Programming Logic and Design
Ninth Edition

Chapter 3
Understanding Structure

Objectives

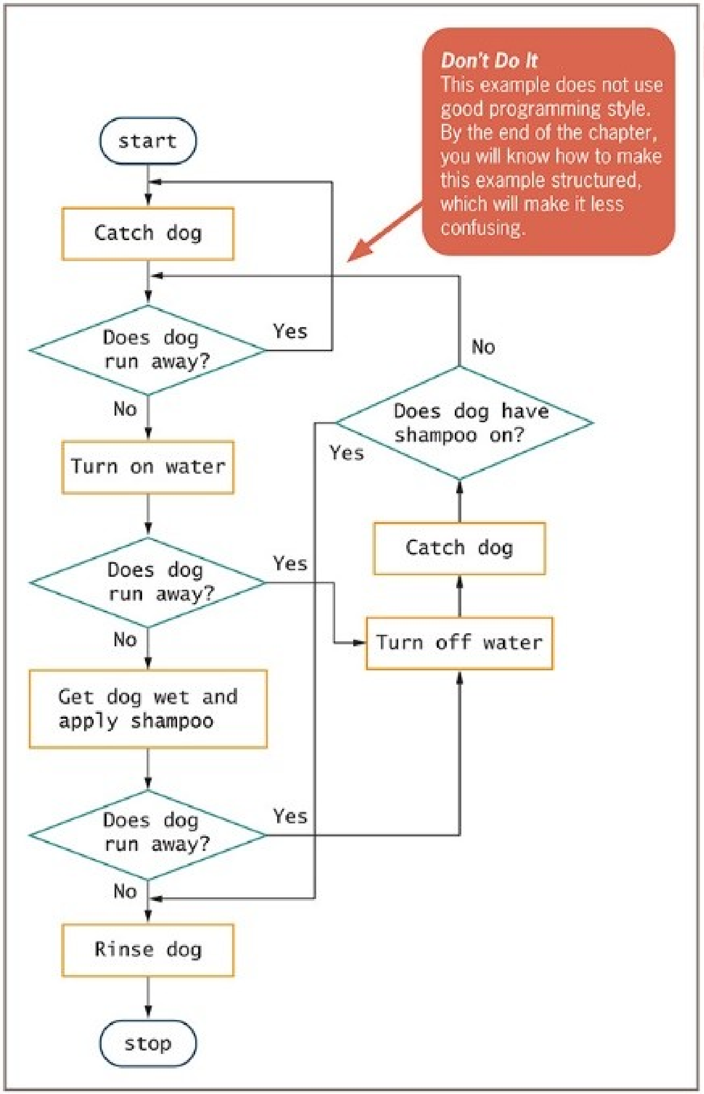
In this chapter, you will learn about:

- The disadvantages 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

The Disadvantages of Unstructured Spaghetti Code

- **Spaghetti code**
  - Logically snarled program statements
  - Often a complicated mess
  - Programs often work but are difficult to read and maintain
  - Confusing and prone to error
- **Unstructured programs**
  - Do not follow the rules of structured logic
- **Structured programs**
  - Follow the rules of structured logic

Unstructured Spaghetti Code



Don't Do It
This example does not use good programming style. By the end of the chapter, you will know how to make this example structured, which will make it less confusing.

Figure 3-1 Spaghetti code logic for washing a dog

Understanding the Three Basic Structures

- **Structure**
  - Basic unit of programming logic
  - Each structure is one of the following:
    - **Sequence structure**
    - **Selection structure** (**decision structure**)
    - **Loop structure**
  - any program can be constructed using one or more of these three structures

Understanding the Three Basic Structures (continued)

- **Sequence structure**
  - Perform actions or tasks in order
  - No branching or skipping any task
- **Selection structure (decision structure)**
  - Ask a question, take one of two actions based on testing a condition. Known as evaluating a **Boolean expression**, a statement that is either true or false
  - Often called **if-then-else**
  - **Dual-alternative `ifs`** or **single-alternative `ifs`**
- **Loop structure**
  - Repeat actions while a condition remains true

The Sequence Structure

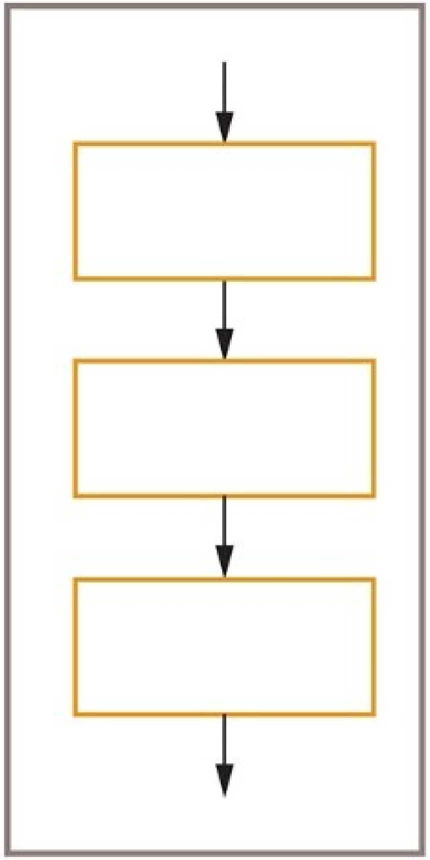


Figure 3-2 Sequence structure

The Selection Structure

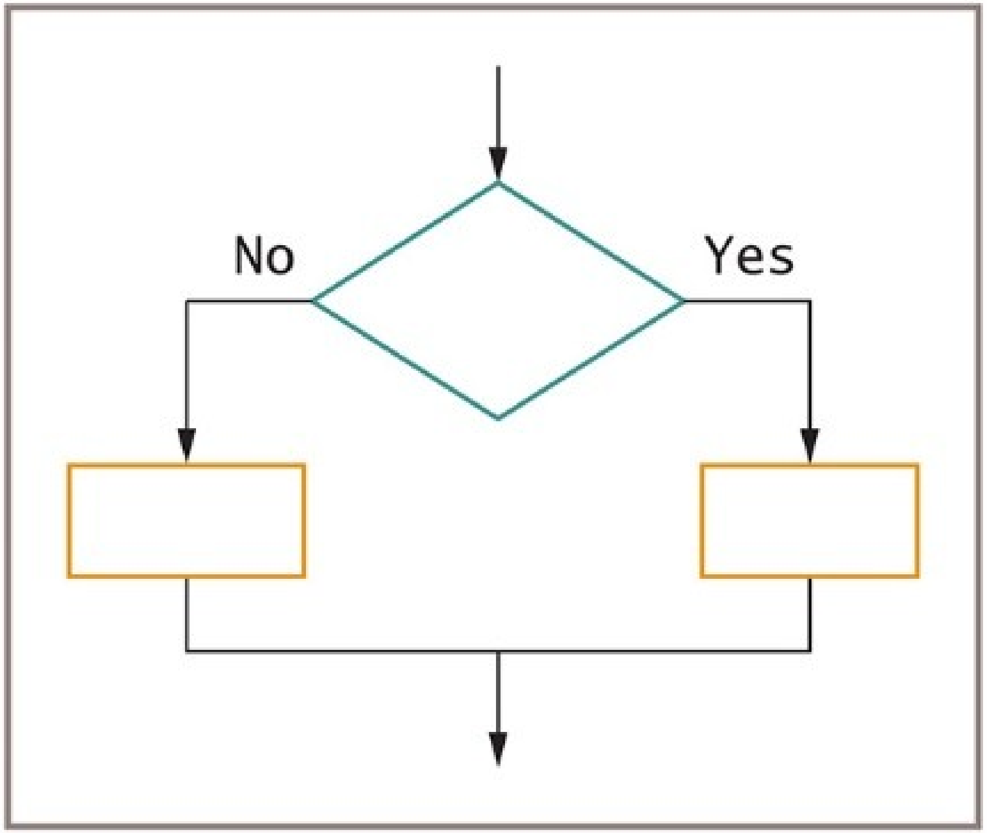


Figure 3-3 Selection structure

The Selection Structure (continued -1)

