Chapter 3

Understanding Structure

At a Glance

• Overview
• Objectives
• Quick Quizzes
• Additional Resources
• Key Terms
Overview

Chapter 3 provides an introduction to structure in programming. Students will learn about the three basic structures of programming. They will also be able to distinguish between unstructured spaghetti code and structured code and learn how to untangle spaghetti code. Finally, students will also learn how to stack and nest structures and how to perform a priming read.

Chapter Objectives

In this chapter, you will learn about:

- The features of unstructured spaghetti code
- The three basic structures—sequence, selection, and loop
- Using a priming input to structure a program
- The need for structure
- Recognizing structure
- Structuring and modularizing unstructured logic

Understanding Unstructured Spaghetti Code

1. Understand the concept of spaghetti code. Most programs written in the early days of programming exhibited this style to some extent because there were no guidelines on program design.

2. Define the terms unstructured and structured and note that structured programs follow certain rules that will be covered in this chapter.

3. Understand the dog washing procedure documented in Figure 3-1.

Understanding the Three Basic Structures: Sequence, Selection, and Loop

1. Understand the concept of a structure and observe that any program can be constructed from these three types of structures.
2. Understand the **sequence structure**, using Figure 3-2 for illustration.

3. Understand the **selection structure** and describe its common implementation as the **if-then-else** statement. Use Figures 3-3 and 3-4 for illustration.

   The selection structure must ask a question that can be answered only with a Yes or No.

4. Explain the difference between a **dual alternative if** statement and a **single alternative if** statement. Explain the **null case** situation.

5. Understand the **loop structure**. The loop structure is the structure that provides the ability to perform repetitive processing on a very large scale, which is one task that computers excel at. Explain that looping is often referred to as repetition or iteration.

6. Describe the use of the **while...do** structure, otherwise known as the **while loop**.

7. All programs can be written by simply combining these three structures in an infinite number of ways.

8. Describe the **stacking** of structures by using the single entry and exit points on each structure, using Figure 3-6 for illustration.

9. Describe the end-structure statements, and discuss how these statements help to delineate a block of code that comprises a structure.

   Structures may only be stacked end to end. Students commonly make the mistake of interweaving structures instead of stacking them.

10. Understand the **nesting** of structures, using Figures 3-7 through 3-9 for illustration.

11. Understand the use of indentation in pseudocode to indicate nesting.

   The nested structures must be totally contained within the outer structure.

12. Understand the concept of statement **blocks**.

   Some languages have block delineation symbols. For example, Java uses curly braces to identify blocks.

13. All programs can be written by simply combining these three structures in an infinite number of ways.
There are an infinite number of ways that structures may be combined, stacked, and nested.

**Quick Quiz 1**

1. What are three structures of programming?
   Answer: sequence, selection, and loop

2. When structures are attached to each other end to end, this is called _______.
   Answer: stacking

3. When one structure is contained completely within another structure, this is called _____.
   Answer: nesting

4. A decision, or selection, structure, allows how many possible courses of actions?
   Answer: two

**Using a Priming Input to Structure a Program**

1. Walk through the example on pages 103-107 (and Figures 3-12 through 3-16) so that you can understand why it is necessary to read the first value before entering a loop.

2. Describe the use of a **priming read**, explaining when and why it is used.

   The main purpose of a priming read is to allow the processing block that follows it to be structured.

3. Understand the importance of keeping the code structured by reviewing flowcharts 3-17 and 3-18.

   Most of the flowcharts in this section are examples of what NOT to do. Take time to carefully explain why these flowcharts represent the incorrect way to design the number doubling program, as it may not be obvious to students new to programming.

   Review how pseudocode is represented with flowcharts and vice versa so that the students are equally comfortable with both representations.

**Understanding the Reasons for Structure**

1. Walk through the bulleted list on pages 110-111 to discuss the advantages of structured programs.
The cost of software maintenance far exceeds the cost of the original development. The life span of most major software applications is five years or more.

**Recognizing Structure**

1. Review the flowchart examples in Figures 3-19 through 3-21 and explain why the first two are structured, but the third is not.

2. Walk through the steps to untangle the code represented by the flowchart in Figure 3-21, using the flowcharts in Figures 3-22 through 3-28.

**Structuring and Modularizing Unstructured Logic**

1. Review the dog-washing process shown in Figure 3-1. Using the flowcharts in Figures 3-29 through 3-33, explain how to restructure the logic.

2. Review the modularized version of the dog-washing process shown in Figure 3-34.

**Quick Quiz 2**

1. A(n) ____ is an added statement that gets the first input value in a program.  
   Answer: priming input or priming read

2. List the reasons for using the three structures discussed in the chapter.  
   Answer: clarity, professionalism, efficiency, maintenance, modularity

3. (True/False) When you are beginning to learn about structured program design, it is easy to detect whether a flowchart of a program’s logic is structured.  
   Answer: False

4. (True/False) No matter how complicated it is, any set of steps can always be reduced to combinations of the three basic structures of sequence, selection, and loop.  
   Answer: True

**Additional Resources**

1. A Wikipedia article about spaghetti code:
http://en.wikipedia.org/wiki/Spaghetti_code

2. An article about the Pasta Theory of Programming:  
www.wisegeek.com/what-is-the-pasta-theory-of-programming.htm

3. More on structured programming and Dijkstra’s contributions:  
http://en.wikipedia.org/wiki/Structured_programming

4. A Wikipedia article on flowcharting:  
http://en.wikipedia.org/wiki/Flowchart

Key Terms

- **Block** – a group of statements that executes as a single unit.
- **Dual-alternative ifs** (or **dual-alternative selections**) define one action to be taken when the tested condition is true and another action to be taken when it is false.
- **End-structure statements** – designate the ends of pseudocode structures.
- **Goto-less programming** – a name to describe structured programming, because structured programmers do not use a “go to” statement.
- **if-then-else** – another name for a selection structure.
- **Loop body** – the set of actions that occur within a loop.
- **Loop structure** – continue to repeat actions based on the answer to a question.
- **Nesting structures** – the act of placing a structure within another structure.
- **Null case** – the branch of a decision in which no action is taken.
- **Priming input** (**priming read**) – the statement that reads the first input data record prior to starting a structured loop.
- **Repetition** and **iteration** – alternate names for a loop structure.
- **Selection structure** (**decision structure**) – ask a question and, depending on the answer, take one of two courses of action. Then, no matter which path you follow, continue with the next task.
- **Sequence structure** – perform an action or task, and then perform the next action, in order. A sequence can contain any number of tasks, but there is no option to branch off and skip any of the tasks.
- **Single-alternative ifs** (or **single-alternative selections**) – take action on just one branch of the decision.
- **Spaghetti code** – snarled, unstructured program logic.
- **Stacking structures** – the act of attaching structures end to end.
- **Structure** – a basic unit of programming logic; each structure is a sequence, selection, or loop.
- **Structured programs** – programs that do follow the rules of structured logic.
- **Unstructured programs** – programs that do *not* follow the rules of structured logic.
- **while-do** (**while** loop) – a process continues while some condition continues to be true.