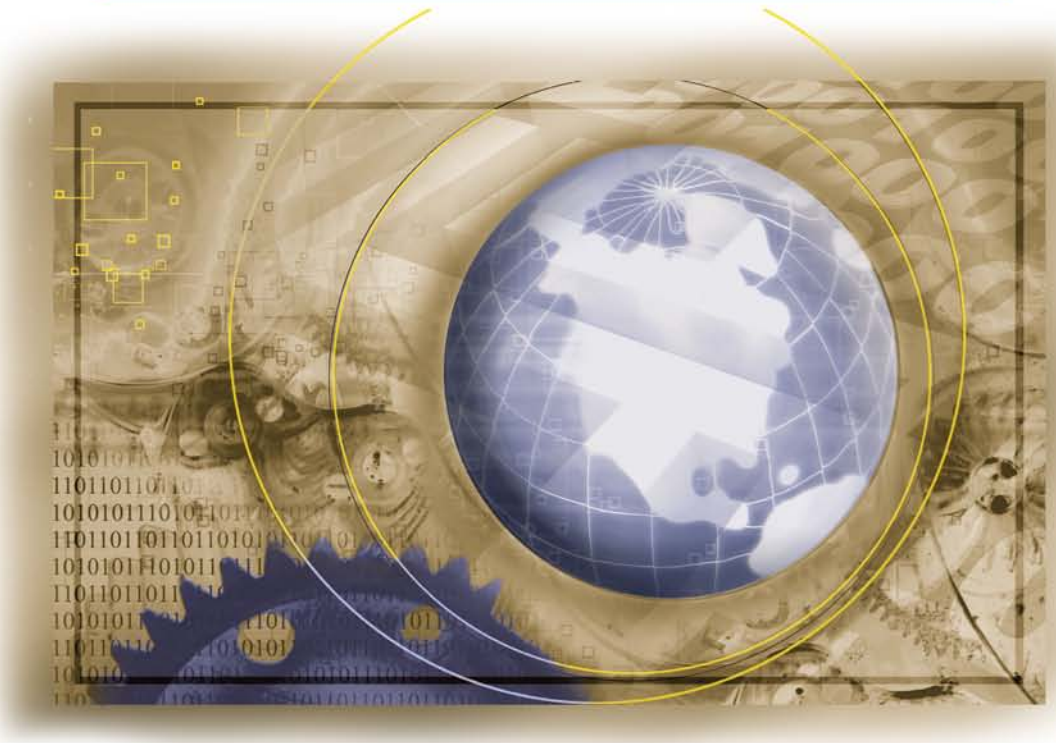
Auerbach Publications
Taylor & Francis Group

Programming Languages for Business Problem Solving



Shouhong Wang
Hai Wang

Programming Languages for Business Problem Solving

Other Auerbach Publications in Software Development, Software Engineering, and Project Management

Accelerating Process Improvement Using Agile Techniques
Deb Jacobs
ISBN: 0-8493-3796-8

Advanced Server Virtualization: VMware and Microsoft Platforms in the Virtual Data Center
David Marshall, Wade A. Reynolds and Dave McCrory
ISBN: 0-8493-3931-6

Antipatterns: Identification, Refactoring, and Management
Phillip A. Laplante and Colin J. Neill
ISBN: 0-8493-2994-9

Applied Software Risk Management: A Guide for Software Project Managers
C. Ravindranath Pandian
ISBN: 0849305241

The Art of Software Modeling
Benjamin A. Lieberman
ISBN: 1-4200-4462-1

Building Software: A Practitioner's Guide
Nikhilesh Krishnamurthy and Amitabh Saran
ISBN: 0-8493-7303-4

Business Process Management Systems
James F. Chang
ISBN: 0-8493-2310-X

The Debugger's Handbook
J.F. DiMarzio
ISBN: 0-8493-8034-0

Effective Software Maintenance and Evolution: A Reuse-Based Approach
Stanislaw Jarzabek
ISBN: 0-8493-3592-2

Embedded Linux System Design and Development
P. Raghavan, Amol Lad and Sriram Neelakandan
ISBN: 0-8493-4058-6

Flexible Software Design: Systems Development for Changing Requirements
Bruce Johnson, Walter W. Woolfolk, Robert Miller and Cindy Johnson
ISBN: 0-8493-2650-8

Global Software Development Handbook
Raghvinder Sangwan, Matthew Bass, Neel Mullick, Daniel J. Paulish and Juergen Kazmeier
ISBN: 0-8493-9384-1

The Handbook of Mobile Middleware
Paolo Bellavista and Antonio Corradi
ISBN: 0-8493-3833-6

Implementing Electronic Document and Record Management Systems
Azad Adam
ISBN: 0-8493-8059-6

Process-Based Software Project Management
F. Alan Goodman
ISBN: 0-8493-7304-2

Service Oriented Enterprises
Setrag Khoshafian
ISBN: 0-8493-5360-2

Software Engineering Foundations: A Software Science Perspective
Yingxu Wang
ISBN: 0-8493-1931-5

Software Engineering Quality Practices
Ronald Kirk Kandt
ISBN: 0-8493-4633-9

Software Sizing, Estimation, and Risk Management
Daniel D. Galorath and Michael W. Evans
ISBN: 0-8493-3593-0

Software Specification and Design: An Engineering Approach
John C. Munson
ISBN: 0-8493-1992-7