- **Dual-alternative `ifs`**
  - Contains two alternatives
  - The **`if-then-else`** structure

```
if someCondition is true then
      do oneProcess
else
      do theOtherProcess
endif
```

The Selection Structure (continued -2)

- **Single-alternative ifs**

```
if employee belongs to dentalPlan then
     deduct $40 from employeeGrossPay
```

  - An `else` clause is not required
- **null case**
  - Situation where nothing is done

The Selection Structure (continued -3)

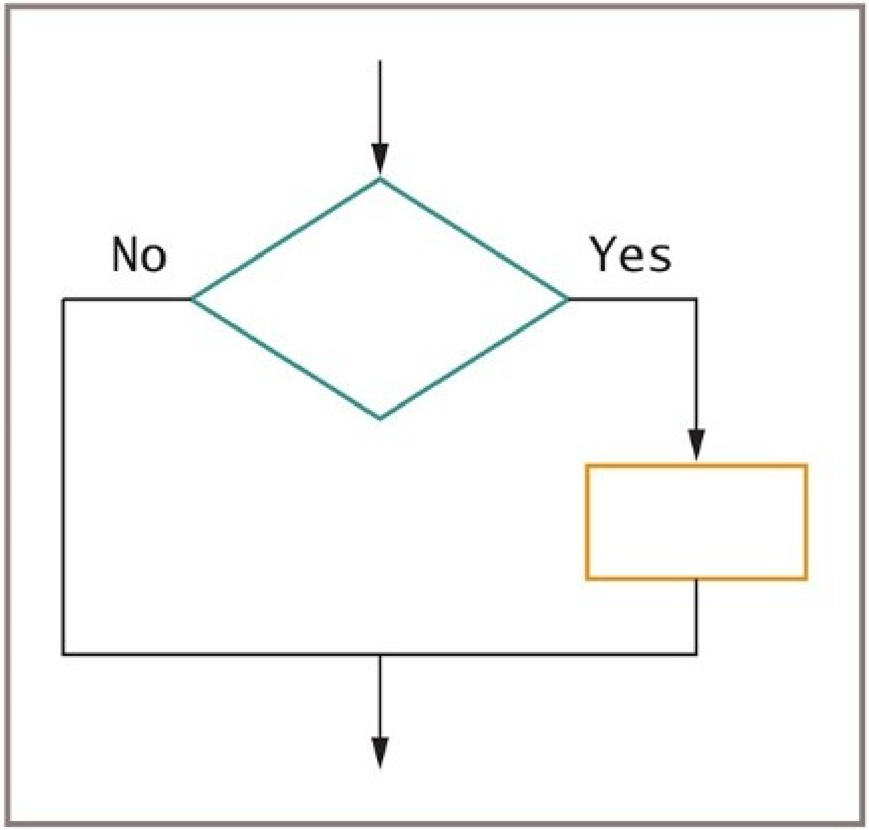


Figure 3-4 Single-alternative selection structure

The Loop Structure

- **Loop structure**
  - Repeats a set of actions while a condition remains true
    - **Loop body**
  - Also called **repetition** or **iteration**
  - Condition is tested first in the most common form of loop
  - The **`while`…do** or **`while` loop**

The Loop Structure (continued -1)

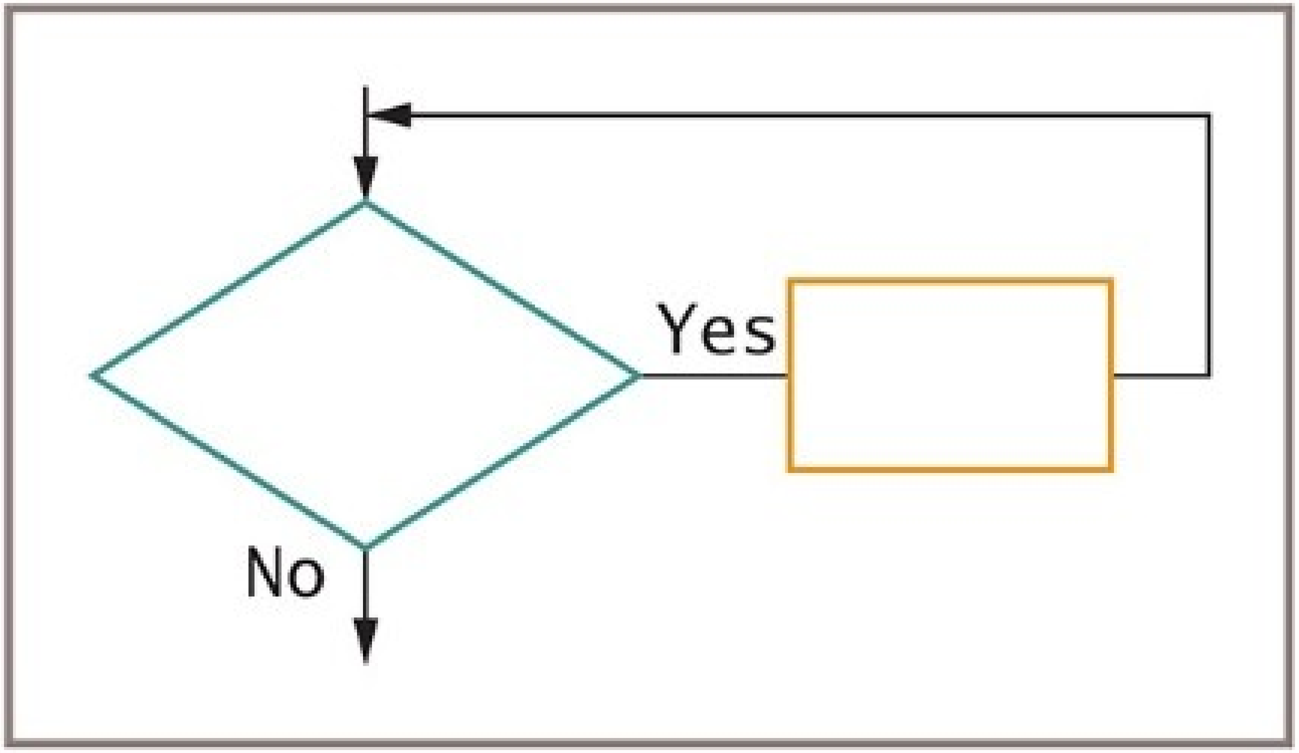


Figure 3-5 Loop structure

The Loop Structure (continued -2)

