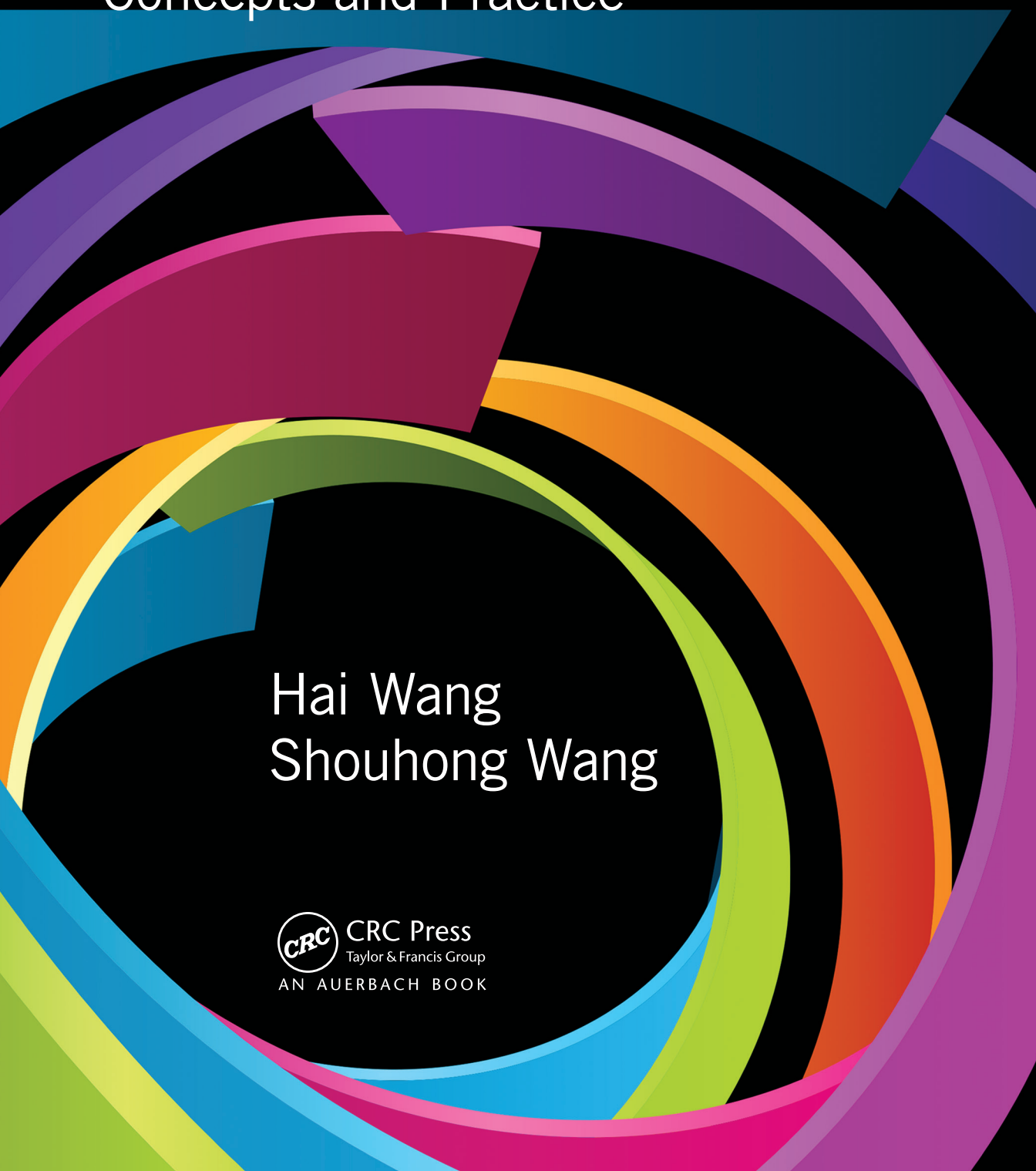
Programming Languages for MIS

Concepts and Practice



Hai Wang
Shouhong Wang

CRC Press
Taylor & Francis Group
AN AUERBACH BOOK

Programming Languages for MIS

Concepts and Practice

Programming Languages for MIS

Concepts and Practice

Hai Wang
Shouhong Wang

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
AN AUERBACH BOOK

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

Version Date: 20130925

International Standard Book Number-13: 978-1-4822-2267-8 (eBook - PDF)

Contents

PREFACE xi

THE AUTHORS xv

ACKNOWLEDGMENTS xvii

CHAPTER 1 INTRODUCTION 1
1.1 Computers 1
1.2 Computer Programming Languages 1
1.2.1 Role of Computer Programming Language 1
1.2.2 Software Systems 2
1.2.3 Taxonomies of Computer Programming Languages 3
1.3 Computing Architecture in the Internet Environment 4
1.4 Key Characteristics Shared by All Procedural Programming Languages 5
1.4.1 Syntax, Sentence, and Word 5
1.4.2 Variable 5
1.4.3 Arithmetic Operation 6
1.4.4 Execution Sequence 6
1.4.5 If-Then-Else Logic 6
1.4.6 Loop 6
1.4.7 Module 7

CHAPTER 2 C++ 9
2.1 Introduction to Function-Oriented and Object-Oriented Programming 9
2.2 A Tour of C Language 9
2.2.1 C and C++ Keyword and User-Defined Word 14
2.2.2 Comment Statements 14
2.2.3 Preprocessor 14
2.2.4 Namespace 14
2.2.5 Structure of a C Program, Functions, and Arguments 15
2.2.6 Statements and Semicolon 16
2.2.7 Data Type 16
2.2.8 Arithmetic Operations 16
2.2.9 `for`-Loop 17

