NEW PERSPECTIVES

Chapter 12
Computer Programming

Computer Concepts 2014

12 Chapter Contents

- Section A: Programming Basics
- Section B: Procedural Programming
- Section C: Object-Oriented Programming
- Section D: Declarative Programming
- Section E: Secure Programming

12 FastPoll True/False Questions
Answer A for True and B for False

- 120100 A line of program code typically contains a keyword or command.
- 120200 BASIC, COBOL, and C are classified as third-generation languages.
- 120300 Programming paradigms include FORTRAN and Ada.
- 120400 In a program, a variable represents a value that can change.
- 120500 VDE is an example of an object-oriented programming language.
- 120600 A programmer who omits a command word from a line of code has made a logic error.

12 FastPoll True/False Questions
Answer A for True and B for False

- 120700 Programmers use a tool called an errata to step through a program to locate syntax errors.
- 120800 Pseudocode is a bug or error in a line of program code.
- 120900 A control structure specifies the sequence in which a program is executed.
- 121000 FOR…NEXT and DO…WHILE are examples of commands for loops.
- 121100 A programmer could define a class called “pizza” to solve the pizza problem using object-oriented programming.
- 121200 Inheritance, methods, messages, and polymorphism are associated with the declarative paradigm.

12 FastPoll True/False Questions
Answer A for True and B for False

- 121300 Goals, rules, and instantiation are associated with the agile paradigm.
- 121400 Java is a declarative programming language.
- 121500 Prolog facts contain an argument and a predicate.
- 121600 Buffer overflows are associated with security vulnerabilities.
- 121700 Programmers can use threat modeling and formal methods to create more secure programs.

12 Section A: Programming Basics

- Computer Programming and Software Engineering
- Programming Languages and Paradigms
- Program Planning
- Program Coding
- Program Testing and Documentation
- Programming Tools

12 Question

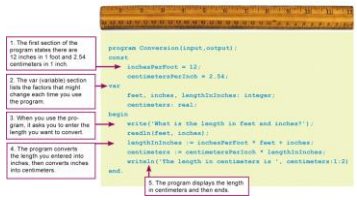
- 122100 Computer programming languages have evolved through several generations. Experts are not in agreement about what constitutes a fifth-generation programming language. What is the controversy?
  - A. Some experts believe that assembly languages should be included, whereas other experts do not.
  - B. Some experts believe declarative languages are fifth-generation languages, whereas other experts believe that fifth-generation languages are those that allow programmers to use graphical tools to construct programs.
  - C. Most experts believe that languages like C, BASIC, and Java are fifth-generation languages, but programmers disagree because those languages follow the procedural paradigm.
  - D. A few experts don't believe there is a fifth-generation of programming languages, but most experts think that Japanese computer scientists invented fifth-generation languages when they produced C++.

12 Computer Programming and Software Engineering

- The instructions that make up a computer program are sometimes referred to as code
- Programs can have millions of lines of code
  - Developed by computer programmers
    - Computer programming

12 Computer Programming and Software Engineering

FIGURE 12-1
A typical computer program consists of lines of code that tell a computer how to solve a problem or carry out a task. This program is written in a computer programming language called Pascal.



12 Programming Languages and Paradigms

- Programming languages are made up of keywords and grammar rules designed for creating computer instructions
  - Keywords can be combined with specific parameters
- Low-level languages typically include commands specific to a particular CPU or microprocessor family
- High-level languages use command words and grammar based on human languages

12 Programming Languages and Paradigms

- First-generation languages
  - Machine language
- Second-generation languages
  - Assembly language
- Third-generation languages
  - Easy-to-remember command words

12 Programming Languages and Paradigms

- Fourth-generation languages
  - More closely resembles human language
- Fifth-generation languages
  - Based on a declarative programming paradigm
- The programming paradigm refers to a way of conceptualizing and structuring the tasks a computer performs

12 Programming Languages and Paradigms

FIGURE 12-8
Programming Paradigms

Paradigm	Languages	Description
Event-driven	Visual Basic, C#	Focuses on selecting user interface elements and defining event-handling routines that are triggered by various mouse or keyboard activities
Procedural	BASIC, Pascal, COBOL, Fortran, Ada	Emphasizes linear steps that provide the computer with instructions on how to solve a problem or carry out a task
Object-oriented	Smalltalk, C++, Java, Scratch	Formulates programs as a series of objects and methods that interact to perform a specific task
Declarative	Prolog	Focuses on the use of facts and rules to describe a problem

12 Program Planning

- The problem statement defines certain elements that must be manipulated to achieve a result or goal
- You accept assumptions as true to proceed with program planning
- Known information helps the computer to solve a problem
- Variables vs. constants

12 Program Planning

- Problem statement:

FIGURE 12-10
Pizza Problem Statement

Assuming that there are two pizzas to compare, that both pizzas contain the same toppings, and that the pizzas could be round or square, and given the prices, shapes, and sizes of the two pizzas, the computer will print a message indicating which pizza has the lower price per square inch.

