.4 Record Structures 86**
 - 4.4.1 Method 1: Making records a predictable length 87
 - 4.4.2 Method 2: Beginning each record with a length indicator 88
 - 4.4.3 Method 3: Using a second file to keep track of addresses 88
 - 4.4.4 Method 4: Placing a delimiter at the end of each record 89
- 4.5 A Record Structure That Uses a Length Indicator 89**
- 4.6 Mixing Numbers and Characters—Use of a Hex Dump 92**
- 4.7 Reading the Variable Length Records from the File 95**
- 4.8 Retrieving Records by Key: Canonical Forms for Keys 95**
- 4.9 A Sequential Search 98**
- 4.10 Evaluating Performance of Sequential Search 99**
- 4.11 Improving Sequential Search Performance: Record Blocking 100**
- 4.12 Direct Access 101**
- 4.13 Choosing a Record Structure and Record Length 103**
- 4.14 Header Records 107**
- 4.15 File Access and File Organization 107**
- Summary 109**
- Key Terms 111**
- Exercises 112**
- Further Readings 117**
- C Programs 118**
- Pascal Programs 131**

5**FILE MAINTENANCE AND RECORD DELETION**

- 5.1** File Maintenance 148
- 5.2** Storage Compaction 149
- 5.3** Overview of Fixed Length Record Deletion 151
- 5.4** Implementing Fixed Length Record Deletion 154
- 5.5** Variable Length Record Deletion 157
- 5.6** Storage Fragmentation 160
- 5.7** Placement Strategies 163
- Summary 165
- Key Terms 168
- Exercises 169
- Further Readings 171

6**FINDING THINGS QUICKLY IN A FILE:
AN INTRODUCTION TO SORTING AND
BINARY SEARCHING**

- 6.1** Finding Things in the Files Already Developed 175
- 6.2** Search by Guessing—Binary Search 175
- 6.3** Sorting a Disk File in RAM 178
- 6.4** Algorithm for the RAM Sort 182
- 6.5** The Sort Function: *shell_sort()* 183
- 6.6** The Limitations of Binary Searching and in-RAM Sorting 185
- 6.7** Keysorting 187
 - 6.7.1 Description of the method 187
 - 6.7.2 Limitations of the keysort method 189
 - 6.7.3 Another solution: Why bother to write the file back? 190
- 6.8** Pinned Records 192
- Summary 193
- Key Terms 195
- Exercises 196
- Further Readings 197
- C Programs 199
- Pascal Programs 203

7**INDEXING**

- 7.1** What is an Index? 210
- 7.2** A Simple Index with an Entry Sequenced File 211
- 7.3** Basic Operations on an Indexed, Entry Sequenced File 214
- 7.4** Indexes That Are Too Large to Hold in Memory 218
- 7.5** Indexing to Provide Access by Multiple Keys 219
- 7.6** Retrieval Using Combinations of Secondary Keys 224
- 7.7** Improving the Secondary Index Structure—Inverted Lists 226
 - 7.7.1 A first attempt at a solution 227
 - 7.7.2 A better solution: Linking the list of references 228
- 7.8** Selective Indexes 232
- 7.9** Binding 233
- Summary 234
- Key Terms 236
- Exercises 237
- Further Readings 240

8**COSEQUENTIAL PROCESSING AND SORTING
LARGE FILES**

- 8.1** A Model for Implementing Cosequential Processes 243
 - 8.1.1 Matching names in two lists 243
 - 8.1.2 Merging two lists 248
 - 8.1.3 Summary of the cosequential processing model 250
- 8.2** Application of the Model to a General Ledger Program 252
 - 8.2.1 The problem 252
 - 8.2.2 Application of the model to the ledger program 255
- 8.3** Extension of the Model to Include Multiway Merging 260
- 8.4** Merging as a Way of Sorting Large Files on Disk 263
 - 8.4.1 Multiple-step merge patterns 267
 - 8.4.2 Increasing run lengths by using replacement selection 270
 - 8.4.3 Average run length for replacement selection 273
 - 8.4.4 Cost of using replacement selection 274

- 8.4.5 Disk configurations 277
- 8.4.6 Summary concerning disk sorting 277
- 8.5 Sorting Files on Tape 278**
 - 8.5.1 The balanced merge 279
 - 8.5.2 Multiphase merges 281
- 8.6 Sort/Merge Packages 283**
 - Summary 284
 - Key Terms 287
 - Exercises 289
 - Further Readings 291