2.2.10 printf() Statement with Conversion Specifier 18
2.2.11 if-Statement 18
2.2.12 String and String Processing 20
2.3 Functional Approach 20
2.3.1 Functional Decomposition 20
2.3.2 A Simple Example of User-Defined Function 21
2.3.3 Declaration of User-Defined Function 22
2.3.4 Calling-Function and Called-Function 22
2.3.5 Structure Diagram 23
2.3.6 An Example of Two Functions 23
2.3.7 An Example of Multiple Functions 25
2.4 Object-Oriented Approach 29
2.4.1 Object and Class 29
2.4.2 Descriptions of Class 31
2.4.3 public and private Statements 32
2.4.4 Constructor 32
2.4.5 Use of Class—Declare Object and Message Sending 32
2.5 Design of Objected-Oriented Program 35
2.6 Connection between Classes—An Example with Two Classes 39
2.7 An Example of Inheritance 43
2.8 Identify Class 48
2.9 Debugging 48
Appendix 2.1: Commonly Used C and C++ Keywords 52
C and C++ Keywords 52
C++ Only Keywords 52

CHAPTER 3 HTML, JAVASCRIPT, AND CSS 53
3.1 Introduction to the Internet 53
3.2 Creating Web Pages Using HTML 54
3.3 Simple Container Tags 55
3.3.1 <HTML> 55
3.3.2 <HEAD> and <TITLE> 55
3.3.3 <BODY> 55
3.3.4 Comments <!-- ... --> 55
3.3.5 Headings <H1> <H2> ... <H6> 56
3.3.6 <P> 56
3.3.7 <I> 56
3.3.8 <TABLE>, <TH>, <TR>, and <TD> 56
3.3.9 <A> 56
3.3.10 <CENTER> 56
3.4 Empty Tags 56
3.4.1 <HR> 56
3.4.2
 56
3.4.3 <IMG> 57
3.5 Complex Container Tags 59
3.5.1 <FORM> 59
3.5.1.1 Attribute ACTION 59
3.5.1.2 Attribute METHOD 60
3.5.1.3 <INPUT> and Its Attributes TYPE, NAME, SIZE, and VALUE 60
3.5.2 FRAME and FRAMESET 60
3.6 Publish Web Page 61

3.7 Introduction to JavaScript 61
3.8 Image Manipulation 62
3.8.1 Object Classes and Their Methods and Attributes 63
3.8.2 Event Handler 64
3.9 FORM Input Data Verification 64
3.9.1 Comparison of JavaScript with C and C++ 66
3.9.2 Function and Calling a Function 67
3.9.3 String Processing 68
3.9.4 if-Statement 68
3.9.5 alert-Statement 69
3.10 FORM Data Calculation 69
3.11 Cookies 71
3.12 Miscellaneous JavaScript Statements 74
3.12.1 new Statement 74
3.12.2 Miscellaneous Functions and Methods 74
3.13 Cascading Style Sheet 74
3.13.1 Inline CSS 75
3.13.2 Internal CSS 76
3.13.3 External CSS 79
3.14 Debugging Source Code of Web Pages 80
Appendix 3.1: List of HTML Commonly Used Tags 85
Appendix 3.2: JavaScript Reserved Words and Other Keywords 86
JavaScript Reserved Words 86

CHAPTER 4 VB.NET 87
4.1 Graphical User Interface 87
4.2 Microsoft Visual Studio and VB.NET Environment 87
4.3 Event Driven 90
4.4 Example of a Single Form 93
4.5 Multiple Forms 96
4.5.1 Design Forms 96
4.5.2 Module 98
4.5.3 Class 99
4.5.4 Coding 100
4.6 Programming with VB.NET 106
4.6.1 General Format of Code, Comments, and Keywords 106
4.6.2 Class and Object 108
4.6.3 Methods 108
4.6.4 Constant Variables 109
4.6.5 Data Types 109
4.6.6 Arithmetic Operations 109
4.6.7 If-Then-Else Statement 110
4.6.8 For-loop 110
4.6.9 String Processing and Format Statement 110
4.6.10 Print Document 110
4.6.11 Message Box 111
4.7 Debugging 111

CHAPTER 5 C#.NET 115
5.1 Microsoft Visual Studio and C# Programming Environment 115
5.2 C# Program Structure 117
5.3 Run a C# Console Application Program 117
5.4 C# Syntax 118

5.4.1 Arrays and `foreach` loop 119
5.4.2 Command Line Arguments 120
5.4.3 Functions 121
5.5 Examples of Console Application 123
5.6 Windows Forms Application 127
5.7 Examples of Windows Forms Application 130
5.8 Debugging 138

CHAPTER 6 ASP.NET 145
6.1 Introduction to ASP.NET 145
6.2 ASP.NET with VB.NET 146
6.2.1 Structure of ASP.NET Program 147
6.2.2 HTML Controls Versus ASP.NET Web Controls 149
6.2.3 HTML Controls 149
6.2.3.1 Submit Button 150
6.2.3.2 Textbox 150
6.2.3.3 Checkbox 151
6.2.3.4 Radio Button 152
6.2.3.5 Select 153
6.2.4 Web Controls 154
6.2.5 Validation Controls 156
6.2.6 The Code-Behind Programming Framework 157
6.2.7 Server-Side File Processing 159
6.2.8 Accessory Features 162
6.2.8.1 Sending E-mail Message 162
6.2.8.2 Calendar 163
6.2.8.3 *`Redirect`* Method 164
6.2.8.4 Security 166
6.2.9 Web Application Design 168
6.2.10 ADO.NET—Server-Side Database Processing 172
6.2.10.1 Database Connection and SQL in ASP.NET 173
6.2.10.2 Search Database 175
6.2.10.3 Update Database 177
6.2.10.4 Use Data of Database for Decision 177
6.3 ASP.NET with C#.NET 179
6.3.1 C# Programming with ASP.NET Web Controls 179
6.3.2 Code-Behind Programming 184
6.3.3 Server-Side File Processing 185
6.3.4 `<asp:SqlDataSource>` Control for Database Processing 192
6.4 Debugging 195