12 Program Coding

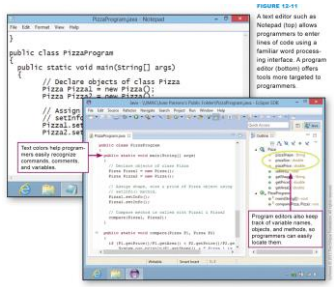


FIGURE 12-11
A text editor such as Notepad (top) allows programmers to enter lines of code using a familiar word processing interface. A program editor (bottom) offers tools more targeted to programmers.

12 Program Coding

- A VDE (visual development environment) provides programmers with tools to build substantial sections of a program
  - Form design grid
  - Control
  - Properties
  - Event
  - Event-handling code

FIGURE 12-12
A form design grid is an important part of a VDE. This form was designed for the pizza program using Visual Basic.

12 Program Coding

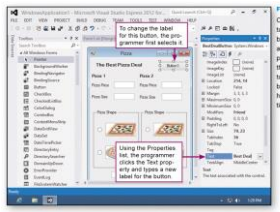
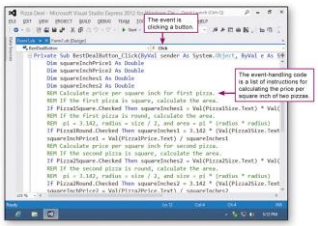


FIGURE 12-13
Controls, such as buttons, can be selected by a programmer from a properties list. Here, a programmer is changing the text property of a button so that its label will be "Best Deal". Learn how to work with properties in a VDE.

12 Program Coding

FIGURE 12-14
The 'Best Deal' event-handling code tells the computer what to do when users click the Best Deal button.



12 Program Testing and Documentation

- A computer program must be tested to ensure that it works correctly
- Program errors include:
  - Syntax errors
  - Runtime errors
  - Logic errors
- A debugger can help a programmer read through lines of code and solve problems

12 Program Testing and Documentation

- Remarks or "comments" are a form of documentation that programmers insert into the program code

FIGURE 12-16
A series of remarks in a BASIC program can explain to programmers the method used to calculate the square inches in a round pizza.

```
Rem The program calculates the number of square inches
Rem in a round pizza using the formula pi r squared
Rem pi = 3.142, size / 2 = radius,
Rem and (size / 2) ^2 = radius squared
Rem SquareInches = 3.142 * (size / 2) ^2
```


12 Programming Tools

- An SDK (software development kit) is a collection of language-specific programming tools that enables a programmer to develop applications for a specific computer platform
- An IDE (integrated development environment) is a type of SDK that packages a set of development tools into a sleek programming application

12 Programming Tools

- A component is a prewritten module, typically designed to accomplish a specific task
- An API is a set of application program or operating system functions that programmers can access from within the programs they create
- C, Java, and C++ are the most popular programming languages
- Microsoft's XNA framework is a set of tools for creating Xbox 360 games
- Objective-C is popular for creating apps for iPhones and iPads

12 Section B: Procedural Programming

- Algorithms
- Expressing an Algorithm
- Sequence, Selection, and Repetition Controls
- Procedural Languages and Applications

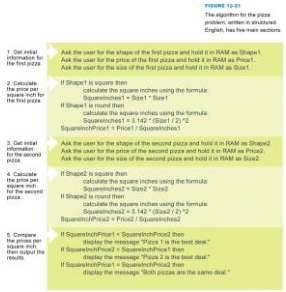
12 Question

- 122200 Procedural programs are based on a step-by-step algorithm. How do programmers devise the algorithms for their programs?
  - A. They create objects, classes, and methods, and then figure out the step-by-step way to send messages back and forth between them.
  - B. They look at APIs and VDEs, which offer templates for common program functions.
  - C. They think about how a task might be carried out manually and devise flowcharts, structured English, or pseudocode to describe the steps.
  - D. They first devise facts about the problem, then they come up with the steps based on rules.