Testing Code Security
Maura A. van der Linden
ISBN: 0-8493-9251-9

Six Sigma Software Development, Second Edition
Christine B. Tayntor
ISBN: 1-4200-4426-5

Successful Packaged Software Implementation
Christine B. Tayntor
ISBN: 0-8493-3410-1

UML for Developing Knowledge Management Systems
Anthony J. Rhem
ISBN: 0-8493-2723-7

X Internet: The Executable and Extendable Internet
Jessica Keyes
ISBN: 0-8493-0418-0

AUERBACH PUBLICATIONS
www.auerbach-publications.com
To Order Call:1-800-272-7737 • Fax: 1-800-374-3401
E-mail: orders@crcpress.com

Programming Languages for Business Problem Solving

Shouhong Wang
Hai Wang


Auerbach Publications
Taylor & Francis Group
Boca Raton New York

Auerbach Publications is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20110720

International Standard Book Number-13: 978-1-4200-6265-6 (eBook - PDF)

Contents

Preface

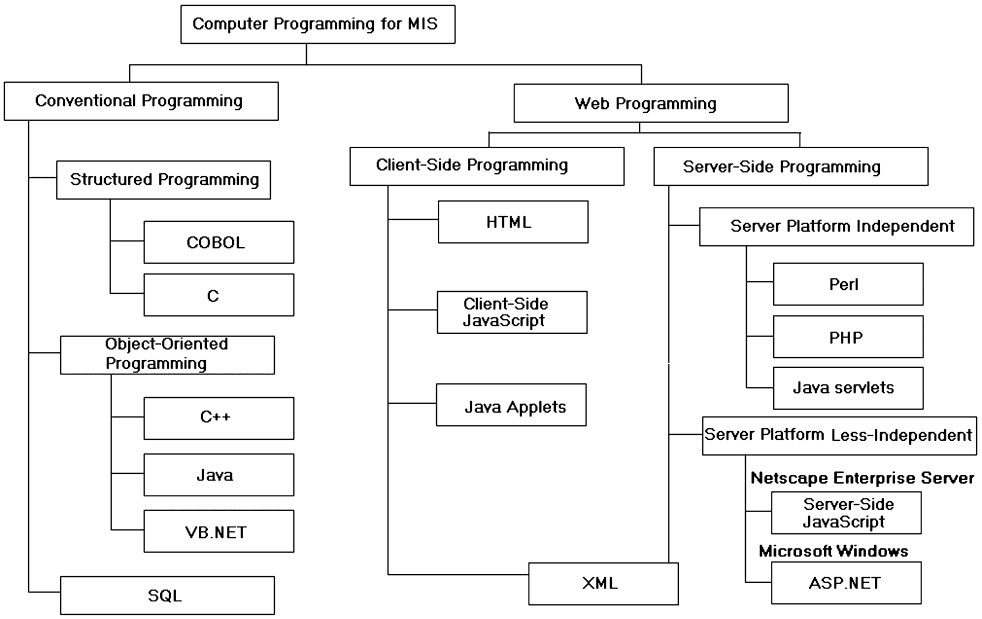
In the information technology era, the computer literacy of managers becomes more and more crucial for business success. Students in management information systems (MIS) must acquire fundamental theories of information systems and essential practical skills in computer applications. They must also develop the ability for life-long learning in information technology during their business education.

In the information technology era, the education for MIS students is quite challenging. Generally speaking, we would like to develop students who are more business oriented and problem driven in applying computer technology. Instead of teaching technical details (such as syntax and less-commonly used commands) and computational algorithms, we focus on the matching of business problems and computer solutions.

Interestingly, more than two decades after so-called fourth-generation computer languages proliferated in 1980s, the information industry still heavily relies on third-generation computer languages such as C, C++, Java, VB.NET, etc. In fact, in the modern computer age, there is a great variety of computer languages. A brief summary of computer languages commonly used in the MIS field is shown in the figure on the next page.

To meet the challenge of the ever changing computer technology, we must teach core components of computer technology in business and encourage students to develop their ability to learn independently.

This textbook is designed for business undergraduate MIS majors who study computer problem solving and programming for business. To avoid unnecessary overlap with the components of computer applications to the business fields covered in other courses, this book excludes word processing, databases and spreadsheets, which are expected to be taught in other information systems courses. The emphasis of this textbook is placed on computer languages that are commonly used in today's information systems for business, including



COBOL, C, C++, HTML, JavaScript, Java, VB.NET, Visual Basic for Applications, Perl, PHP, ASP.NET, XML, and SQL. This textbook consists of eleven chapters. It can be used for two computer programming courses for MIS majors: one is an introductory programming course, and the other is a Web application development course that focuses on server-side programming. The instructor may choose several chapters for a one-semester course, depending upon the needs. Overall, this book is a survey handbook of programming languages for MIS majors.

Chapter 1 provides an overview of COBOL which is old, but is still one of the most popular computer languages used in management information systems. It begins by describing the concepts of data files. Data file processing is not only the central part of COBOL, but also important for all third-generation computer languages. In this chapter, student will learn three organization structures of data files, and their applications to business data process. Using an example of payroll processing, this chapter explains the use of core statements of COBOL in business data processing. The appendices of this chapter provide manuals for running COBOL programs on mainframe.

Chapter 2 explains C++ and object-oriented programming. C++ are more likely to be used in software development instead of in business applications directly; nevertheless, the concept of object-oriented programming of C++ has been generally applied to all fields of business computing. Through several examples, students will learn basic skills of object-oriented programming.