CHAPTER 7 PHP 201
7.1 Introduction to PHP and PHP Development Environment 201
7.2 Format of PHP Program 202
7.3 Structure of PHP Program 205
7.4 Activate PHP in Web Page and Process Form Data on Server 206
7.5 Programming in PHP 207
7.5.1 PHP Functions 207
7.5.2 `if`-Statement 209
7.5.3 Read Data File from Server 209
7.5.4 `fopen()` and `fclose()` 210

7.5.5 feof() and fgets() 211
7.5.6 while-loop 211
7.5.7 Write Data File to Server and fputs() 211
7.6 Relay Data through Multiple Dynamic Web Pages Using Hidden Fields 212
7.7 Example of Web Application Design 215
7.8 PHP and MySQL Database 219
7.8.1 Set MySQL Database 219
7.8.2 Create and Delete Table in PHP Using SQL 221
7.8.3 Insert Data to Table 222
7.8.4 Access Database 222
7.8.5 Search Database 224
7.8.6 Use ODBC Connection 225
7.9 Debugging 225

CHAPTER 8 XML 229
8.1 Introduction to XML 229
8.1.1 HTML Documents Are Difficult to Process 229
8.1.2 Databases Need Common Data Format to Exchange Data 230
8.2 XML Documents Are Data Sheets 231
8.2.1 XML Instance Documents 231
8.2.2 Declaration 232
8.2.3 Tags and Element 232
8.2.4 Attribute 232
8.2.5 Comment Line and Editorial Style 233
8.3 Cascading Style Sheets 233
8.4 Extensible Style Language 234
8.4.1 <xsl:stylesheet> 235
8.4.2 <xsl:template> 235
8.4.3 HTML Presentation 235
8.4.4 <xsl:value-of> 235
8.4.5 Empty Tag 236
8.4.6 <xsl:for-each> 236
8.5 XML Data Tree 236
8.6 CSS Versus XSLT 237
8.7 Document Type Definition and Validation 239
8.7.1 Simple Example of Internal DTD 240
8.7.2 Simple Example of External DTD 240
8.7.3 <!DOCTYPE> 241
8.7.4 <!ELEMENT> 241
8.7.5 <!ATTLIST> 242
8.7.6 <!ENTITY> 242
8.8 XML Schema 242
8.8.1 Schema Element 243
8.8.2 Data Element, Attribute, and Data Type 244
8.8.3 complexType 244
8.8.4 sequence 244
8.8.5 Cardinality 244
8.8.6 Attribute 244
8.8.7 XML Validation 244
8.9 Summary of Application of XML 245
8.10 An Example of XML Application 246

8.11 Advanced Subjects of XML 251
8.11.1 Conversion of Relational Database into XML Tree 251
8.11.2 xlink and xsl:if 254
8.11.2.1 xlink 259
8.11.2.2 <xsl:if> 260
8.12 XHTML 260
8.13 XBRL 262
8.13.1 Comparison of XBRL with XML 262
8.13.2 Taxonomy 263
8.13.3 Prepare XBRL-Based Reports 263

CHAPTER 9 SQL 267
9.1 Introduction to SQL 267
9.2 CREATE and DROP 267
9.3 INSERT, UPDATE, DELETE 268
9.4 Query—SELECT 269
9.5 WHERE Clause and Comparison 271
9.6 ORDER BY Clause 272
9.7 Aggregate Functions 273
9.8 GROUP BY Clause and HAVING Clause 273
9.9 Joining Tables 274
9.10 Subquery 275
9.10.1 Subquery—Reducing Computational Workload of Join Operation 275
9.10.2 Subquery as an Alternative to GROUP BY 277
9.10.3 Subquery—Determining an Uncertain Criterion 277
9.11 Tactics for Writing Queries 278
9.12 SQL Embedded in Host Computer Programming Languages 278

Preface

There have been critical discussions on the management information systems (MIS) curriculum design during the last several years. The most notable trend in the MIS curriculum renewal movement is to develop more new MIS courses to meet the needs of the job market of MIS graduates. The needs of the job market have considerable implications for the design of MIS courses to educate the next generation of MIS professionals. MIS students must acquire the fundamental theories of MIS as well as the essential practical skills of computer applications to develop the lifelong learning ability in information technology. Technical skills should focus more on problem solving and practical applications. Regardless of changes in the MIS curricula over the past years to meet the requirements of the job market, as well as the requirements of accreditation organizations such as AACSB and ABET, programming remains a core requirement in most MIS programs.

In the modern service-oriented age, development and maintenance of web-based applications still rely heavily on applications of computer languages regardless of the advances of a variety of software packages. To meet the challenges of the ever changing information technologies, educators need to offer courses in important programming languages for their MIS majors. On the other hand, MIS majors cannot afford to learn multiple computer languages on the one-language/one-course basis. The key to the solution to this problem is to make a pedagogical paradigm shift and to develop courses in multiple computer languages.

Few guidelines for MIS courses of computer programming can be found in the literature or on the Internet. The selection of computer languages for programming courses is a crucial task for the pedagogy design. The design components of such courses are based on four considerations. First, the selected computer languages must be representative and should cover essential concepts and features of all kinds of computer languages that are used in business organizations. Second, the selected computer

languages must be commonly used in the industry. Third, the selected computer languages should not require additional computing resources in the ordinary computing labs of the MIS programs. Fourth, the scope and the workload for MIS students to learn these computer languages should be manageable.

Considering these factors, we selected the following computer programming languages for this book: C++, HTML, JavaScript, CSS, VB.NET, C#.NET, ASP.NET, PHP (with MySQL), XML (with XSLT, DTD, and XML Schema), and SQL. Java is a full-scale computer programming language and has been widely used in the industry. This book does not include Java because it requires the Java platform and installation of the Java computing environment on computers with the Windows platform, which could be demanding. In addition, .NET and Java, the two major computer language platforms, share a great similarity of language characteristics. The interested reader who wants to learn Java is referred to our book *Programming Languages for Business Problem Solving,* published by Taylor & Francis, 2007 (ISBN 1-4200-6264-6), for its chapter on Java.

