Chapter 2: Understanding Structure

Programming Logic and Design, 4th Edition Introductory
Objectives

• After studying Chapter 2, you should be able to:
  • Describe the features of unstructured spaghetti code
  • Describe the three basic structures of sequence, selection, and loop
  • Use a priming read
  • Appreciate the need for structure
  • Recognize structure
  • Describe two special structures—case and do until
Understanding Unstructured Spaghetti Code

- The popular name for snarled program statements is spaghetti code.
- The reason for the name should be obvious—the code is as confusing to read as following one noodle through a plate of spaghetti.

FIGURE 2-2: SPAGHETTI CODE EXAMPLE

Programming Logic and Design, Third Edition Introductory
Understanding the Three Basic Structures

• A structure is a basic unit of programming logic; each structure is a sequence, selection, or loop

• The first of these structures is a sequence, as shown in Figure 2-3

• With a sequence structure, you perform an action or event, and then you perform the next action, in order
Understanding the Three Basic Structures (continued)

• The second structure is called a selection structure or decision structure (if-then-else)
  – You ask a question
  – depending on the answer,
    • you take one of two courses of action
• no matter which path you follow, you continue with the next event
Understanding the Three Basic Structures (continued)

• **selection structure or decision structure** (else-if)
  – You ask a series of questions
  – Each question has a separate answer

• If none of the conditions are met, the last answer is chosen.
Understanding the Three Basic Structures (continued)
Understanding the Three Basic Structures (continued)

• The selection structure is sometimes called an if-then-else because it fits the following statement:
  – if someCondition is true then
    do oneProcess
  else
    do theOtherProcess
Understanding the Three Basic Structures (continued)

**Figure 2-5: Single-Alternative Decision Structure**
Understanding the Three Basic Structures (continued)

• **In a loop structure**, you ask a question; if the answer requires an action, you perform the action and ask the original question again.

• If the answer requires that the action be taken again, you take the action and then ask the original question again.

• Continues until the answer to the question is such that the action is no longer required; then you exit the structure.
Understanding the Three Basic Structures (continued)

• You may hear programmers refer to looping as repetition or iteration
Understanding the Three Basic Structures (continued)

- All logic problems can be solved using only these three structures—sequence, selection, and looping
- The three structures can be combined in an infinite number of ways
- Attaching structures end-to-end is called stacking structures
Understanding the Three Basic Structures (continued)

**Figure 2-1:** Structure Flowchart and Pseudocode

```
<table>
<thead>
<tr>
<th>Sequence</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>stepA</td>
</tr>
<tr>
<td></td>
<td>stepB</td>
</tr>
</tbody>
</table>

**Selection**

- **Condition C?**
  - **No**
    - stepE
  - **Yes**
    - stepD

**Loop**

- **Condition F?**
  - **Yes**
    - stepG
  - **No**

---

do stepA

do stepB

if conditionC is true then
  do stepD
else
  do stepE
endif

while conditionF is true
  do stepG
endwhile
Understanding the Three Basic Structures (continued)

- Placing a structure within another structure is called **nesting** the structures.
Using the Priming Read

• A priming read or priming input is the first read or data input statement in a program

• If a program will read 100 data records, you read the first data record in a statement that is separate from the other 99

• You must do this to keep the program structured

• With a selection structure, the logic goes in one of two directions after the question, and then the flow comes back together; the question is not asked a second time
Using the Priming Read (continued)

• In a loop, if an answer results in the loop being entered and loop statements executing, then
  – the logic returns to the question that started the loop

• when the body of a loop executes, the question that controls the loop is always asked again
Using the Priming Read (continued)

Figure 2-12: Unstructured Flowchart of a Number-Doubling Program

1. Start
2. Get inputNumber
3. Check if EOF (end of file)
   - Yes: Stop
   - No: Calculate calculatedAnswer = inputNumber * 2
4. Print calculatedAnswer

Programming Logic and Design, Third Edition Introductory
Using the Priming Read (continued)
Using the Priming Read (continued)
Using the Priming Read (continued)
Using the Priming Read (continued)
Using the Priming Read (continued)

FIGURE 2-17: FUNCTIONAL, STRUCTURED FLOWCHART AND PSEUDOCODE FOR THE NUMBER-DOUBLING PROBLEM

start

get inputNumber

Priming read

start

get inputNumber

while not eof

calculatedAnswer = inputNumber * 2

print calculatedAnswer

get inputNumber

endwhile

stop

stop

eof?

No

Yes

calculatedAnswer = inputNumber * 2

print calculatedAnswer

get inputNumber

Programming Logic and Design, Third Edition Introductory
Using the Priming Read (continued)
Understanding the Reasons for Structure

• Until you have some programming experience, it might be difficult to appreciate the reasons for using only the three structures—sequence, selection, and loop

• However, staying with these three structures is better for the following reasons:
  – Clarity
  – Efficiency
  – Modularity
  – Professionalism
  – Maintenance
Flowchart and Pseudocode of Structured College Admission Program

**FIGURE 2-19: FLOWCHART AND PSEUDOCODE OF STRUCTURED COLLEGE ADMISSION PROGRAM**