Chapter 3 introduces tools of Web page development—HTML and JavaScript. In this chapter, students will learn how to develop Web pages by applying these so-called client-side programming tools, and will understand the source behind Web pages.

Chapter 4 discusses the modern computer language Java. Java covers a huge computational spectrum. In this chapter, students will learn Java applets, standard

Java programming (AWT and non-AWT), and Java servlets. The focal point of this chapter is placed on the computing capability of Java in the Internet computing environment.

Chapter 5 introduces Visual Basic (VB) and the concept of graphical user interface. VB.NET is becoming one of the most popular computer languages in MIS computing. Through several business application examples, this chapter explains core features of VB.NET.

Chapter 6 is an extension of Chapter 5. It provides an overview of Visual Basic for Applications (VBA), which is integrated in Microsoft spreadsheets (Excel) and databases (Access). Through this chapter, students will learn how to use Visual Basic for Applications to integrate Excel spreadsheets and develop decision support systems.

Chapter 7 introduces Perl for CGI programming. Students will learn typical examples of Perl programming, and understand the principles of the use of Perl to implement Web-based business applications based on the conventional CGI technology.

Chapter 8 introduces PHP script. Students will learn typical examples of PHP programming, and understand the unique features of PHP to implement Web-based business applications.

Chapter 9 introduces ASP.NET, the Web application development environment of the Microsoft platform. Students will learn major features of ASP.NET which are similar to Java servlets, CGI programming, or PHP in terms of Web application development, as well as unique features of ASP.NET.

Chapter 10 introduces XML. Students will learn typical examples of XML programming, and understand the application context of XML and its structure. The concepts of XHTML and XBRL are also introduced in this chapter.

Chapter 11 introduces SQL. Students will learn major features of SQL for querying and updating the databases. The role of SQL in Web applications is also briefly introduced through an ADO.NET example.

This textbook makes a balance between the classical information system languages, namely, COBOL and C++, and other modern computer languages such as Java and ASP.NET. Through the study of these computer languages, students will learn the migration of computer languages. The book introduces many key concepts of procedural programming languages in Chapters 1 and 2 for beginners. To avoid unnecessary duplications, other chapters do not repeat these key concepts in detail while applying them to examples. The book summarizes the six key concepts shared by all procedural programming languages on the last pages of the book. This summary is particularly useful for students who might not start with Chapters 1 and 2, but need to understand these concepts during the study of other chapters.

Students who use this book are expected to develop practical skills as well as have a bird's-eye view of computer programming. The objective of this book is that students will understand the characteristics of traditional file processing in legacy information systems, the philosophies of structured programming and object-oriented programming, the means of multimedia presentations on the Internet, the concept of human–computer interface design and decision support systems, and the basics of Web-based business applications for

Key Knowledge Elements	*Requirements of IS Education (IS'02—Competency Level and Body of Knowledge Elements)*	***COBOL** (Legacy File Processing)* ***SQL (Database query)***	***C** (Classical Function-Oriented)* ***C++*** *(Classical Object Oriented)*	***HTML*** ***JavaScript*** ***PHP*** ***Java Applets*** *(Web Applications—Client-Side)*	***Java Servlets*** ***APS.NET*** ***Perl*** ***XML*** *(WebApplications—Server-Side)*	***VB.NET*** ***Java (AWT)*** *(Graphical User Interfaces)*
File processing	4 1.2.1 Formal problems and problem solving	x				
Simple data types	4 1.2.4 Abstract data types	x	x	x	x	x
If-then control	4 1.2.1 Formal problems and problem solving	x	x	x	x	x
Loop	4 1.2.1 Formal problems and problem solving	x	x	x	x	x
Function orientation	3 1.2.4.4 Modules and coupling	x	x	x		
Object orientation	3 3.3.6 Object-oriented methodologies		x	x	x	x
Web page development	4 3.9.7 Software development			x	x	
Client-server computing	3 3.1.2 Systems concepts			x	x	
Graphical user-computer interfaces	4 3.9.6 Human-computer interfaces			x	x	x
Environments of languages	3 1.3.7 Programming languages, design, implementation	x	x	x	x	x

e-commerce. Upon completion of the courses, students should be able to write computer programs in these computer languages to solve simple business problems related to information systems. More importantly, students will have developed the ability to learn details of these programming languages and new programming languages by themselves.

The relationships between the key knowledge elements offered in individual languages and the general requirements of IS education is summarized in the table earlier in this section. As shown in the table, the IS2002 model curriculum <http://www.is2002.org/> defines computer literacy as a set of knowledge elements related to problem solving using computer technology. Most knowledge elements listed in the table, such as abstract data types, modules and coupling, software development, human–computer interfaces, object-oriented methodologies, and programming languages can be acquired only through learning these computer languages.

In this textbook, a huge amount of material about computer languages that are commonly used in the modern MIS computing field are boiled down to a practical workable volume. Students must use this textbook fully. The textbook includes many practical computer programming examples. Reading these examples is necessary, but not sufficient. To learn computer programming, students must perform hands-on practices on computers. Students are required to conduct several programming projects for the courses. In order to learn the syntax and programming techniques through the textbook examples before they start working on a project for a language, students ought to edit (type and think!) these programming examples by themselves, run these programs, and examine the execution results. In many cases, students should read additional relevant reference books and manuals about these computer languages to complete course projects in creative ways. Similar to learning human natural languages, students are always encouraged to learn more about programming languages beyond a single textbook.