12 Algorithms

- Set of steps for carrying out a task that can be written down and implemented
- Start by recording the steps you take to solve the problem manually
- Specify how to manipulate information
- Specify what the algorithm should display as a solution

12 Algorithms

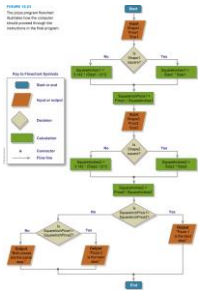


12 Expressing an Algorithm

- Structured English
- Pseudocode

12 Expressing an Algorithm

- Flowchart



12 Expressing an Algorithm

- Perform a walkthrough to verify that your algorithm works

12 Sequence, Selection, and Repetition Controls

- Sequence control structure

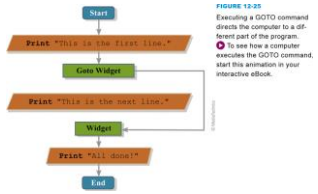
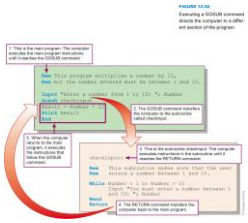


FIGURE 12-25
Executing a GOTO command directs the computer to a different part of the program. To see how a computer executes the GOTO command, start this animation in your interactive eBook.

12 Sequence, Selection, and Repetition Controls

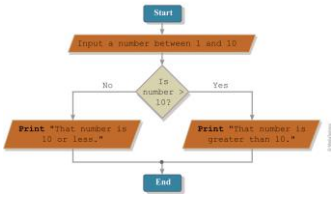
- Subroutines, procedures, and functions are sections of code that are part of the program, but not included in the main sequential execution path



12 Sequence, Selection, and Repetition Controls

- Selection control structure

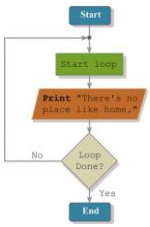
FIGURE 12-27
On a flowchart, a diamond shape indicates a decision. Step through a decision structure using your interactive eBook.



12 Sequence, Selection, and Repetition Controls

- Repetition control structure

FIGURE 12-28
To execute a loop, the computer repeats one or more commands until some condition indicates that the looping should stop. Remember Dorothy in *The Wizard of Oz*? Watch how a computer would execute her "There's no place like home." loop.



12 Procedural Languages and Applications

- Popular procedural languages: COBOL, FORTH, APL, ALGOL, PL/1, Pascal, C, Ada, and BASIC
- The procedural approach is best for problems that can be solved by following a step-by-step algorithm
- Produces programs that run quickly and use system resources efficiently

12 Section C: Object-Oriented Programming

- Objects and Classes
- Inheritance
- Methods and Messages
- Object-oriented Program Structure
- Object-oriented Languages and Applications

12 Question

- 122300 Object-oriented programming has become quite popular. Why?
  - A. It allows programmers to structure problems in a cognitively similar way as they perceive the real world.
  - B. Object-oriented programs are the fastest, most efficient type of programs for today's computer hardware.
  - C. It creates the most secure programs, with the fewest security holes.
  - D. It is the best programming paradigm for working with words and concepts.

12 Objects and Classes

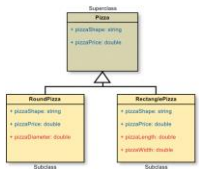
- An object represents an abstract or real-world entity
- A class is a template for a group of objects with similar characteristics
  - A class attribute defines the characteristics of a set of objects
    - Public vs. private attributes

FIGURE 12-30
A class, such as the Pizza class, is a general template for a group of objects with similar characteristics.

12 Inheritance

- Passing certain characteristics from one class to other classes
  - Class hierarchy
    - Superclass
    - Subclass

FIGURE 12-33
The two subclasses inherit the attributes shown in blue text (pizzaShape and pizzaPrice) from the Pizza superclass. The attributes in red are unique to the subclasses. The plus sign indicates that these inherited attributes are public.



12 Methods and Messages

- A method is a segment of code that defines an action
  - Collect input, perform calculations, etc.
  - A method is activated by a message
  - Can be defined along with the class they affect
- Polymorphism refers to the ability to redefine a method in a subclass
  - Helps simplify program code

12 Object-Oriented Program Structure

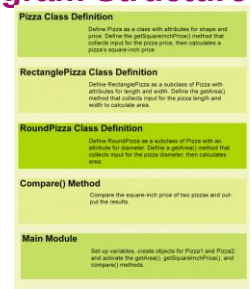


FIGURE 12-39
Program Structure for the Pizza Program

12 Object-Oriented Program Structure

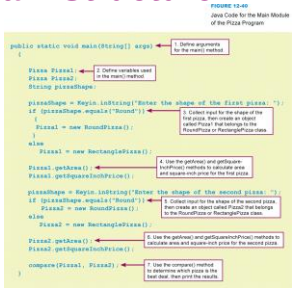


FIGURE 12-40
Java Code for the Main Module of the Pizza Program

12 Object-Oriented Program Structure

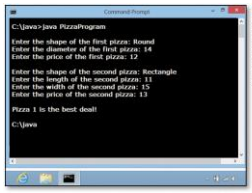


FIGURE 12-41
When the pizza program runs, on-screen prompts ask for the shape, size, and price of each pizza; then the program displays a message that indicates which pizza is the best deal. Watch the program run in your interactive eBook.