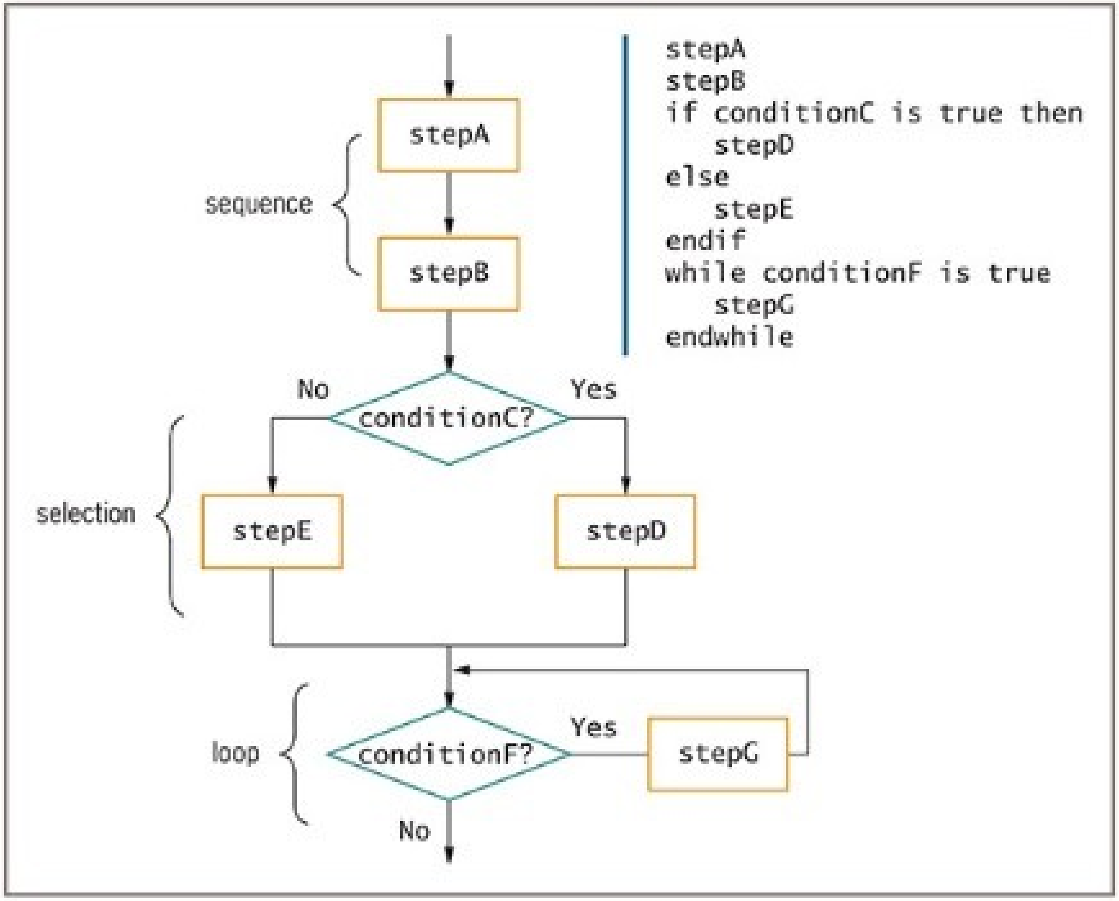
- **Loop structure**

```
while testCondition continues to be true
     do someProcess
endwhile
while you continue to be hungry
     take another bite of food
     determine if you still feel hungry
endwhile
```

Combining Structures

- All logic problems can be solved using only sequence, selection, and loop
- Structures can be combined in an infinite number of ways
- **Stacking structures**
  - Attaching structures end-to-end
- **End-structure statement**
  - Indicates the end of a structure
  - The `endif` statement ends an `if-then-else` structure
  - The `endwhile` statement ends a loop structure

Combining Structures (continued -1)



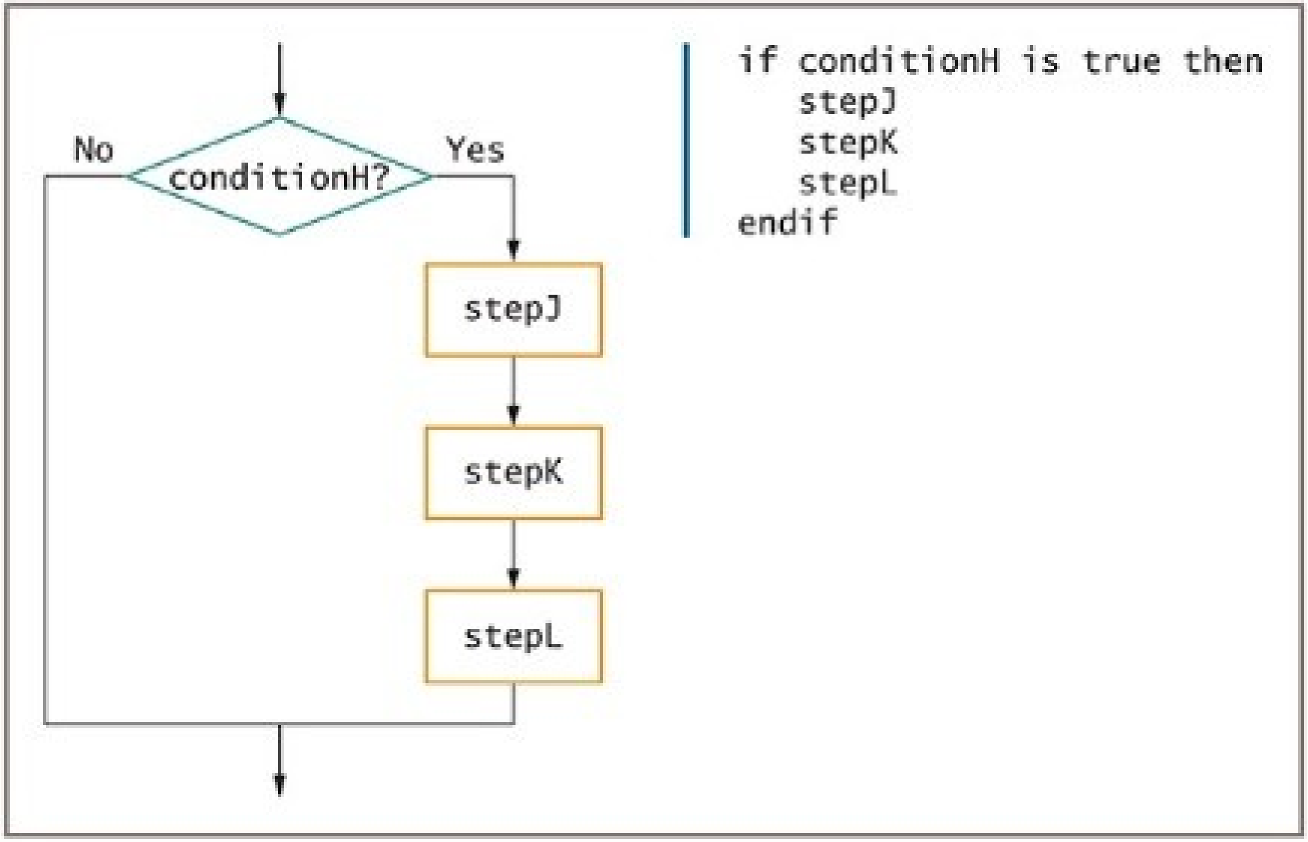
```
stepA
stepB
if conditionC is true then
   stepD
else
   stepE
endif
while conditionF is true
   stepG
endwhile
```

Figure 3-6 Structured flowchart and pseudocode with three stacked structures

Combining Structures (continued -2)

- Any individual task or step in a structure can be replaced by a structure
- **Nesting structures**
  - Placing one structure within another
  - Indent the nested structure’s statements
- **Block**
  - A group of statements that execute as a single unit

