Chapter 11
Computer Programming
Section A: Programming Basics

Chapter Preview
After this chapter, you should be able to:
- Describe the role of computer programmers and software engineers
- Categorize today’s popular computer programming languages by generation and paradigm
- Explain the process of planning, coding, and testing a computer program
- Describe the advantages and disadvantages of generic text editors, program editors, and VDEs for coding computer programs
- Define the term algorithm and describe how it relates to procedural programming

Chapter Preview
- Identify various tools that programmers use to plan programs, such as flowcharts, structured English, pseudocode, UML diagrams, and decision tables
- Follow the control structures for a procedural program
- Describe the major concepts of object-oriented programming, such as objects, classes, inheritance, messages, methods, polymorphism, and encapsulation
- Create some basic facts and rules using Prolog syntax
- Describe how a declarative language, such as Prolog, satisfies goals by instantiation

What is program code?
- A computer program is a set of detailed, step-by-step instructions that tell a computer how to solve a problem or carry out a task
- The instructions that make up a computer program are sometimes referred to as code
- Today, program code contains familiar English-like words
- Programs can exist as a single module or multiple modules

What is program code?

Who creates computer programs?
- A computer programmer typically focuses on creating computer programs
- Coding refers to entering the list of commands that become a computer program
- The term computer programming encompasses a broad set of activities that include planning, coding, testing, and documenting computer programs

What is the difference between computer programming and software engineering?
- Software engineering refers to a program development process in which mathematical, engineering, and management techniques are applied to reduce the cost and complexity of a computer program while increasing its reliability and modifiability
- A software engineer designs, codes, tests, and documents software, but tends to focus on designing and testing activities

What is a programming language?
- A programming language is a set of keywords and grammar rules designed for creating instructions that can ultimately be processed, or executed, by the computer
How are programming languages categorized?
- Low-level languages vs. high-level languages
- Generation
- Paradigm

What is a low-level language?
- A low-level language typically includes commands that are specific to a particular CPU or microprocessor family
- Includes machine languages and assembly languages

What is a high-level language?
- A high-level language hides the underlying low-level assembly or machine language, and provides command words and grammar that are more like human languages
  - COBOL
  - BASIC
  - Java
  - C

What was the first generation of computer languages?
- Machine languages were the first languages available for programming computers
  - First-generation languages
- A machine language provides a set of commands, represented as a series of 1s and 0s

What is a second-generation language?
- An assembly language allows programmers to use abbreviated command words, called op codes, rather than 1s and 0s
- Classified as a low-level language
- Assembly languages are most often used to write system software

What is a third-generation language?
- Third-generation languages use easy-to-remember command words to take the place of several lines of assembly language or endless strings of machine language
  - COBOL
  - FORTRAN
  - Pascal
  - C
  - BASIC

What is a fourth-generation language?
- Fourth-generation languages more closely resemble human languages
- Eliminate many of the strict punctuation and grammar rules complicating third-generation languages
- Typically used for database applications
What about fifth-generation languages?
• Some believe a fifth-generation language is a computer programming language based on a declarative programming paradigm
• Other experts feel fifth-generation languages are those that allow programmers to use graphical or visual tools to construct programs

Program Planning: How does a programmer plan a computer program?
• The programming process begins with a problem statement that helps you clearly define the purpose of a computer program

What is a problem statement?
• A problem statement defines certain elements that must be manipulated to achieve a result or goal
  – Specifies any assumptions that define the scope of the problem
  – Clearly specifies the known information
  – Specifies when the problem has been solved

What is an assumption?
• An assumption is something that you accept as true in order to proceed with program planning

How does “known information” apply to a problem statement?
• The known information in a problem statement is the information that you supply to the computer to help it solve a problem

Program Coding: How do I code a computer program?
• It depends on the programming language you use, the programming tools that you selected, and the programming paradigm that best fits the problem you’re trying to solve

What is a generic text editor?

What is a program editor?
• A program editor is a type of text editor specially designed for entering code for computer programs

What is a VDE?

How does a programmer know if a program works?
• A computer program must be tested to ensure that it works correctly
• Testing consists of running the program and entering test data to see whether the program produces the correct results

What can cause program errors?
• A syntax error occurs when an instruction does not follow the syntax rules, or grammar, of the programming language
  – Omitting punctuation or misspelling a keyword
• A runtime error occurs when the computer can’t execute an instruction
  – Divide by 0
• A logic error is an error in the logic or design of a program
  – Incorrect formula

28 How do programmers find errors?
• Proofreading lines of code
• Programmers can also use a debugger, which allows programmers to step through a program and monitor the status of variables, input, and output

29 Do computer programs contain any special documentation?

30 Chapter 11
  Computer Programming
  Section B: Procedural Programming

31 Algorithms: What is an algorithm?
• An algorithm is a set of steps for carrying out a task, which can be written down and implemented
• Carefully following the steps of a correctly formulated algorithm guarantees that you can accomplish the task for which the algorithm was designed

32 How do I write an algorithm?
• Typically written in a format that is not specific to a particular programming language
• This approach allows the programmer to focus on correctly formulating the algorithm, without being distracted by the detailed syntax of a computer programming language

33 How do I figure out an algorithm?

34 What’s the best way to express an algorithm?
• Structured English – a subset of the English language with a limited selection of sentence structures that reflect processing activities
• Pseudocode – a notation system for algorithms that has been described as “a mixture of English and your favorite programming language”
• Flowcharts – a graphical representation of the way that a computer should progress from one instruction to the next when it performs a task

35

36 How do I know if my algorithm is correct?
• Before finalizing the algorithm, perform a walkthrough to verify that your algorithm works
  – Step through the problem using realistic test data

37 How do I specify the order in which program instructions are performed by the computer?
• During sequential execution, the computer performs each instruction in the
order it appears
– Normal pattern of program execution

Print “This is the first line.”
Print “This is the second line.”