12 Object-Oriented Languages and Applications

- SIMULA was believed to be the first object-oriented computer language
- The Dynabook project was the second major development in object-oriented languages
- Popular hybrid languages today are Ada 2005, C++, Visual Basic, Objective-C, and C# and include both procedural and object-oriented techniques
- Facets of the object-oriented paradigm can also increase a programmer's efficiency because encapsulation allows objects to be adapted and reused in a variety of different programs

12 Section D: Declarative Programming

- The Declarative Paradigm
- Prolog Facts
- Prolog Rules
- Input Capabilities
- Declarative Languages and Applications

12 Question

- 122400 Declarative languages, such as Prolog, are very powerful for programs that involve words, concepts, and complex logic, but why aren't these languages a first choice for programming computer games?
  - A. They don't execute as fast as programs written with procedural languages.
  - B. They are too difficult to learn.
  - C. They have too many security holes.
  - D. They require expensive compilers.

12 The Declarative Paradigm

- Describes aspects of a problem that lead to a solution
  - A fact is a statement for solving a problem
  - Rules describe the relationship between facts

12 The Declarative Paradigm

- A decision table is a tabular method for visualizing and specifying rules based on multiple factors

FIGURE 12-43
Decision Table

Lowest price?	Y	N	Y	N	Y	N	Y	N
Delivery available?	Y	Y	N	N	Y	Y	N	N
Ready in less than 30 minutes?	Y	Y	Y	Y	N	N	N	N
Buy it?	Y	Y	N	N	Y	N	N	N

12 Prolog Facts

FIGURE 12-44
A Prolog fact follows specific syntax rules.

12 Prolog Facts

- You can query a Prolog program's database by asking a question, called a goal

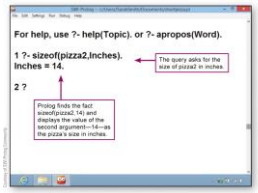


FIGURE 12-45
The ?- prompt allows you to work with a knowledge base by entering facts, rules, and goals. Suppose you have opened a knowledge base containing facts about two pizzas. You can enter goals to find their prices, sizes, and shapes. Go step-by-step through some goals for learning about the pizzas in this knowledge base.

12 Prolog Facts

- Finding a value for a variable is referred to as instantiation

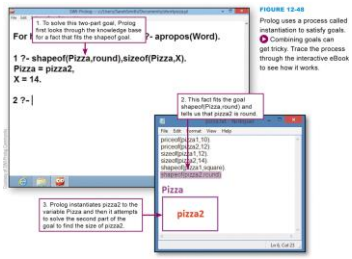


FIGURE 12-48
Prolog uses a process called instantiation to satisfy goals. Combining goals can get tricky. Trace the process through the interactive eBook to see how it works.

12 Prolog Rules

- The order of program instructions is critically important

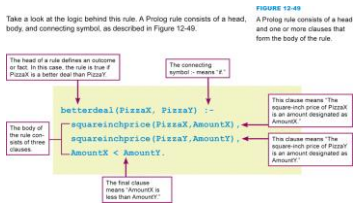


FIGURE 12-49
A Prolog rule consists of a head and one or more clauses that form the body of the rule.

12 Input Capabilities

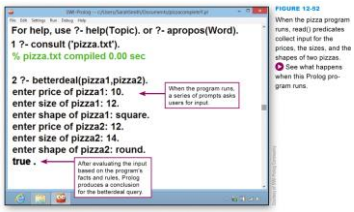


FIGURE 12-52
When the pizza program runs, read() predicates collect input for the prices, the sizes, and the shapes of two pizzas. See what happens when this Prolog program runs.

12 Declarative Languages and Applications

- Declarative programming languages are most suitable for problems that involve words, concepts, and complex logic
  - Highly effective programming environment
  - Not commonly used for production applications
  - Minimal input and output capabilities

12

Section E: Secure Programming

- Black Hat Exploits
- Secure Software Development
- Mitigation

12

Question

- 122500 Consumers are told to use security software because their computers are vulnerable to security exploits, but what is the source of security vulnerabilities?
  - A. Most security vulnerabilities are the fault of the user.
  - B. Threat modeling causes many of the vulnerabilities in today's software.
  - C. Faulty programming that allows buffer overflows is one of the main causes of security vulnerabilities.
  - D. Operating system patches and DREAD categories are the source of the security vulnerabilities that affect most consumers.