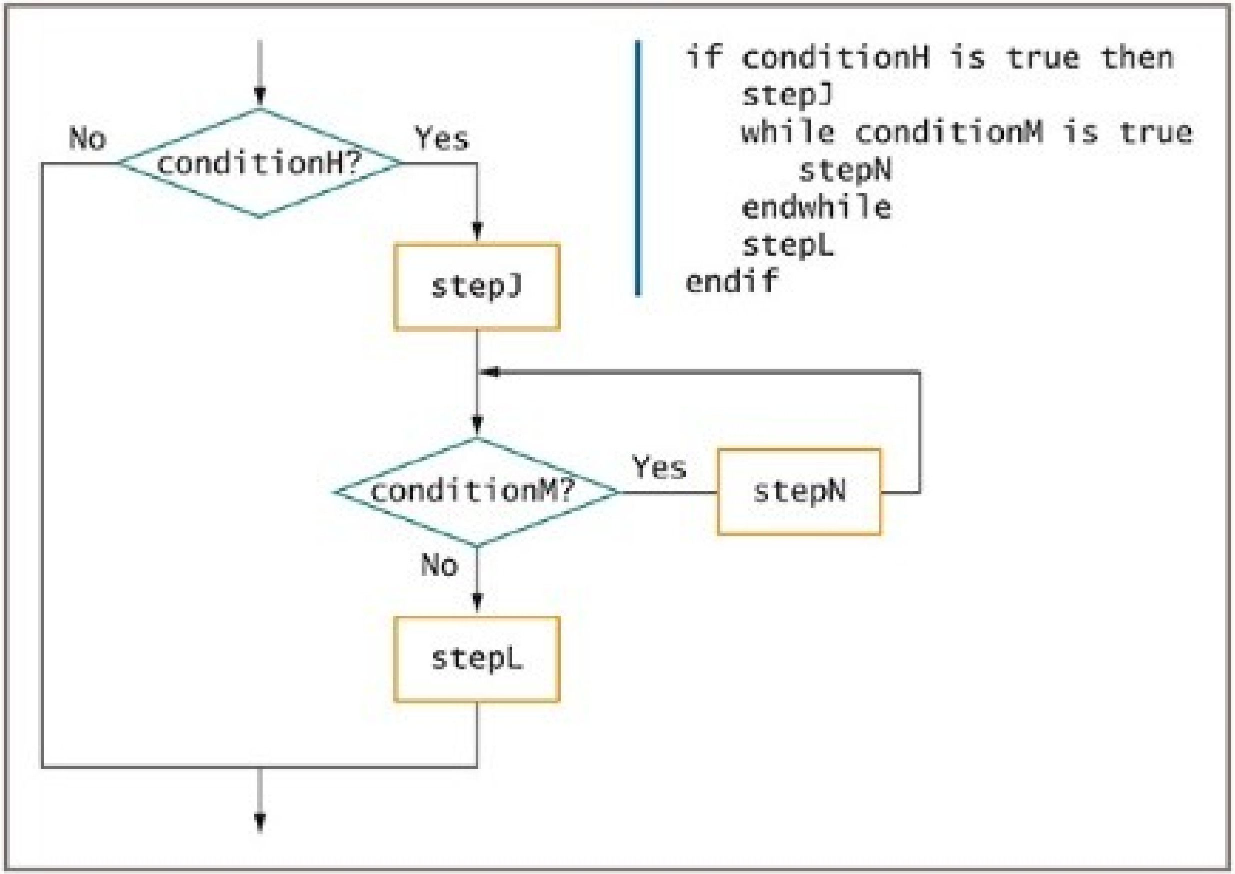
Combining Structures (continued -3)



```
if conditionH is true then
   stepJ
   stepK
   stepL
endif
```

Figure 3-7 Flowchart and pseudocode showing nested structures—a sequence nested within a selection

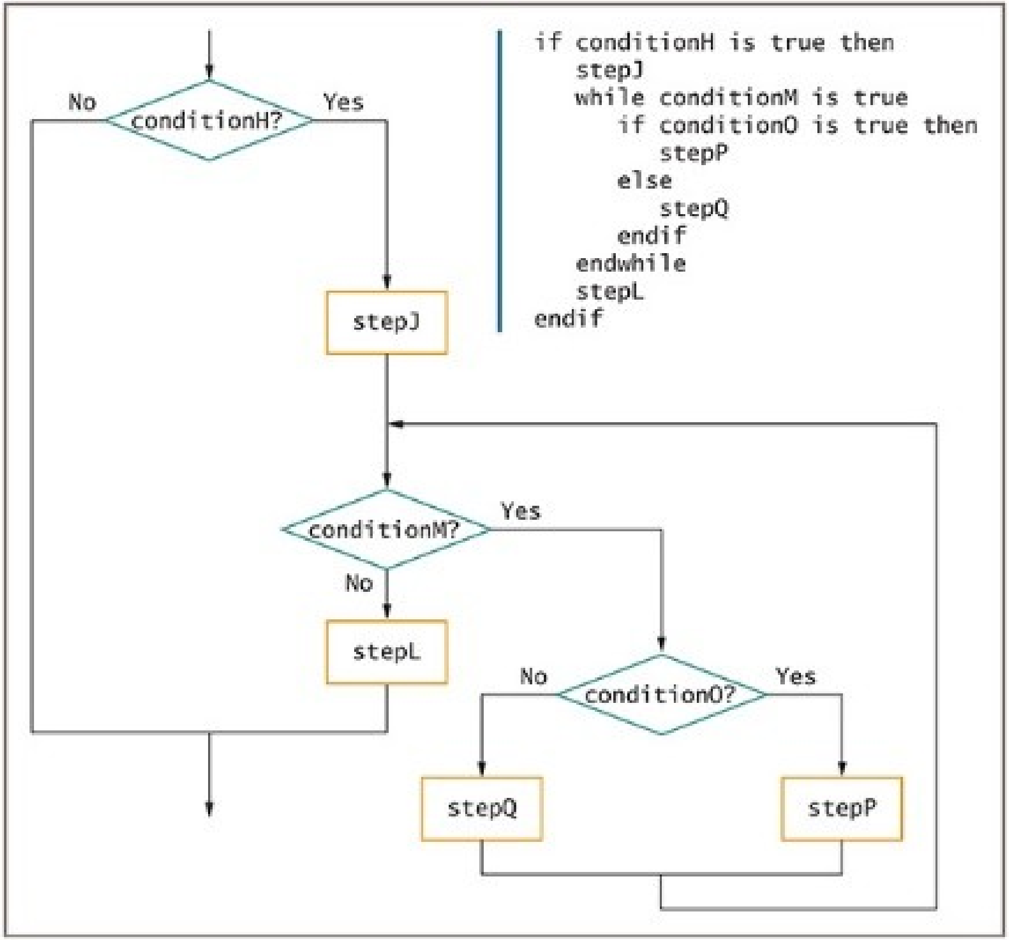
Combining Structures (continued -4)



```
if conditionH is true then
   stepJ
   while conditionM is true
      stepN
   endwhile
   stepL
endif
```

Figure 3-8 Flowchart and pseudocode showing nested structures—a loop nested within a sequence, nested within a selection