Due to time constraints, it is impossible for students to learn all these languages in great detail. Nevertheless, students are expected to have general knowledge of commonly used computer languages and to be able to develop basic skills of programming. Our methodology applied to the programming courses is to learn languages through typical examples. Specifically, we teach typical problems of MIS applications and their solutions through the use of these computer languages.

A course that uses this book usually consists of two distinct modules: the teaching module and the project module. The teaching module provides an overview of representative computer languages. The project module provides an opportunity for students to practice the computer languages involving hands-on projects. The interested instructor is referred to our pedagogical research papers for the relevant discussions on teaching and learning multiple computer languages in a single course: "An Approach to Teaching Multiple Computer Languages," *Journal of Information Systems Education,* 12(4), 2002, 201–211; and "Design and Delivery of Multiple Server-Side Computer Languages Course," *Journal of Information Systems Education,* 22(2), 2011, 159–168.

The book includes an introduction and eight chapters. The introduction discusses basics of computer languages and the key characteristics of all procedural computer languages. Chapter 2 introduces C++ and explains the fundamental concepts of the two programming paradigms: function oriented and object oriented. Chapter 3 includes HTML, JavaScript, and CSS for web page development. Chapter 4 introduces VB.NET for graphical user interface development. Chapter 5 introduces C#.NET, which is similar to Java. Chapter 6 explains ASP.NET, an important server-side programming language for the Windows platform. ASP.NET incorporates VB.NET, C#.NET, and ADO.NET. Chapter 7 introduces PHP, a popular open source programming language, and explains the use of the MySQL database in PHP. Chapter 8 discusses XML and its companion languages, including XSTL, DTD, and

XML Schema. Finally, Chapter 9 discusses SQL, which is a part of application of server-side programming for database processing.

MIS students will be able to use the concepts and practices in this book as the starting point in their journey to become successful information technology professionals.

Shouhong Wang, PhD
University of Massachusetts, Dartmouth

Hai Wang, PhD
Saint Mary's University, Halifax, Nova Scotia, Canada

The Authors

Hai Wang is an associate professor at the Sobey School of Business at Saint Mary's University, Halifax, Nova Scotia, Canada. He received his BSc in computer science from the University of New Brunswick, and his MSc and PhD in computer science from the University of Toronto. He has published more than 50 research articles in the areas of MIS, big data, data mining, database management, knowledge management, and e-business. His research has continuously been funded by the Natural Sciences and Engineering Research Council of Canada in the past years.

Shouhong Wang is a professor at University of Massachusetts, Dartmouth. He received his PhD in information systems from McMaster University. He has over 30 years' experience of higher education in the MIS field. He has published more than 100 research papers in academic journals and several books on the subject of MIS.

Acknowledgments

Windows, Notepad, WordPad, Windows Explorer, Internet Explorer, Visual Basic, Excel, Access, VB.NET, C#, ASP.NET, Visual Studio.NET, and SQL Server are trademarks of Microsoft Corporation.

Mozilla Firefox is copyrighted by Mozilla Corporation and Mozilla Foundation.

MySQL, Java, and JavaScript are trademarks of Oracle Corporation.

PHP is copyrighted by the PHP Group.

Apache is copyrighted by The Apache Software Foundation.

EasyPHP is copyrighted by EasyPHP and distributed under the general public license.

CSS and XML are trademarks of World Wide Web Consortium (W3C).

Notepad++ is distributed as free software under the GNU general public license.

Dev-C++ is a free integrated development environment developed by Bloodshed Software and distributed under the GNU general public license.

1
Introduction

1.1 Computers

A computer is a general purpose machine that can be programmed to carry out computation and data processing operations. Since programs can be readily changed by humans through programming, the computer can solve a variety of problems. A computer has a central processing unit (CPU), which interprets and executes programs, and primary memory, which stores programs and data. The components of a computer system also include secondary memory, input device, and output device, as shown in Figure 1.1. An input device converts human signals and data into the signals that can be processed by the CPU. The keyboard and mouse are examples of input devices. An output device converts the signals from the CPU into a form understandable to a human. The monitor and printer are examples of output devices. A device, such as the touch screen or network communication device, can be both an input and output device. Similar to a primary memory, a secondary memory can also be used to store programs and data. There are two main differences between primary memory and second memory. First, primary memory is volatile in nature, while secondary memory is nonvolatile. The programs and data that are stored in the primary memory cannot be retained when the power is turned off. A secondary memory can retain the stored programs and data even if the power is turned off. Second, it is much faster for the CPU to access programs and data in the primary memory than in the second memory. The programs or data stored in the secondary memory are read in batches into the primary memory before they are used by the CPU.

1.2 Computer Programming Languages

1.2.1 Role of Computer Programming Language

A computer programming language is an artificial language designed to communicate instructions to a computer. Programming languages are used to create programs that control a computer to perform the tasks as designed. The tasks a computer can carry out include:

- Manipulating data and information
- Reading data from and/or writing data to the secondary memory or other input/output devices
- Presenting data for a human through the user–computer interface

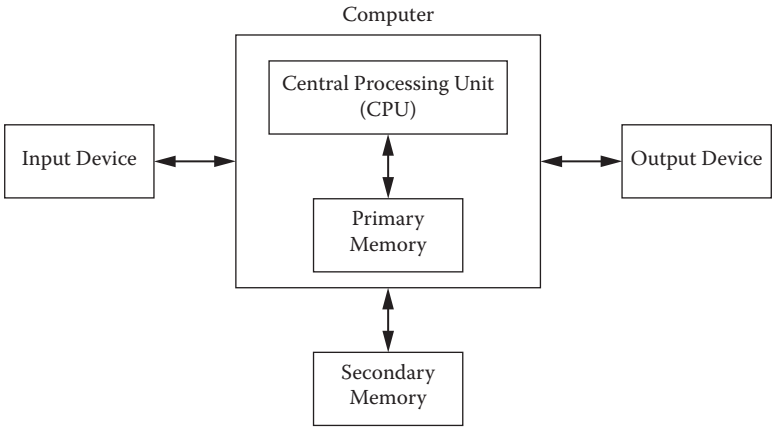


Figure 1.1 A computer system.