In summary, this textbook is to teach students "how to walk," and students are supposed to learn "how to run."

Finally, the first author would like to thank the faculty members and students of Earle P. Charlton College of Business, University of Massachusetts Dartmouth, who have encouraged the teaching–learning approach of "one-course-to-many-languages" for management information systems majors since 1998.

Shouhong Wang, PhD
University of Massachusetts, Dartmouth
<swang@umassd.edu>

Hai Wang, PhD
Saint Mary's University,
Halifax, Nova Scotia
<hwang@smu.ca>

List of Credits

Alpha VMS COBOL, VMS C, and VMS C++ are trademarks of Hewlett-Packard and COMPAQ.

PowerTerm for Windows is a trademark of ERICOM Software.

Windows, MS-DOS, Notepad, WordPad, Windows Explorer, Internet Explorer, Visual Basic, Visual Basic for Applications, Excel, Access, Jscript, Active Server Pages (ASP), Window Media Player, .NET, VB.NET, C#, ASP.NET, Visual Studio .NET, Visual Web Developer 2005 Express Edition, are trademarks of Microsoft Corporation.

Java, J2SE, Java 2 SDK, JDK, JSWDK, and Javascript are trademarks of Sun Microsystems.

Netscape is trademark of Netscape Communications Corporation.

ActivePerl is a trademark of ActiveState Tool Corp.

PHP is copyrighted by The PHP Group.

Apache is copyrighted by The Apache Software Foundation

EasyPHP is copyrighted by EasyPHP.

XML is a trademark of World Wide Web Consortium (W3C).

Typographical Conventions

Certain typographical conventions are adopted throughout the book.

Bold text is used to set off important words or concepts.

`Mono-space` text is used for computer language code. The numerical line numbers marked in computer programs are used for explanations, and should not be typed for programming.

`[Mono-space]` text is used to indicate menu items, or general terms of user-defined statements. The brackets [ ] should not be typed for programming.

`Mono-Space` text is used to indicate general names of variables and arguments in syntax descriptions.

`Underlined` text is used to indicate user-typed commands in the operating system.

Compilers/Interpreters Used for the Programs in This Book

COBOL: VMS COBOL
C/C++: VMS C/C++
HTML: Netscape or Microsoft Internet Explorer
JavaScript: Netscape or Microsoft Internet Explorer
Java Applets, Free-Standing Java: JDK (for Windows)
Java Servlets: JDK and JSWDK (for Windows)
VB.NET: Microsoft .NET 2005
Visual Basic for Application: Microsoft Excel
Perl: ActivePerl (for Windows)
PHP: EasyPHP
ASP.NET: Microsoft .NET 2005
XML: Microsoft Internet Explorer
SQL: Microsoft Access, and Microsoft .NET 2005

Chapter 1

COBOL and File Processing

1.1 Introduction to COBOL

Before the early 1960s, no language was specifically suited to business tasks. In 1960, **C**ommon **B**usiness **O**riented **L**anguage (COBOL) was developed by Conference on Data Systems Languages (CODASYL). COBOL was a great success. In fact, a significant proportion of software in many management information systems (MIS) was written in COBOL and is still in use. Such systems are called **legacy systems** in that they are too valuable to discard but they represent a drain on information systems resources in maintenance and reengineering. COBOL has been standardized by American National Standard Institute (ANSI). Knowledge of COBOL is definitely an asset for business students who pursue their major in information systems.

Traditionally, COBOL is a file processing language. However, COBOL on most mainframe computers can be a **host language** of a **database management system**; namely, programmers are able call up functions of a database management system to manipulate databases in a COBOL program. This book focuses on COBOL for file processing.

Typically, COBOL programs run on mainframe computers, as illustrated in Appendix 1.2. However, there have been PC versions of COBOL compilers. NetCOBOL (previously called Fujitsu COBOL) is one of them, which is available on the Internet and can be downloaded free of cost.

1.2 Legacy Information Systems

Before the 1970s, MIS were usually built on mainframe computers. Such computing environment is called monolithic computing. In monolithic computing,

the mainframe computer processes all information for the business, although there might be many dummy terminals linked to the computer. The computer languages used in that period are known as **third generation computer languages** (3GL), such as COBOL and PL/1. 3GL have simple human-computer interfaces and limited built-in computational functions. A programmer who is using 3GLs must learn the syntax of the language, and describe detailed procedures to instruct the computer to perform a processing task. Today, in the year 2007, many enterprises are still using computer programs developed 30–40 years ago. This is because these computer programs are still useful and valuable. These information systems built in the 1960s and 1970s are called **legacy systems** as has been defined earlier.

The issues of legacy systems are not easy to solve. The reality is that the information industry still needs people to maintain legacy systems by manipulating 3GL. For the purpose of electronic commerce education, it is imperative that MIS students have essential knowledge and practical programming skills of third generation languages. By learning 3GL (typically, COBOL), students understand what is taking place within the computer during processing of business information. Practical programming skills are certainly beneficial for students in lifelong learning and job selection.

COBOL, FORTRAN, BASIC, APL, PL/1, and C are a few of the many 3GL. Third generation computer languages are flexible to use, and have been the foundations of computer applications. In fact, **fourth generation computer languages** (4GL) are built upon third generation languages, and many spreadsheets and database management systems are commonly implemented in C.