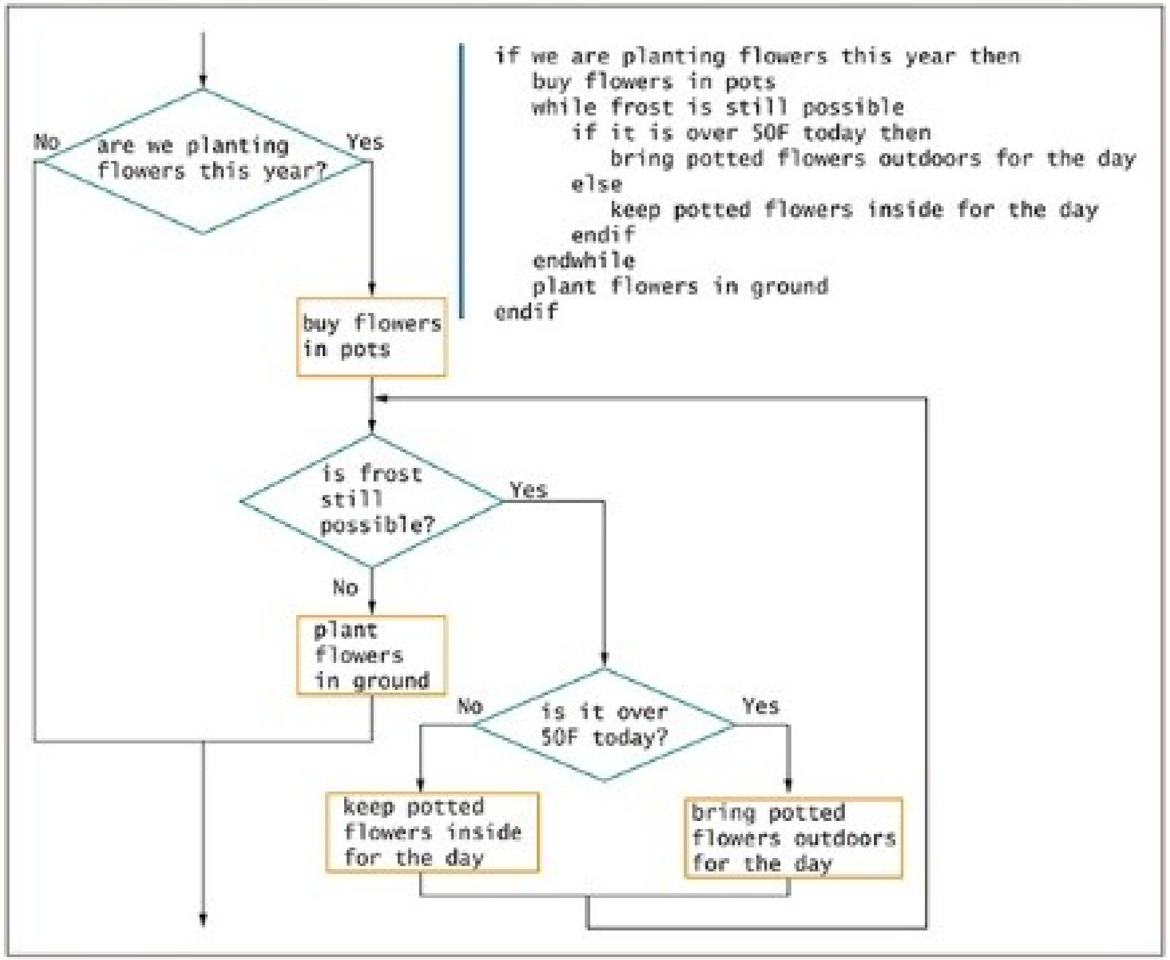
Combining Structures (continued -5)



```
if conditionH is true then
   stepJ
   while conditionM is true
      if conditionO is true then
         stepP
      else
         stepQ
      endif
   endwhile
   stepL
endif
```

Figure 3-9 Flowchart and pseudocode for a selection within a loop within a sequence within a selection

Combining Structures (continued -6)



No
are we planting flowers this year?
Yes
buy flowers in pots
is frost still possible?
Yes
No
plant flowers in ground
No
is it over 50F today?
Yes
keep potted flowers inside for the day
bring potted flowers outdoors for the day

```
if we are planting flowers this year then
   buy flowers in pots
   while frost is still possible
      if it is over 50F today then
         bring potted flowers outdoors for the day
      else
         keep potted flowers inside for the day
      endif
   endwhile
   plant flowers in ground
endif
```

Figure 3-10 The process of buying and planting flowers in the spring

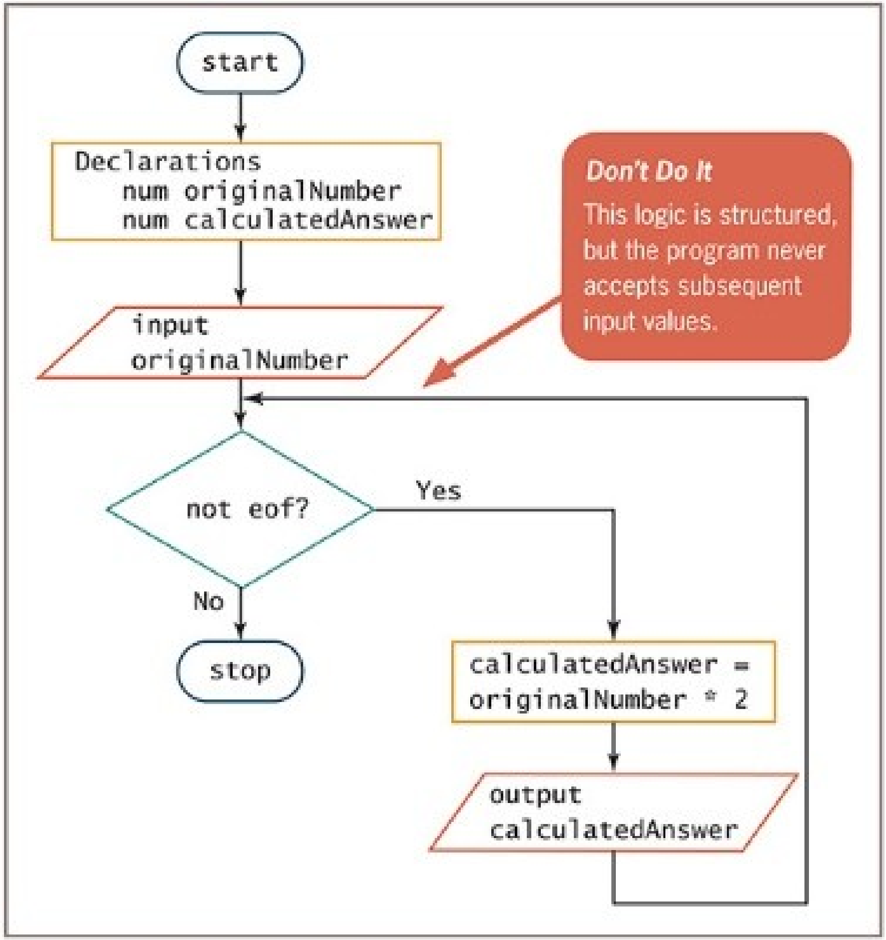
Combining Structures (continued -7)

- Structured programs have the following characteristics:
  - Include only combinations of the three basic structures
  - Each structure has a single entry point and a single exit point
  - Structures can be stacked or connected to one another only at their entry or exit points
  - Any structure can be nested within another structure

Using a Priming Input to Structure a Program

- **Priming input** (or **priming read**)
  - Reads the first input data record
  - Is outside the loop that reads the rest of the records
  - Helps keep the program structured
- Analyze a flowchart for structure one step at a time
- Watch for unstructured loops that do not follow this order
  - First ask a question
  - Take action based on the answer
  - Return to ask the question again

Using a Priming Input to Structure a Program



Don't Do It

This logic is structured, but the program never accepts subsequent input values.