There are many computer programming languages. Each computer programming language has its syntax. There is no single computer programming language that can fit all types of applications.

1.2.2 Software Systems

The software systems in a computer are structured in layers, as illustrated in Figure 1.2. As shown in the figure, application software is built by the software developer using high-level programming languages that programmers can easily understand and use. However, the programs in high-level programming languages cannot be executed by the computer unless the programs are translated into the machine executable code (i.e., specific strings of binary digits). To translate a program in a high-level programming language into the machine executable code, a special program, called the compiler or interpreter for that high-level language, must be applied, as shown in Figure 1.3. Once a program in a high-level programming language is translated into the machine-executable code, it can be used an infinite number of times.

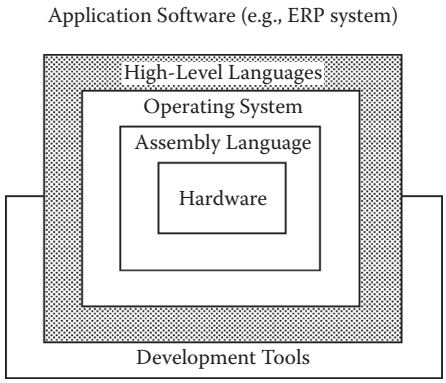


Figure 1.2 The role of computer programming language.

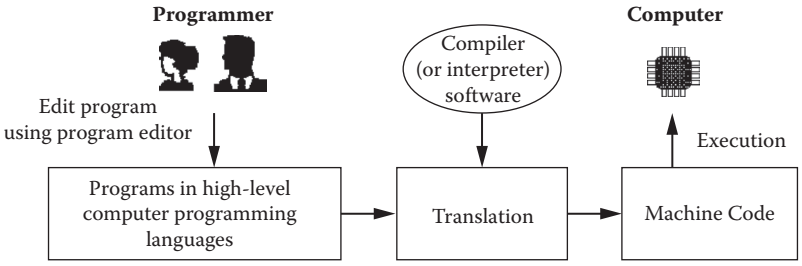


Figure 1.3 Translation of computer programs.

If a program in a high-level programming language has a syntax error, the translation will fail and machine-executable code will not be generated. On the other hand, a program without a syntax error could have a logical error, or semantic error, and the final execution result could be incorrect. To ensure that a program is executed correctly, the computer programmer must do the following three tasks.

1. Understand the application to be developed.
2. Design the program for the application.
3. Debug to fix all syntax errors as well as logical errors.

1.2.3 Taxonomies of Computer Programming Languages

There is no overarching classification scheme for programming languages. In this book, we introduce three major classifications.

1. *Procedural language versus markup language.* A procedural language is capable of commanding a computer to carry out arithmetic or logical operations. All programming languages except for HTML and XML are procedural languages. A markup language is used for annotating a document (or a data set) in a way that is syntactically distinguishable from the text. HTML and XML are markup languages.
2. *Function-oriented language versus object-oriented language.* A function-oriented language uses functions as modules. C is a typical function-oriented language. An object-oriented language uses objects as modules. C++ is a typical object-oriented language. A computer language can be a blended language of function-oriented and object-oriented languages, such as JavaScript and VB.NET.
3. *Client-side language versus server-side language.* A client-side language is used to create the computer programs that are executed on the client side on the web. JavaScript and HTML are typical client-side languages. In contrast, programs in server-side languages such as PHP and ASP.NET are executed by the Web server and have greater access to the information and functional resources available on the server in response to the client's request.

1.3 Computing Architecture in the Internet Environment

Massive client–server networks are connected to build the Internet (or World Wide Web). A general computing architecture in the Internet environment is illustrated in Figure 1.4. Computers are linked to the Internet through the Internet providers.

Client is a computer that accesses a service made available by a server. It is equipped with client-side programs.

Firewall is a computer with special software to protect the Web server, database server, and the database from unauthorized access, viruses, and other suspicious incoming code.

Web server stores the web portal, processes all applications (e.g., order process and payment), and makes all responses to the Internet users' requests. To support applications, a web server has three important software components: API, middleware, and ODBC:

API (*application program interface*) is a set of functions that allow data exchange between the application software and the database.

Middleware is specialized software of server-side programs to access the database.

ODBC (*open database connectivity)* is a software interface to relational databases. On a computer of the Windows platform, you can set ODBC for a particular relational database (e.g., structured query language server) or tabular data (e.g., Excel) through `[Administrative Tools]` in the `[Control Panel]` of `[Settings]` in the Windows operating systems. In the Java platform, JDBC (Java database connectivity) plays a similar role.

Database server is the dedicated server for the data retrieval and maintenance of the database.

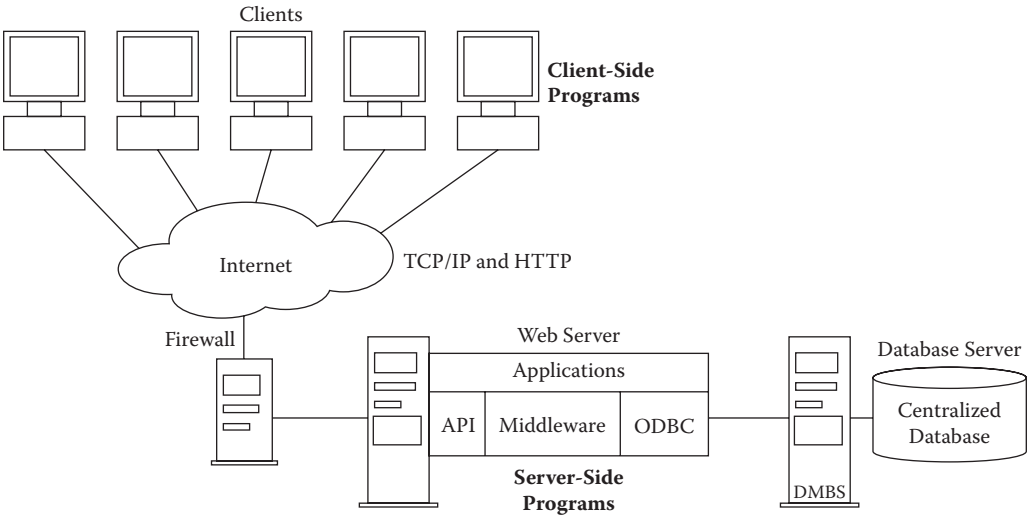


Figure 1.4 Computing architecture in the Internet environment.