```plaintext
start
read testScore, classRank
if testScore >= 90 then
    if classRank >= 25 then
        print "Accept"
    else
        print "Reject"
    endif
else
    if testScore >= 80 then
        if classRank >= 50 then
            print "Accept"
        else
            print "Reject"
        endif
    else
        if testScore >= 70 then
            if classRank >= 75 then
                print "Accept"
            else
                print "Reject"
            endif
        else
            print "Reject"
        endif
    endif
endif
stop
```
Flowchart and Pseudocode of Structured
College Admission Program (continued)

Figure 2-19 (continued)
Recognizing Structure

- Any set of instructions can be expressed in a structured format
- If you can teach someone how to perform any ordinary activity, then you can express it in a structured way
Pseudocode for the Rock, Paper, Scissors Game

- Any task with applied rules can be expressed logically using only combinations of sequence, selection, and looping.
Two Special Structures—Case and Do Until

• Many programming languages allow two more structures: the case structure and the do until loop

• Never needed to solve any problem though sometimes are convenient

• Programmers consider them both to be acceptable, legal structures
**The Case Structure**

- Use the case structure when there are several distinct possible values for a single variable being tested and each value requires different actions.

```plaintext
if class = "Freshman" then
tuitionFee = 75
else
    if class = "Sophomore" then
tuitionFee = 50
    else
        if class = "Junior" then
tuitionFee = 30
        else
tuitionFee = 10
        endif
    endif
endif
```
The Case Structure (continued)

case based on class
  case "Freshman"
    tuitionFee = 75
  case "Sophomore"
    tuitionFee = 50
  case "Junior"
    tuitionFee = 30
  default
    tuitionFee = 10
endcase
The Do Until Loop

• In a **do while loop**, you ask a question and, depending on the answer, you might or might not enter the loop to execute the loop’s procedure.

• Conversely, in a **do until loop**, you ensure that the procedure executes at least once; then, depending on the answer to the controlling question, the loop may or may not execute additional times.
The Do Until Loop (continued)

- Because programmers understand that a do until can be expressed with a sequence followed by a do while, most languages allow the do until.

- Again, you are never required to use a do until; you can always accomplish the same events with a sequence followed by a do while.
The Do Until Loop (continued)
The Do Until Loop (continued)

FIGURE 2-35: FLOWCHART AND PSEUDOCODE FOR DO UNTIL LOOP

```
do
  A
until B is not true
```

A

B?

Yes

No
The Do While Loop (continued)

FIGURE 2-3b: FLOWCHART AND PSEUDOCODE FOR SEQUENCE FOLLOWED BY DO WHILE LOOP

```
do A
while B is true
   do A
endwhile
```
Summary

- The popular name for snarled program statements is spaghetti code
- A priming read or priming input is the first read or data input statement prior to beginning a structured loop
- The last step within the loop gets the next, and all subsequent, input values
Summary (continued)

• You can use a case structure when there are several distinct possible values for a variable you are testing

• In a do while loop, you ask a question and, depending on the answer, you might never enter the loop to execute the loop’s procedure

• In a do until loop, you ensure that the procedure executes at least once