Figure 3-14 Structured, but nonfunctional, flowchart of number-doubling problem

Using a Priming Input to Structure a Program

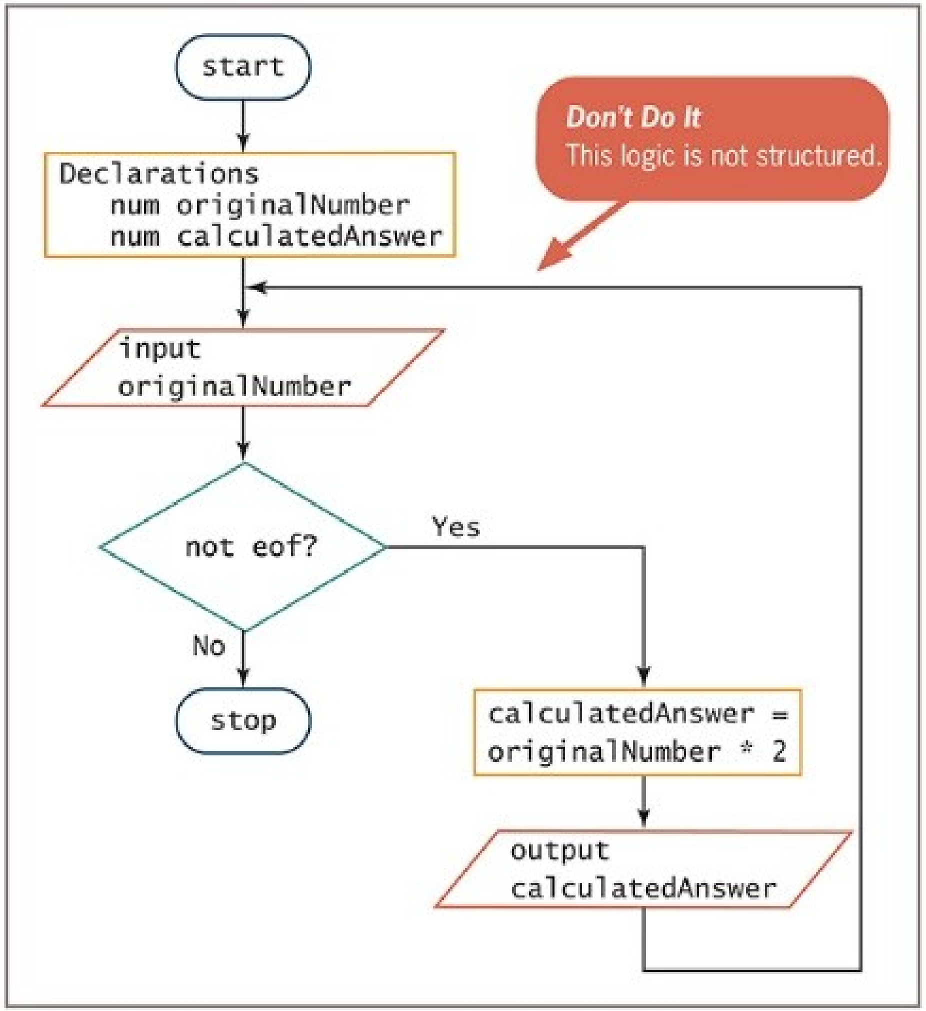


Figure 3-15 Functional but unstructured flowchart

Using a Priming Input to Structure a Program

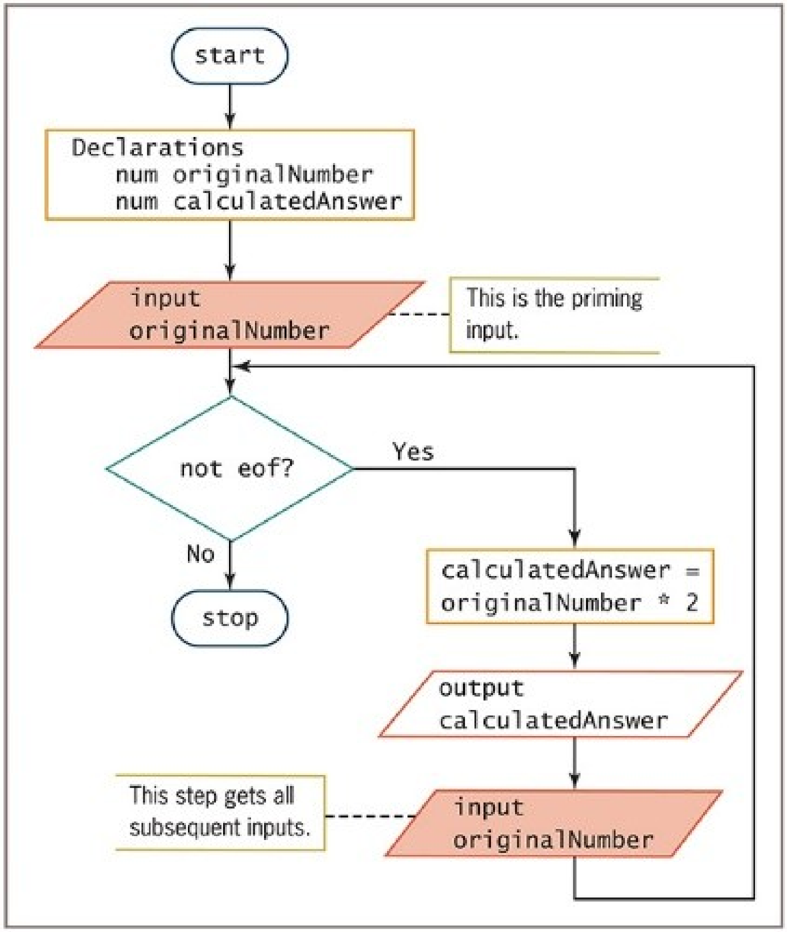
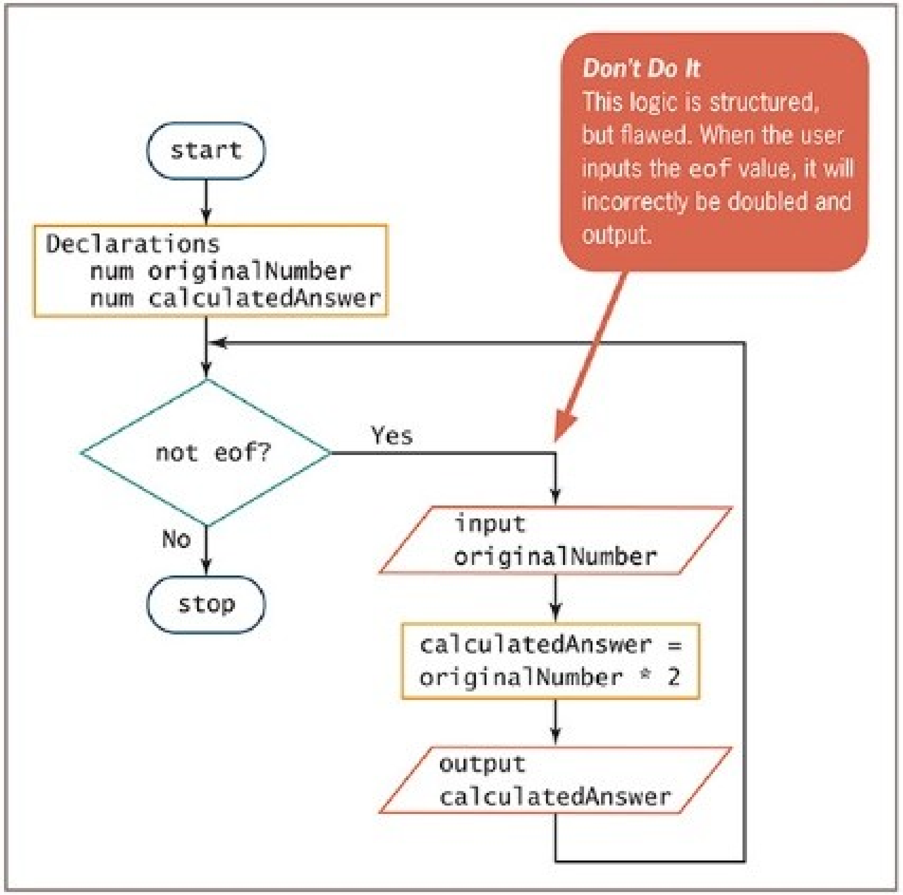


Figure 3-16 Functional, structured flowchart for the number-doubling problem

Using a Priming Input to Structure a Program



Don't Do It
This logic is structured, but flawed. When the user inputs the eof value, it will incorrectly be doubled and output.