38 Is there an alternative to sequential execution?

39 Can the computer make decisions while it executes a program?
   • A selection control structure (decision structure or branch) tells a computer what to do, based on whether a condition is true or false

40 Can a computer automatically repeat a series of instructions?
   • A repetition control structure directs the computer to repeat one or more instructions until a certain condition is met

41 What kinds of problems are best suited to the procedural approach?
   • Problems that can be solved by following a step-by-step algorithm
   • Transaction processing
     – Characterized by the use of a single algorithm applied to many different sets of data

42 What are the advantages and disadvantages of the procedural paradigm?
   • This approach tends to produce programs that run quickly and use system resources efficiently
   • Does not fit gracefully with certain types of problems
     – Unstructured problems
     – Complex algorithms
   • Forces programmers to view problems as a series of steps

43 Chapter 11
   Computer Programming
   Section C: Object-Oriented Programming

44 What is the basic focus of the object-oriented paradigm?
   • The object-oriented paradigm is based on the idea that the solution for a problem can be visualized in terms of objects that interact with each other
   • An object is a unit of data that represents an abstract or real-world entity, such as a person, place, or thing

45 What’s the difference between an object and a class?
   • A class is a template for a group of objects with similar characteristics

46 Inheritance: How flexible are classes for defining different
types of objects?
- The object-oriented paradigm endows classes with quite a bit of flexibility
- An OO feature called “inheritance” provides flexibility to deal with objects’ unique characteristics

What is inheritance?
- *Inheritance* refers to passing certain characteristics from one class to other classes
- The process of producing new classes with inherited attributes creates a *superclass* and *subclasses*

Methods and Messages: How does an OO program use objects?
- Programs use objects to manipulate them with methods
- A *method* is a segment of code that defines an action

What can a method do?
- Collect input
- Perform calculations
- Make comparisons
- Execute decisions
- Produce output

What activates a method?
- A method is activated by a *message*, which is included as a line of program code, sometimes called a "call"

How do methods relate to classes?
- Methods can be defined along with the class they affect

What is polymorphism?
- Polymorphism, sometimes called “overloading,” is the ability to redefine a method in a subclass

How does a Java program begin?
- The computer begins executing a Java program by locating a standard method called `main()`, which contains code to send messages to objects by calling methods

What happens when the completed program runs?

How did object-oriented languages originate?
- Computer historians believe that SIMULA, developed in 1962, was the first computer language to work with objects, classes, inheritance, and methods
- The second major development in object-oriented languages came in 1972 when Alan Kaye began work on the Dynabook project

Which object-oriented languages are popular today?
- Ada95
- C++
• Visual Basic
• C#
• Java

57 What are the advantages and disadvantages of the OO paradigm?
• Programmers may be able to visualize the solutions to problems more easily
• Encapsulation allows objects to be adapted and reused in a variety of different programs
  – Encapsulation refers to the process of hiding the internal details of objects and their methods
• OO programs tend to require more memory and processing resources than procedural programs

58 Chapter 11
Computer Programming
Section D: Declarative Programming

59 What is unique about the declarative paradigm?
• Describes aspects of a problem that lead to a solution
• Focuses on describing the problem

60 What are the building blocks for the declarative paradigm?
• Use a collection of facts and rules to describe a problem
  – A fact is a statement that provides the computer with basic information for solving a problem

61 How does a programmer plan a declarative program?
• The core of most declarative programs is a set of facts and rules that describe a problem
• A decision table is a tabular method for visualizing and specifying rules based on multiple factors

62 Prolog Facts: How does a programmer code facts?
• An argument represents one of the main subjects that a fact describes
• The predicate describes the relationship between the arguments

63 What is a goal?

64 What is instantiation?

65 Prolog Rules: How does a programmer code Prolog rules?

66 Does the order of rules affect the way a Prolog program works?
• The order or sequence of rules in a Prolog program is not critical

67 What does a complete program look like in Prolog?

68 Input Capabilities: How do I collect input from the user?
69 **How can I add facts at runtime?**

- When the program runs, you can use Prolog’s built-in asserta predicate to temporarily add facts to the program

```prolog
asserta(priceof(pizza1,10))
```

70 **Debugging with Trace: How do I test a Prolog program?**

- Most Prolog compilers provide a *trace feature* that allows you to track through each instantiation

71 **What kinds of problems are suitable for the declarative approach?**

- Databases that contain complex relationships
- Decision support systems that handle semi-structured problems
- Expert systems that require analysis of multiple, interrelated factors

72 **What are the advantages and disadvantages of declarative languages?**

- Provide a great deal of flexibility for querying a set of facts and rules
- Not very popular and not widely used for production applications
- Provide minimal input and output capabilities
- Relatively poor performance on today’s personal computer architecture

73 **Conclusion**

You should now be able to:
- Describe the role of computer programmers and software engineers
- Categorize today’s popular computer programming languages by generation and paradigm
- Explain the process of planning, coding, and testing a computer program
- Describe the advantages and disadvantages of generic text editors, program editors, and VDEs for coding computer programs
- Define the term algorithm and describe how it relates to procedural programming

74 **Conclusion**

- Identify various tools that programmers use to plan programs, such as flowcharts, structured English, pseudocode, UML diagrams, and decision tables
- Follow the control structures for a procedural program
- Describe the major concepts of object-oriented programming, such as objects, classes, inheritance, messages, methods, polymorphism, and encapsulation
- Create some basic facts and rules using Prolog syntax
- Describe how a declarative language, such as Prolog, satisfies goals by instantiation