9
**B-TREES AND OTHER TREE-STRUCTURED
FILE ORGANIZATIONS**

- 9.1 Introduction—The Invention of the B-tree 294**
- 9.2 Statement of the Problem 296**
- 9.3 Binary Search Trees as a Solution 297**
- 9.4 AVL Trees 300**
- 9.5 Paged Binary Trees 304**
- 9.6 The Problem with the Top-down Construction
of Paged Trees 306**
- 9.7 B-trees: Working Up from the Bottom 308**
- 9.8 Splitting and Promoting 308**
- 9.9 Algorithms for B-tree Searching and Insertion 313**
- 9.10 B-tree Nomenclature 323**
- 9.11 Formal Definition of B-tree Properties 324**
- 9.12 Worst-case Search Depth 325**
- 9.13 Deletion, Redistribution, and Concatenation 327**
 - 9.13.1 Redistribution 331
- 9.14 Redistribution During Insertion: A Way to Improve
Storage Utilization 332**
- 9.15 B* Trees 333**
- 9.16 Buffering of Pages: Virtual B-trees 334**
 - 9.16.1 LRU replacement 336
 - 9.16.2 Replacement based on page height 337

- 9.16.3 Importance of virtual B-trees 338
- 9.17 Placement of Information Associated with the Key 338**
- 9.18 Variable Length Records and Keys 340**
- Summary 341
- Key Terms 343
- Exercises 345
- Further Readings 348
- C Program to Insert Keys into a B-tree 350
- Pascal Program to Insert Keys into a B-tree 357


10
THE B⁺ TREE FAMILY AND INDEXED SEQUENTIAL FILE ACCESS

- 10.1 Indexed Sequential Access 366**
- 10.2 Maintaining a Sequence Set 367**
 - 10.2.1 The use of blocks 367
 - 10.2.2 Choice of block size 370
- 10.3 Adding a Simple Index to the Sequence Set 372**
- 10.4 The Content of the Index: Separators Instead of Keys 374**
- 10.5 The Simple Prefix B⁺ Tree 376**
- 10.6 Simple Prefix B⁺ Tree Maintenance 378**
 - 10.6.1 Changes localized to single blocks in the sequence set 378
 - 10.6.2 Changes involving multiple blocks in the sequence set 379
- 10.7 Index Set Block Size 382**
- 10.8 Internal Structure of Index Set Blocks: A Variable Order B-tree 383**
- 10.9 Loading a Simple Prefix B⁺ Tree 386**
- 10.10 B⁺ Trees 390**
- 10.11 B-trees, B⁺ trees, and Simple Prefix B⁺ Trees in Perspective 392**
- Summary 395
- Key Terms 398
- Exercises 399
- Further Readings 404

**11****HASHING**

- 11.1 Introduction 406**
 - 11.1.1 What is hashing? 408
 - 11.1.2 Collisions 409
- 11.2 A Simple Hashing Algorithm 411**
- 11.3 Hashing Functions and Record Distributions 414**
 - 11.3.1 Distributing records among addresses 414
 - 11.3.2 Some other hashing methods 415
 - 11.3.3 Predicting the distribution of records 417
 - 11.3.4 Predicting collisions for a full file 421
- 11.4 How Much Extra Memory Should be Used? 422**
 - 11.4.1 Packing density 422
 - 11.4.2 Predicting collisions for different packing densities 423
- 11.5 Collision Resolution by Progressive Overflow 426**
 - 11.5.1 How progressive overflow works 427
 - 11.5.2 Search length 429
- 11.6 Storing More Than One Record per Address—Buckets 431**
 - 11.6.1 Effects of buckets on performance 433
 - 11.6.2 Implementation issues 437
- 11.7 Making Deletions 440**
 - 11.7.1 Tombstones for handling deletions 441
 - 11.7.2 Implications of tombstones for insertions 442
 - 11.7.3 Effects of deletions and additions on performance 443
- 11.8 Other Collision Resolution Techniques 443**
 - 11.8.1 Double hashing 444
 - 11.8.2 Chained progressive overflow 444
 - 11.8.3 Chaining with a separate overflow area 447
 - 11.8.4 Scatter tables—indexing revisited 448
 - 11.8.5 Extendible hashing 448
- 11.9 Patterns of Record Access 450**
- Summary 451**
- Key Terms 454**
- Exercises 457**
- Further Readings 463**

Appendix A ASCII Table 466

Appendix B String Functions in Pascal: *tools.prc* 467

Appendix C An Introduction to C 472

Appendix D Comparing Disk Drives 524

Bibliography 527

Index 531