12

Black Hat Exploits

- Viruses, worms, bots, malicious Web scripts, and other exploits creep into computer systems
  - Black-hat exploits
- A buffer overflow (also called a buffer overrun) is a condition in which data in memory exceeds its expected boundaries and flows into memory areas intended for use by other data

12

Black Hat Exploits

FIGURE 12-54
A buffer overflow allows attackers to surreptitiously change the way a program works.

1. The programmer expects to collect a 2-byte state abbreviation as input for variable A. Variable B holds the address of the next instruction.

Variable A		Variable B	
		0	1	9

2. An attacker enters MI999. The 5-byte string overflows the buffer for variable A and spills over into variable B. Now variable B holds 999, which is not the correct address for the next instruction.

Variable A		Variable B		
M	I	9	9	9

3. When the program checks variable B to locate the next instruction, it goes to 999, where an attacker might have stored code for a virus, worm, bot, or malicious HTML script.

12

Black Hat Exploits

- Error messages can help programmers locate the source of errors if they contain information pertinent to the location of defective code and the state of variables

FIGURE 12-55
This verbose error message provides attackers with information about restricted access folder names and locations, as well as account names (Manager, Member, and Owner) that can gain access to these folders.

12

Secure Software Development

- Most software security problems can be traced back to defects that programmers unintentionally introduce in software during design and development
- Formal methods help programmers apply rigorous logical and mathematical models to software design, coding, testing, and verification
- Threat modeling (risk analysis)

12 Secure Software Development

Spoofing: Pretending to be someone else
Tampering: Changing, adding, or deleting data
Repudiation: Covering tracks to make attacks difficult to trace
Information disclosure: Gaining unauthorized access to information
Denial of service: Making a system unavailable to legitimate users
Elevation of privilege: Modifying user rights to gain access to data

FIGURE 12-56
STRIDE categories help software developers anticipate threats from attackers.

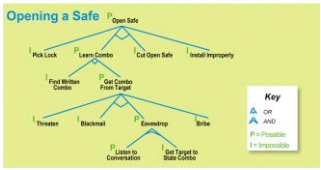
Damage: How much damage can a particular attack cause?
Reproduce: Is this attack easy to reproduce?
Exploit: How much skill is needed to launch the attack?
Affected: How many users would be affected by an attack?
Discovered: How likely is it that this attack would be discovered?

FIGURE 12-57
DREAD categories help software developers gauge the severity of threats.

12 Secure Software Development

- An attack tree is a hierarchical diagram of potential attacks against a system

FIGURE 12-58
Attack trees are used by security analysts in many fields. This diagram illustrates an attack tree that might be devised by a bank security officer examining the vulnerability of the bank vault.



12 Secure Software Development

- Defensive programming (also referred to as secure programming) is an approach to software development in which programmers anticipate what might go wrong as their programs run and take steps to smoothly handle those situations
  - Source code walkthroughs
  - Simplification
  - Filtering input

12 Secure Software Development

- Signed code is a software program that identifies its source and carries a digital certificate attesting to its authenticity

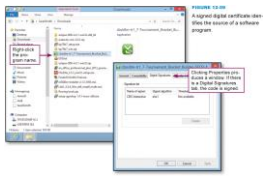


FIGURE 12-59
A signed digital certificate identifies the source of a software program.

12 Mitigation

- Despite defensive programming and other tactics to produce secure software, some defects inevitably remain undiscovered in products that end up in the hands of consumers
- When bugs are discovered, the programmer's remaining line of defense is to produce a bug fix or patch

12 Mitigation

- Take the following steps to avoid security problems that stem from software defects:
  - Select applications from software publishers with a good security track record
  - Read reviews of products before you download them
  - Watch for patches and apply them
  - Consider using open source software, which has been extensively reviewed by the programming community
  - Keep your firewall and antivirus software deployed and up to date

12

What Do You Think?

- 123100 Have you played violent videogames?
  - A. Yes B. No C. Not sure
- 123200 Do you believe that violent videogames contribute to teen violence?
  - A. Yes B. No C. Not sure
- 123300 Do you think that states might be able to craft legislation that limits violent videogames without eroding principles of free speech?
  - A. Yes B. No C. Not sure

NEW PERSPECTIVES

Chapter 12 Complete

Computer Concepts 2014