1.4 Key Characteristics Shared by All Procedural Programming Languages

As discussed in the previous sections, a procedural programming language is used to carry out arithmetic or logical operations. All procedural programming languages share key characteristics, although individual procedural programming language can have its unique features. Thus, the knowledge of the key characteristics learned from one procedural programming language can be applied to other procedural programming languages.

1.4.1 Syntax, Sentence, and Word

A computer programming language has its syntax—the rules that govern the structure of sentences of the programs written in the language. In a procedural programming language, a sentence consists of words, numbers, and punctuation. There are two types of words in a procedural programming language: keyword (or reserved word) and user-defined word. A keyword represents a specific meaning of the language (e.g., a specific instruction). A user-defined word is defined by the programmer to name a variable or a module. A word used in a procedural programming language must not contain a space and is usually case sensitive.

1.4.2 Variable

A variable is the name of a piece of CPU memory that holds data. A variable name is defined by the programmer and must be a user-defined word. Clearly, variable names are case sensitive; that is, `AVariable` is different from `avariable`. In addition, a name of a variable must be a single user-defined word without a space. A variable has its data type, such as integer, character, etc. The data held by the variable are called the value of the variable. The original value of a variable could be a default value depending on its data type (such as 0 for an integer and space for a character). The value of a variable can be changed through operations, but can never be lost unless the computer program is terminated. Figure 1.5 shows examples of the basic property of variables.

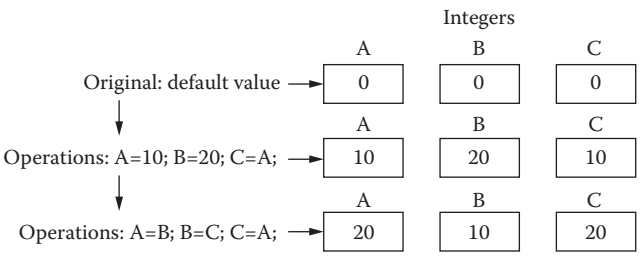


Figure 1.5 Examples of the basic property of variables.

1.4.3 Arithmetic Operation

Arithmetic operations in procedural programming are similar to day-to-day arithmetic calculations, but use reverse expression. For instance, instead of `A+B=C`, `C=A+B` is used in programming; this means: "Let `C` equal to `A` plus `B`." Multiplication is denoted by the asterisk symbol"`*`", and division is denoted by the slash symbol "/". The following are several examples of arithmetic operations:

`A=10`	Let A equal to 10.
`C=A+B`	Let C equal to A plus B.
`B=A*10+(B/10)`	Let B equal to 10 times A plus 1/10 of the original value of B.

1.4.4 Execution Sequence

A computer program consists of a set of instructions. During the execution of the procedure of a program, instructions are executed one after another in a sequence (so-called execution sequence) in which they are encountered, but not in the order in which they are listed in the program. Logical instructions (e.g., if-statement and loops) can control the execution sequence of the program, as explained next.

1.4.5 If-Then-Else Logic

An if-then-else statement controls the computer execution sequence based on a condition that is defined by the current value of a particular variable(s). The if-then-else logic is illustrated in Figure 1.6.

1.4.6 Loop

A loop is a group of instructions that are specified once but are executed several times in succession. A loop statement defines such an iteration procedure, as illustrated

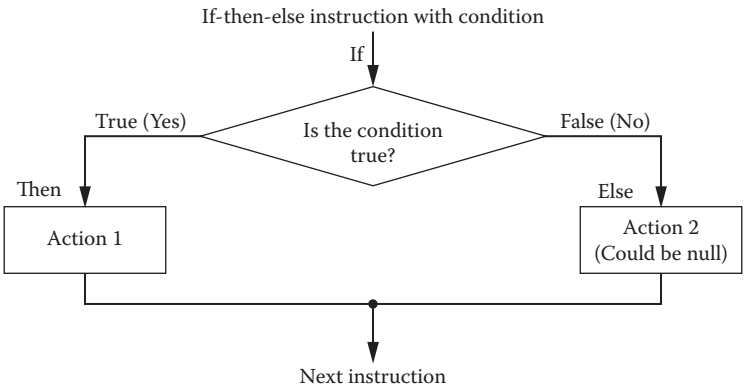


Figure 1.6 If-then-else logic.

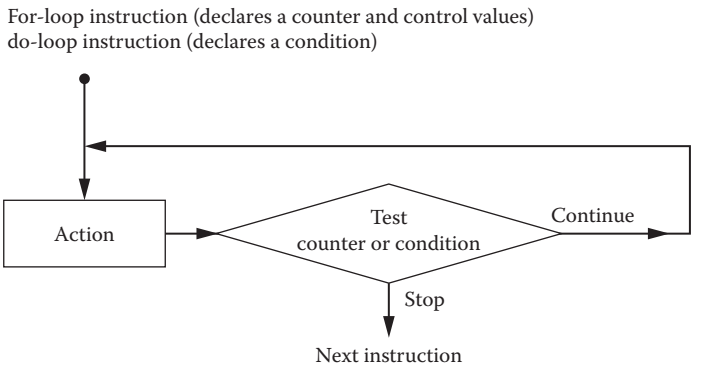


Figure 1.7 Loop.

in Figure 1.7. Loop is actually a variation of if-then-else logic. The common loops include for-loop and do-loop. The variable used in a loop to control the execution of the loop is called a counter.