One of the common characteristics of 3GL is **file processing**, which is important for MIS. In business, people need to process huge volumes of data, such as personnel data, customer data, and inventory data. Because the CPU memory is limited and volatile, these data must be stored on a secondary storage medium such as disk. However, it takes much longer time to read or write data on secondary storage than it takes in the CPU memory. We will describe how these data are organized into files and processed. File is the simplest form of data storage, and is the basis for **data base**. In database courses you will learn more about the differences between file processing systems and database systems.

1.2.1 File, Record, Data Item, and Key

Terms related to data processing that are commonly used in the field are:

File A collection of a set of records. Note that, the term file is often overloaded. For example, a computer program is stored on disk as a "file." However, in this section, we use file exclusively for **data file**, but not computer program file.

- **Record** A description of a single transaction or entity in a repetitive file. A record is a collection of a set of data items.
- **Data Item**, or **Field**, or **Attribute** A description of a single characteristic of a transaction or entity. The three terms are interchangeable.

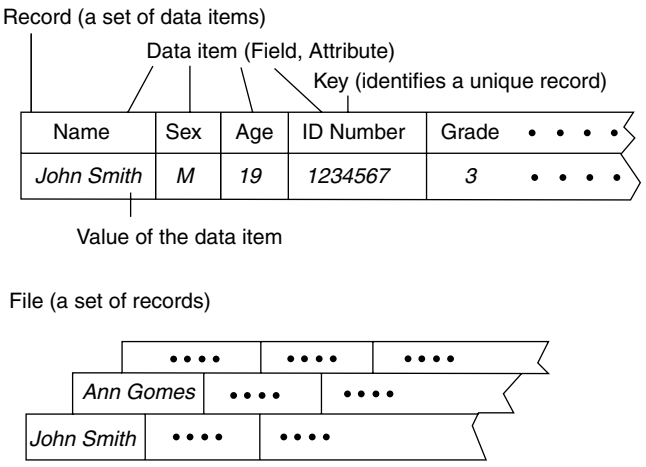


Figure 1.1 File, record, data item, and key.

- **Key** A data item, or a group of data items, that uniquely identifies a record in a file.
- **Data** Individual facts, or the value of a data item.

The hierarchy representing the relationship between file, record, data item, and key is illustrated in Figure 1.1.

1.2.2 Tape and Disk

Because CPU memory is limited and temporary, computer programs and data must be stored on secondary storage. Paper, magnetic tape, magnetic disk, and optical disk are some of the many forms of secondary storage. Magnetic disk is the most commonly used second storage media. Second to disk, magnetic tape is also often used in mass data storage. Conceptually, a magnetic tape for computers is similar to a cassette. Figure 1.2 illustrates the two major secondary storage devices.

1.2.3 Three Basic File Organizations

1.2.3.1 Sequential File

In a sequential file, records are stored in physically adjacent locations on the storage medium (e.g. tape, disk, paper printout) as depicted in Figure 1.3.

To find a particular record, the computer has to search the file by checking the records one by one, starting at the beginning of the file, until it finds the desired record or reaches end of file (EOF) in the case of failure.

Advantages:

1. Saves space.
2. No record key is required.

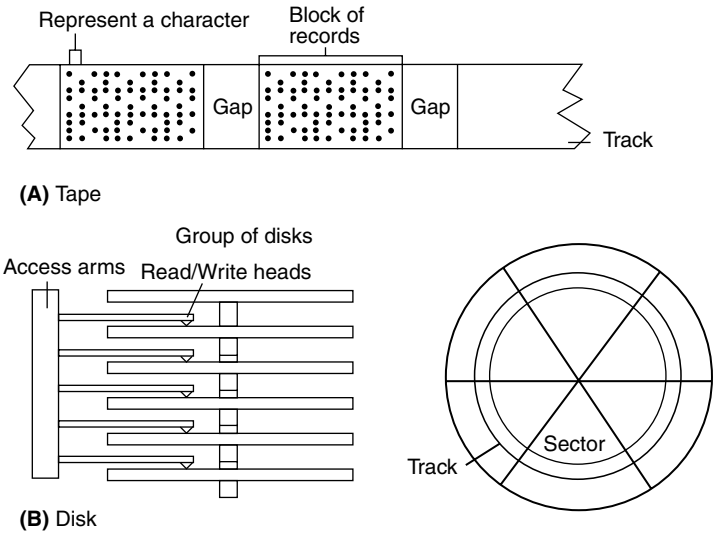


Figure 1.2 Tape and disk.

3. Efficient when all of the records are sequentially processed (e.g. payroll processing).
4. Can be implemented on any media.

Disadvantages:

1. It would take a long time to find a particular record.
2. It is difficult to update. For example, it is difficult to insert a record.

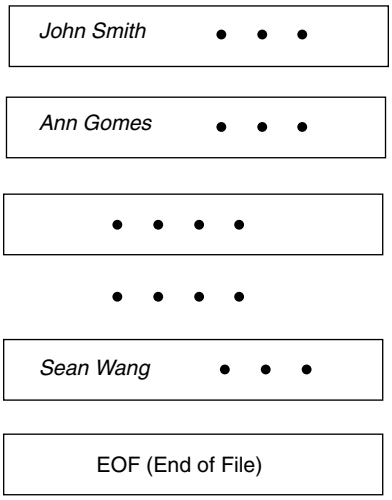


Figure 1.3 Sequential file.