1.4.7 Module

A large program must be divided into modules to make the program easy to debug. Also, a module can be reused. Here, a module could be a paragraph of instructions, an independent function, or a class, depending upon the specific language in discussion. An instruction in a module can call another module to accomplish a specific task carried out by the called module, as illustrated in Figure 1.8. A module has its name, which is a single user-defined word. The communication between the calling module and the called module can be implemented by passing the values of special variables termed arguments or parameters. Argument and parameter are exchangeable terms in this book.

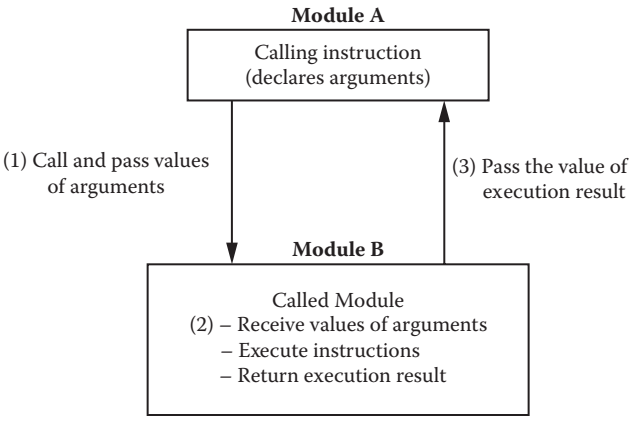


Figure 1.8 Module.

Chapter 1 Exercises

1. Discuss the general model of a computer system. Why does it include secondary memory?
2. Discuss the role of computer programming languages.
3. Discuss how a computer program in a high-level language can be executed by the computer.
4. Discuss the taxonomies of computer programming languages.
5. Discuss the components of computing architecture in the Internet environment.
6. Provide examples of user-defined words that can be used for programs written in a procedural programming language.
7. Suppose that there are two variables: `x` and `y`. `x` stores "`Beer`" and `y` stores "`Water`." How can you swap the values of the two variables to let `x` store "`Water`" and `y` store "`Beer`"?
8. Suppose that there are three variables: `Purchase`, `TaxRate`, and `Payment`. `Purchase` stores the money value of the purchased merchandise, and `TaxRate` stores the state sales tax rate. Write an arithmetic operation to let `Payment` store the payment amount after tax.
9. Write an if-then-else statement using structured English for the GPA scheme: Grade "A" = 4.0 points, grade "B" = 3.0 points, grade "C" = 2.0 points, grade "D" = 1.0 point, and grade "F" = 0 points.
10. Write a loop statement using structured English to let the computer list 0.3, 0.6, 0.9, 1.2, 1.5, …, 30.
11. Discuss the advantages of the use of modules in programming.

2
C++

2.1 Introduction to Function-Oriented and Object-Oriented Programming

In the 1960s and 1970s, the structured program theorem was the main stream of programming methodology. In structured programming, a computer program can be expressed by a computable function or a combination of functions. In this book, the structured program theorem is called function-oriented programming. C is a typical function-oriented programming language.

Object-oriented programming (OOP) was first discussed in the late 1960s by people who were working on the SIMULA language. OOP did not become a popular method until the 1980s. Recently, the object-oriented philosophy has been extended to systems development. The computational environments for networking, multimedia, cloud computing, and mobile computing all require object-oriented systems. C++ is a typical OOP language.

This chapter will explain the basic concepts of function-oriented and object-oriented approaches and provide necessary knowledge of both programming paradigms for students. We will use examples to describe the characteristics of the two programming theorems. Traditionally, C and C++ are two languages, although C++ was migrated from C. Actually, C and C++ share many syntax features. Recently, C++ has become nearly a superset of C. In this chapter of C++, we call a typical function-oriented program a "C program" and a typical objected-oriented program a "C++ program." Since C and C++ languages have been the fundamental computer languages, we believe that the benefit of knowing C and C++ languages would be far beyond what we initially desired. Learning C and C++ together is the best way to gain a comparative view of the two programming theorems. In fact, many commonly used computer programming languages adhere to the concept and the characteristics of C and C++. Many procedural programming languages have blended features of function-oriented and object-oriented programs.

2.2 A Tour of C Language

C is a "mid-level" language. Compared to low-level languages (assembly languages), C programs are easier to write and take fewer instructions. They allow the programmer to take full advantage of the built-in capacities of the computer. Compared to high-level languages (e.g., VB.NET), C programs are more compact and efficient; they

provide the programmer with flexibility in writing a set of programmed instructions at a low level.

Let us examine the style of C program. Suppose we want to display the string "`Hello, World !`" on the screen. The C program could be written as follows:

Listing 2.1: An Example of C Program (`HelloWorld.cpp`)

```
/* C Programming Example */
#include<iostream>
using namespace std;
void main()
{
   printf("Hello, world ! \n");
}
```

We use a Microsoft Visual Studio computing environment to run this program. Start Microsoft Visual Studio. After the start page has been loaded, you may simply close it and start to edit your own program (see Figure 2.1).

Click on `[File]` on the top menu and then `[New Project]`; you will be allowed to create a project. In the `New Project` window, select `[Win32]` on the left pane in `[Visual C++]` and `[Win32 Console Application]` on the right pane. It would be a good practice to choose your own folder (e.g., `F:\Wang`), which will hold your project and the project name (e.g., `C-Project`), which will keep your programs (see Figure 2.2).