1.2.3.2 Random File

A random file organization allows immediate, direct access to individual records in a file. The essence of random accessing is the ability to quickly produce an address from a record's key. Note that random files are only for disk, not for tape or paper.

The fundamental component of data access in the random file organization is the conversion of the record key of a record to the address of the record on the disk though a formula called **hashing function**.

The feature of hashing functions can be explained with an example. However, hashing functions used in real systems are much more complicated than the example given. Here, hashing function is defined as:

Address of record = remainder of [Record Key/111]

Suppose there is a record with its Record Key = 4567. Then,

Address of record = remainder [4567/111] = 16

That is, this record (its key value = 4567) is stored to the location with the address 16. Later, if one wants to find a record with the key = 4567, then the computer will quickly calculate the address according to the hashing function and immediately find it at this address.

Conflict Problem

No matter how sophisticated a hashing function is, there always exist synonyms, or conflicts; that is, several key values map onto the same address. For example, in the above case of hashing function,

```
Key 4567    ==>  Address 16
Key 349     ==>  Address 16
Key 1126    ==>  Address 16
    . . . . . .
```

To solve this problem, data structure techniques of pointer and overflow area are commonly used in dealing with conflict. Figure 1.4 briefly illustrates the solution.

You will learn that COBOL supports random files, and it can handle conflict automatically.

Advantages:

1. It can access an individual record very fast.
2. It is efficient in updating (e.g., adding and modifying records).

Disadvantages:

1. Sequential access is impossible.
2. A record key is necessary.

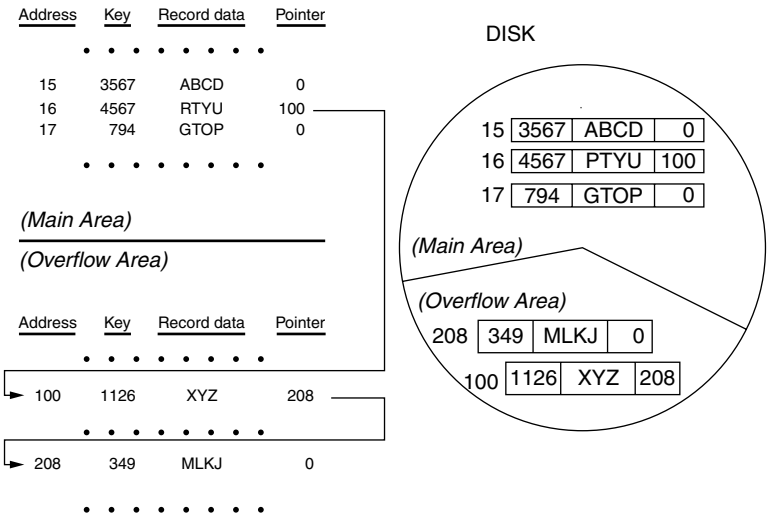


Figure 1.4 Conflict and its solution in random files.

3. Wastes spaces.
4. Synonyms make the process slower.

1.2.3.3 Indexed File

An indexed file keeps an index table which associate record keys with addresses. An index table is comparable to a content table in a textbook (see Figure 1.5).

An index table may be sorted based on a sequence of the record key values if needed. Usually, an index table is small and can be manipulated in the CPU memory. Hence, the conversion of a record key to the address of the record takes little time. However, if a data file is extremely large, then its index table will be so large that the index table itself is a disk file supported by a "second level" index table.

Advantages:

1. It can be used in both sequential and random access. Processing speed is fairly good for both.
2. It is efficient in adding a record, sorting and updating the file.

Disadvantages:

1. A record key is necessary.
2. If the data file is huge, then several levels of index tables are needed, and processing will be slow.

COBOL provides functions to support all of these three file organizations. COBOL programmers do not design the access mechanism in detail; that is, programmers do not define hashing functions for a random file and do not

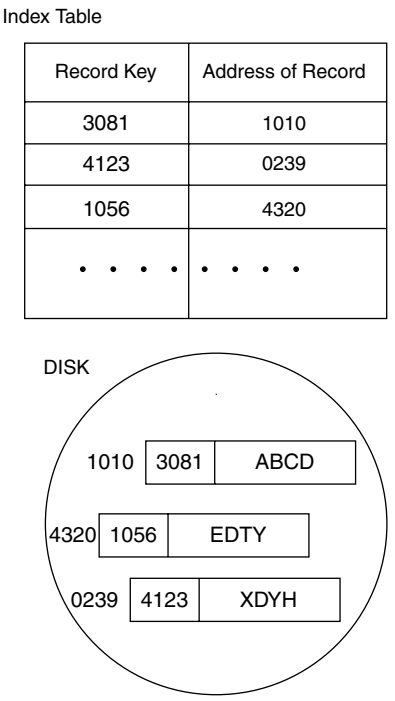


Figure 1.5 Index table in an indexed file.

build an index table for an indexed file. They only write sentences in a COBOL program to declare what organizations of files are being used.

1.2.4 Types of Business Data Files

An information system usually uses many data files for business processes. Some of the major types of files in business data processing are elaborated.

1.2.4.1 Master Files

A master file stores permanent data for the firm. For example, an employee file in the payroll system, a student file in the registration system, and an inventory file in the warehouse system are all master files. Master files can be read, modified, and appended, but can never be deleted from the system. Usually, a master file can be processed not only sequentially, but also randomly. For instance, the marketing department not only needs to quickly find a particular customer from the customer master file, but also would like to print out an entire list of all the customers. Thus, the organization for a master file is usually an indexed file.

1.2.4.2 Transaction Files

A transaction file stores day-to-day business operational data. An order log, which records date of order, number of the item to be ordered, amount of

order, etc., in the order processing system is a transaction file. A transaction file is used to update a master file. Usually, a transaction file must be processed from the first record to the last one. Also, no record key is usually involved in transaction file. Hence, a transaction file is usually a sequential file. In principle, transaction files are temporary.

1.2.4.3 Reference Files

Reference files provide reference data. For example, tax rate tables and zip code tables are reference files. These are permanent data files and are usually random files.

1.2.4.4 Backup Files

For the purposes of security and archive, a data file is often saved to a backup file from time to time. The created backup file is occasionally used for system recovering or auditing. To save space, the sequential organization is used for backup files.

1.2.4.5 Working Files

In processing a huge amount of data, CPU memory may not be large enough. Hence, to use the second storage as an extended memory, the programmer must create working files. Working files are temporary. All the three basic file organizations may be used for working files. Few working files are used in a simple computer program.

1.2.4.6 Report Files

Report files are different from other types of files discussed earlier, in that report files are used to present information for humans, and are stored (or printed) on paper. Physically, the organization of a report files is always sequential.

1.2.5 Design of Organizations of Files

As discussed earlier, each of the three organizations of data files has its advantages and disadvantages. The design of the organization of a file always depends upon the business application context. However, there are general match relationships between the types of data files and organizations, as summarized in Figure 1.6. For instance, a master file is commonly an indexed file, but can also be a random file if sequential access is not required.

1.3 General Structure of COBOL—Four Divisions

A COBOL program is organized like the English language, but is much formalized. It is divided into four divisions, each of which must be dealt with in order.

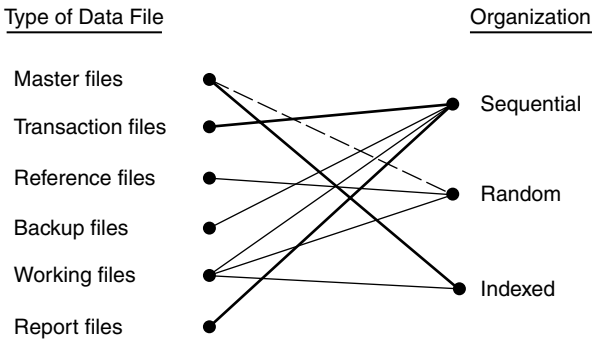


Figure 1.6 Select the organization for a data file.

1. **IDENTIFICATION DIVISION** shows the name of the program, and may give any other documentation deemed necessary.
2. **ENVIRONMENT DIVISION** describes the hardware configuration, and specifies the data files.
3. **DATA DIVISION** describes the data structure of files and working storage used in the program. The formats of data records are described in this division. Working storage in the main memory is also defined in this division.
4. **PROCEDURE DIVISION** expresses the program execution logic. The previous three divisions do not cause explicit actions of execution. Only this division contains the processing instructions.

The following is a toy-example COBOL program. The program in Listing 1.1 is to display "HELLO, WORLD!" on the computer screen. This example is used to explain some concepts of COBOL.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.    EXAMPLE1.
ENVIRONMENT    DIVISION.
DATA           DIVISION.
PROCEDURE      DIVISION.
DISPLAY-MY-MESSAGE.
    DISPLAY    "HELLO, WORLD!".
    STOP RUN.
```

Listing 1.1 Simple COBOL example.

1.4 COBOL Words

A COBOL program is a set of **COBOL words** and **symbols**. A legal COBOL word consists of character (A–Z), digits (0–9), and hyphens (-). It must contain at least one character, and must be no longer than 30 letters. All characters in a COBOL word must be in capital. A legal COBOL word must not contain a space, and must not start with or terminate with a hyphen. A COBOL symbol

could be +, −, *, /, $, etc. Any COBOL word and symbol must be surrounded with spaces.

COBOL words are of two types, **reserved words** and **user-defined words**. COBOL reserved words have specific meanings and represent specific features or functions. In the example of Listing 1.1, those in bold font (just for the purpose of illustration) are COBOL reserved words, whereas, user-defined words are **program-name**, **data-names**, and **procedure-names**, defined by the programmer. A list of commonly used COBOL reserved words is shown in Appendix 1.1. Reserved words cannot be used as user-defined words. To avoid misuses of reserved words, we often use digits (0–9) and hyphens (-) in addition to alphabetic characters (A–Z) for user-defined words.

One of the characteristics of COBOL is self-documenting. COBOL requires very precise structure, whereas, users may define data names and procedure names as naturally as possible.

1.5 COBOL Program Format—Positioning, Spacing, and Punctuation

In COBOL, each sentence must be written in a specific format. For instance, a certain type of sentence must be written from a certain column, and each sentence must end with a **period**. In the standard COBOL coding form, any COBOL sentence starts after positions 8 except for a comment line which must have an asterisk sign (*) in position 7. Some sentences even start after position 12. However, many computer editors, such as the VMS Editor on Alpha computer, changed the rules and made online editing easier (see Appendix 1.2).

To improve the readability of the program, programmers often insert comment lines in the program (see an example in Listing 1.2 where the first column is assumed to be position 7). During compiling, computer ignores those comment lines.