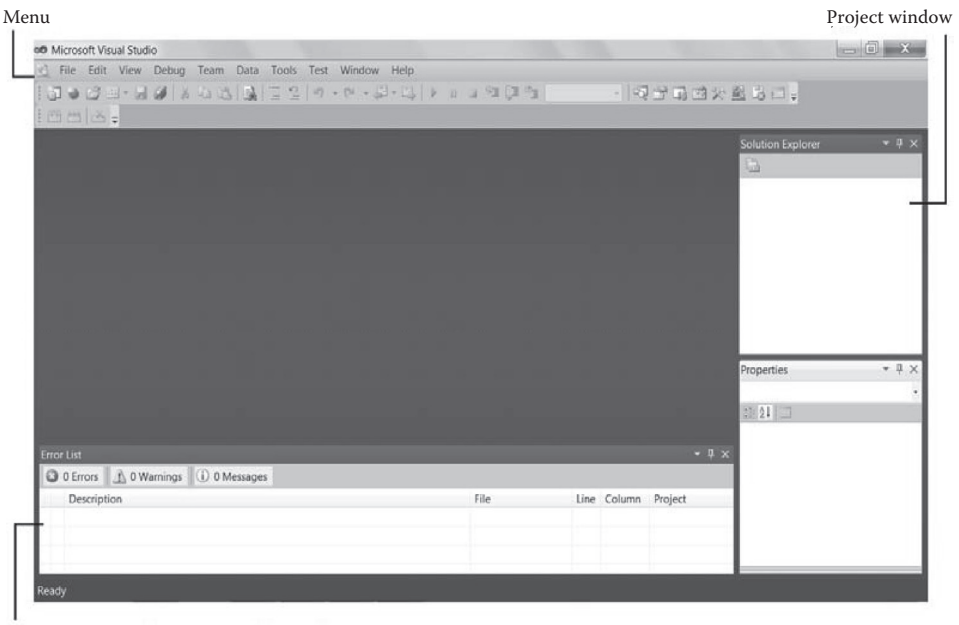


Figure 2.1 Microsoft visual studio environment.

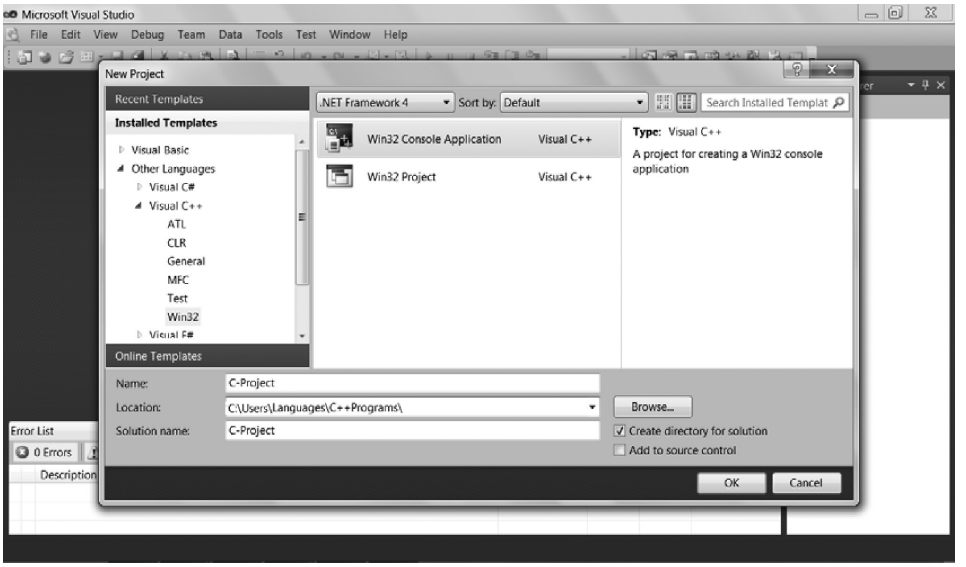


Figure 2.2 Create your project.

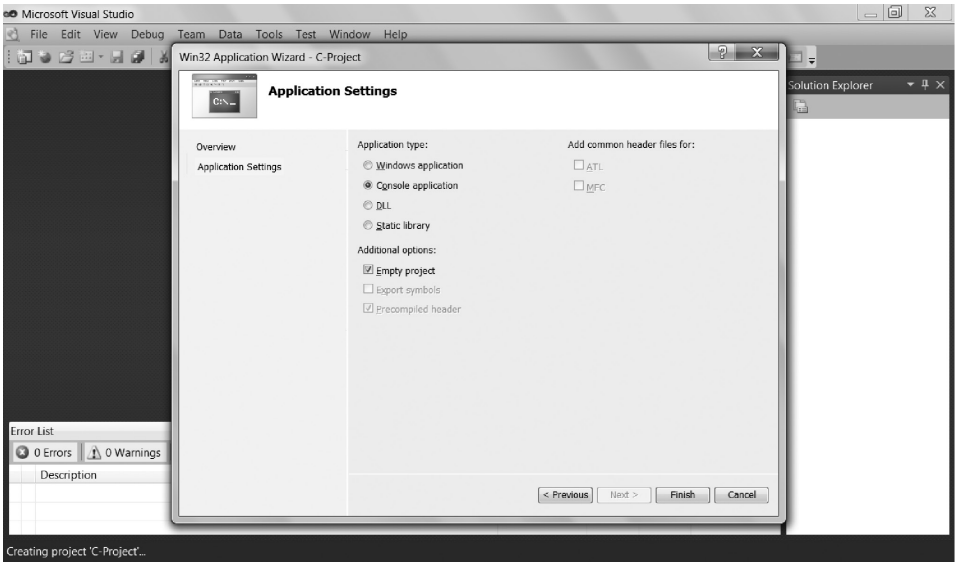


Figure 2.3 Set your project.

Click on [OK] and you will see the Win32 Console Application Wizard window. You choose [Application Setting] on the left pane and select [Empty Project] (see Figure 2.3). Click on [Finish] and the environment creates your project in your folder. Right-click on your project name in the [Solution Explore] pane; you will see a pop-up menu. Select [Add] on the menu and then [New Item] on the second pop-up menu (see Figure 2.4).