```
*******************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID.   MYCBL1.
*******************************
 ENVIRONMENT    DIVISION.
*                              *
* (SELECT FILES HERE)           *
*                              *
*******************************
 DATA           DIVISION.
*                              *
* (DEFINE DATA STRUCTURE HERE) *
*                              *
*******************************
 PROCEDURE     DIVISION.
 DISPLAY-MY-MESSAGE.
```

```
     DISPLAY    "HELLO, WORLD!".
     STOP RUN.
********* E N D ****************
```

Listing 1.2 Documenting COBOL programs by inserting comment lines.

Some important selected rules of COBOL programming are:

1. A word should not exceed 30 characters (no space), and should not begin or end with a hyphen.
2. At least one blank space must appear between words, or between a word and an arithmetic operator.
3. Each sentence must be ended with a period, and it must be followed by at least one blank space.

1.6 Typical Examples of COBOL Programs

Remembering syntax and rules through rote memorization is the most tedious method of learning computer languages. "Reading and programming" is the best method to learn COBOL. The first step of learning COBOL is to read several typical COBOL programs. To provide typical COBOL programs, four examples of COBOL program for a payroll processing system are given in Listings 1.3, 1.4, 1.5, and 1.6. After each of the example programs, we provide detailed explanations for the COBOL sentences used in the program.

The purpose of the first COBOL program in Listing 1.3 is to build an indexed file for employees. Once the employee master file has been built, it can be used for queries and other data processing. The second COBOL program in Listing 1.4 is to create a sequence data file that records each employee's work hours. These two programs in our examples are actually data entry support programs. The third COBOL program in Listing 1.5 manipulates the data files created by the previous two programs for payroll processing. The fourth COBOL program in Listing 1.6 is a maintenance program that is used to make modifications to the master file. The relationships between the COBOL programs and the data files are depicted in Figure 1.7.

1.6.1 Build a Master File

To build a master file for a COBOL application, one must create a COBOL program to allow the user to do data entry. The computer program accepts the data from the keyboard, and then writes them to the disk. Since a master file is usually used for various purposes, the organization of master file should be indexed.

```
1    IDENTIFICATION    DIVISION.
2    PROGRAM-ID.       EMPLOYEE.

3    ENVIRONMENT       DIVISION.
4    CONFIGURATION     SECTION.
5    SOURCE-COMPUTER.  ALPHA.
```

```
 6    INPUT-OUTPUT      SECTION.
 7    FILE-CONTROL.
 8         SELECT  EMPLOYEE-FILE
 9          ASSIGN TO "EMP.DAT"
10            ORGANIZATION IS INDEXED
11            ACCESS MODE IS DYNAMIC
12            RECORD KEY IS EMPLOYEE-ID.

13    DATA              DIVISION.
14    FILE              SECTION.
15    FD  EMPLOYEE-FILE
16        LABEL RECORDS ARE STANDARD.
17    01  EMPLOYEE-RECORD.
18        02  EMPLOYEE-ID         PIC X(2).
19        02  EMPLOYEE-NAME       PIC X(30).
20        02  EMPLOYEE-ADDRESS    PIC X(30).
21        02  EMPLOYEE-PAY-RATE   PIC 99.

22    WORKING-STORAGE   SECTION.
23    01  W-EMPLOYEE-ID           PIC X(2)  VALUE "00".

24    PROCEDURE         DIVISION.
25    MAIN-PROCEDURE.
26         PERFORM INITIALIZE-PROCESS.
27         PERFORM ACCEPT-DATA-PROCESS
28                 UNTIL W-EMPLOYEE-ID = "99".
29         PERFORM TERMINATION-PROCESS.
30         STOP    RUN.

31    INITIALIZE-PROCESS.
32         OPEN  OUTPUT  EMPLOYEE-FILE.
33         DISPLAY  "INPUT EMPLOYEE ID NUMBER (2 DIGITS):".
34         ACCEPT W-EMPLOYEE-ID.

35    ACCEPT-DATA-PROCESS.
36         MOVE  W-EMPLOYEE-ID  TO  EMPLOYEE-ID.
37         DISPLAY  "INPUT EMPLOYEE NAME (30 LETTERS):".
38         ACCEPT  EMPLOYEE-NAME.
39         DISPLAY  "INPUT EMPLOYEE ADDRESS (30 LETTERS):".
40         ACCEPT  EMPLOYEE-ADDRESS.
41         DISPLAY  "INPUT EMPLOYEE PAY RATE (2 DIGITS):".
42         ACCEPT  EMPLOYEE-PAY-RATE.
43         WRITE   EMPLOYEE-RECORD
44              INVALID KEY DISPLAY "INVALID KEY ERROR!".
45         DISPLAY "INPUT NEXT EMPLOYEE ID NUMBER (2 DIGITS):".
46         ACCEPT W-EMPLOYEE-ID.

47    TERMINATION-PROCESS.
48         CLOSE  EMPLOYEE-FILE.
49         DISPLAY  "THANK YOU FOR DATA ENTRY!".
```

Listing 1.3 Example of COBOL program: build a master file.

Note that the line numbers in the program are just for references, and must not be typed in the program. Before we learn detailed syntax of COBOL statements, we explain how the COBOL program in Listing 1.3 works.