Figure 3-17 Structured but incorrect solution to the number-doubling problem

Understanding the Reasons for Structure

- **Use structured programming for:**
  - **Clarity**—unstructured programs are confusing
  - **Professionalism**—other programmers expect it
  - **Efficiency**—most languages support it
  - **Maintenance** —other programmers find it easier to read
  - ***Modularity*** —easily broken down into modules
- Structured programming is sometimes called **goto-less programming**

Recognizing Structure

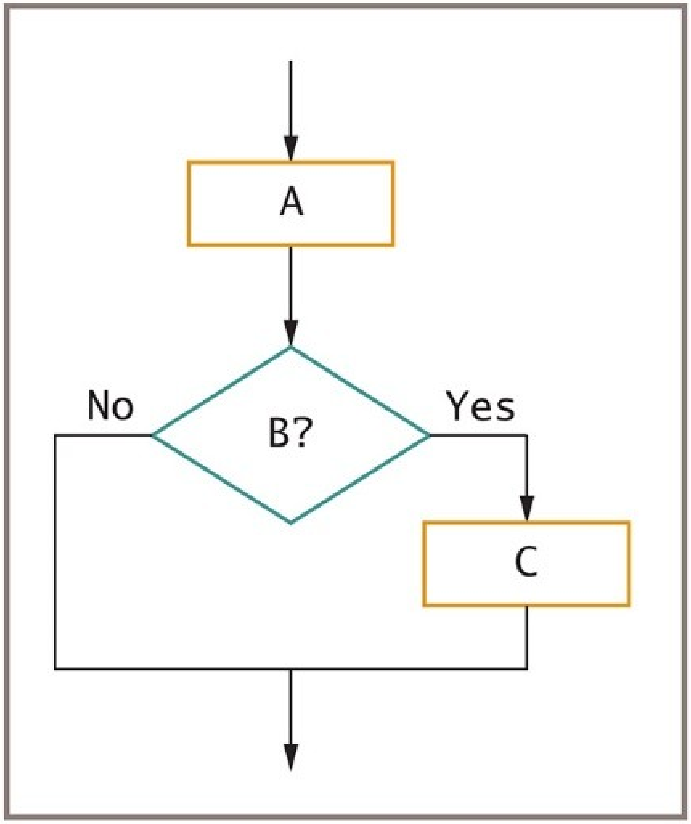


Figure 3-18 Example 1

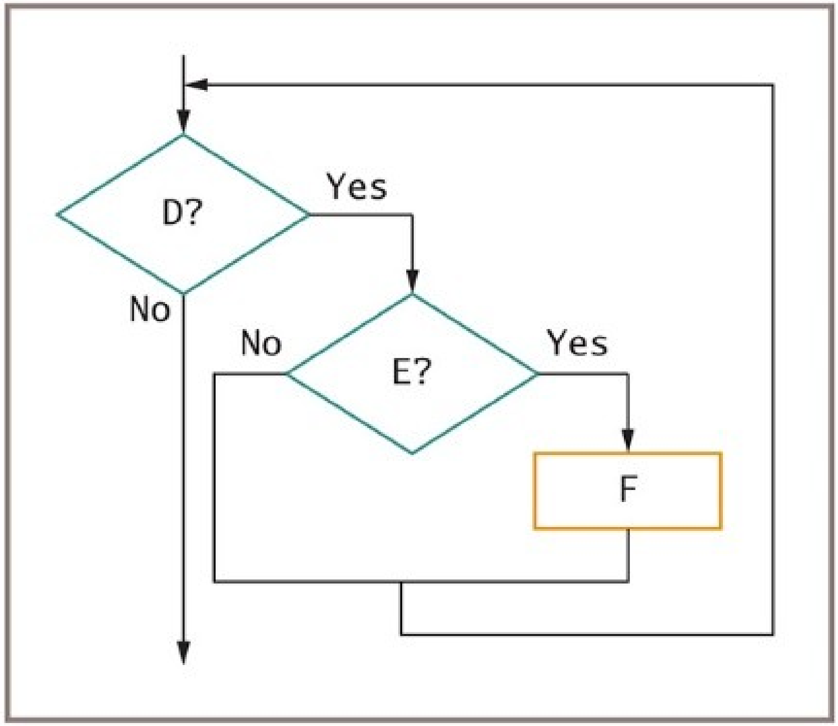


Figure 3-19 Example 2

Recognizing Structure (contin

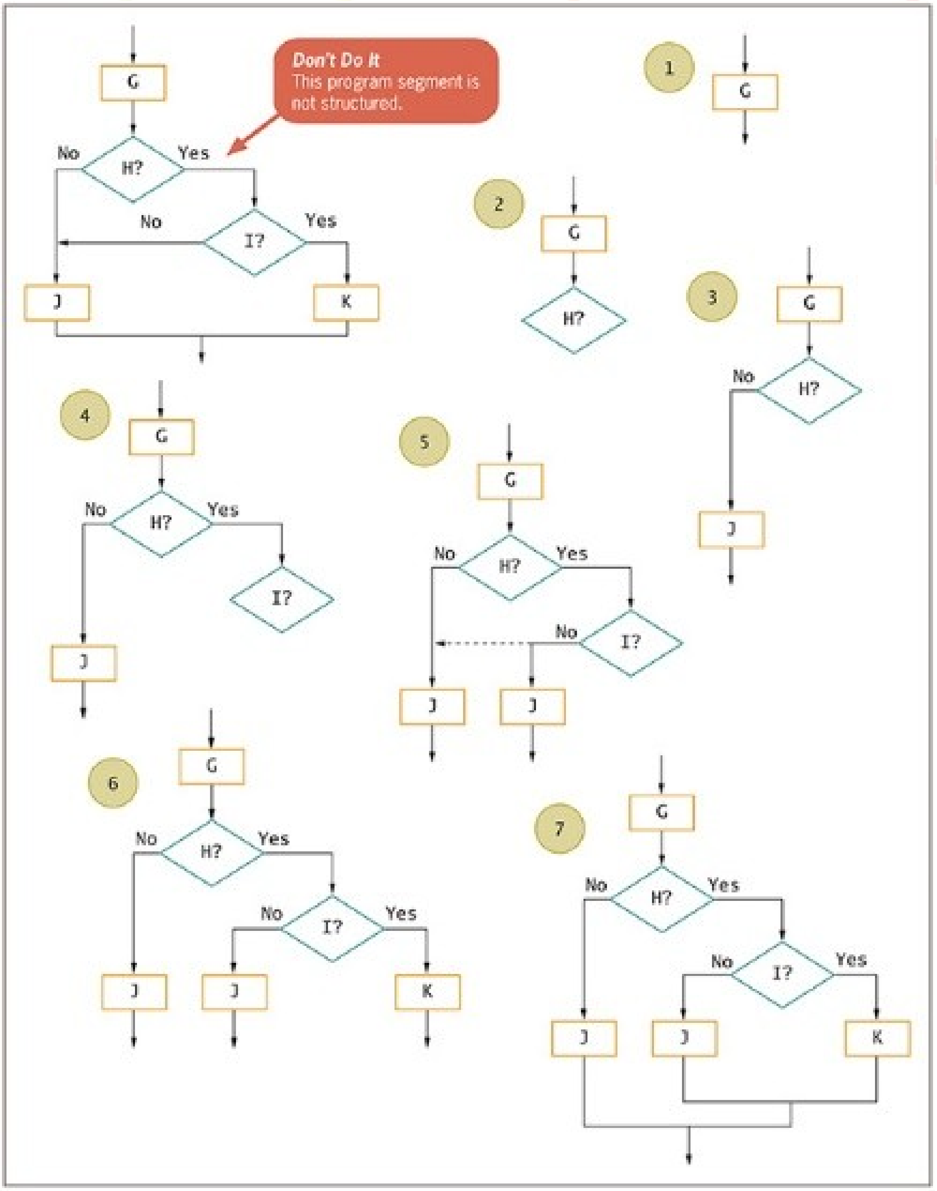
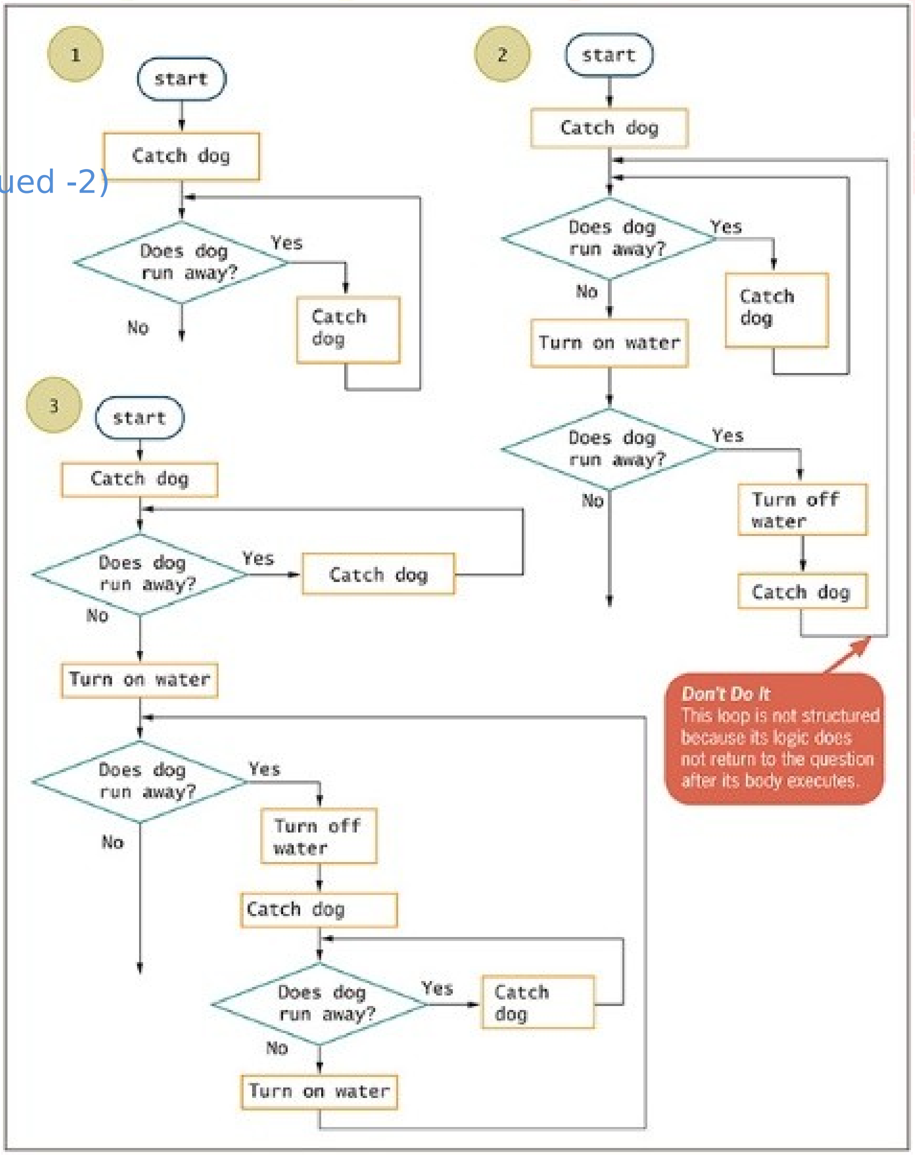


Figure 3-20 Example 3 and process to structure it

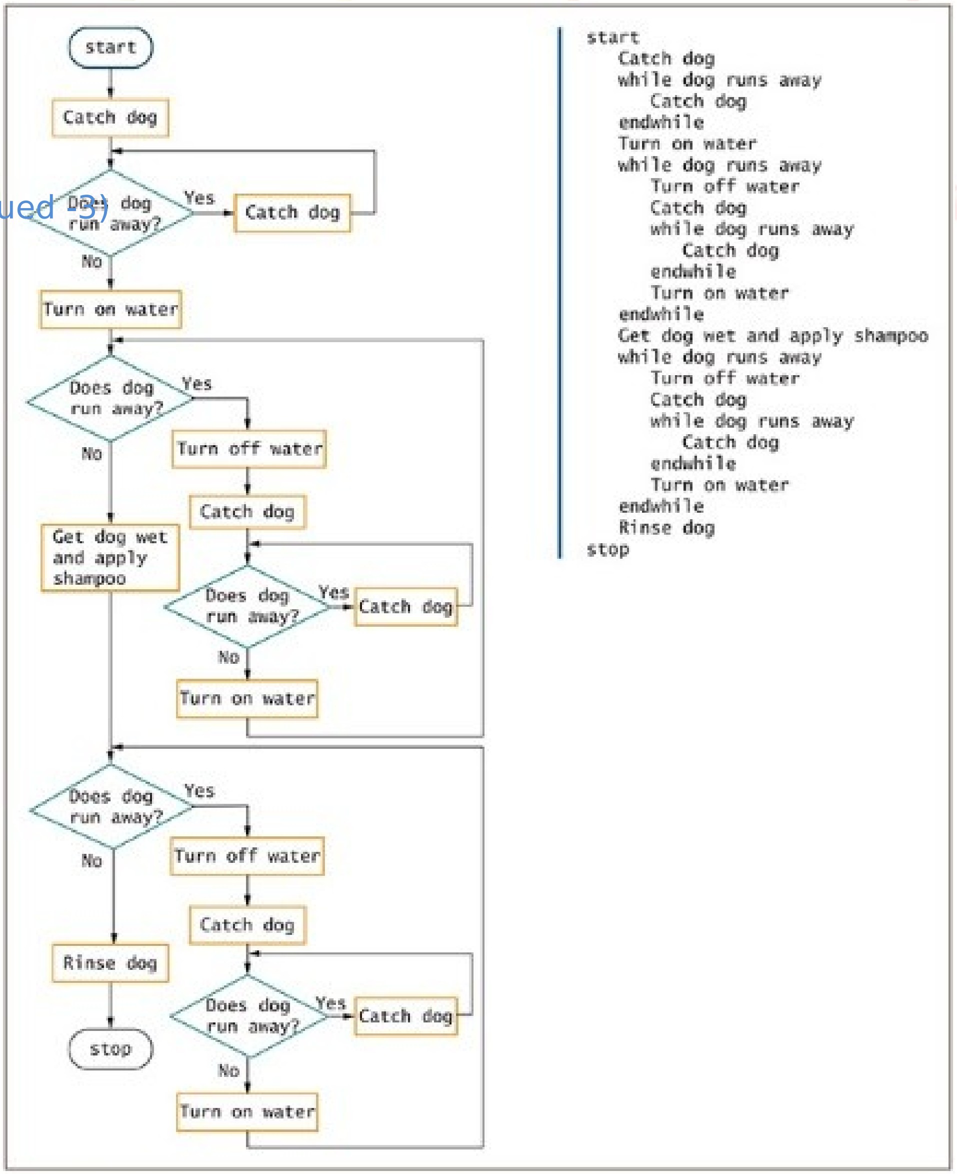
Recognizing Structure (continued -2)



Don't Do It
This loop is not structured because its logic does not return to the question after its body executes.

Figure 3-21 Steps to structure the dog-washing process

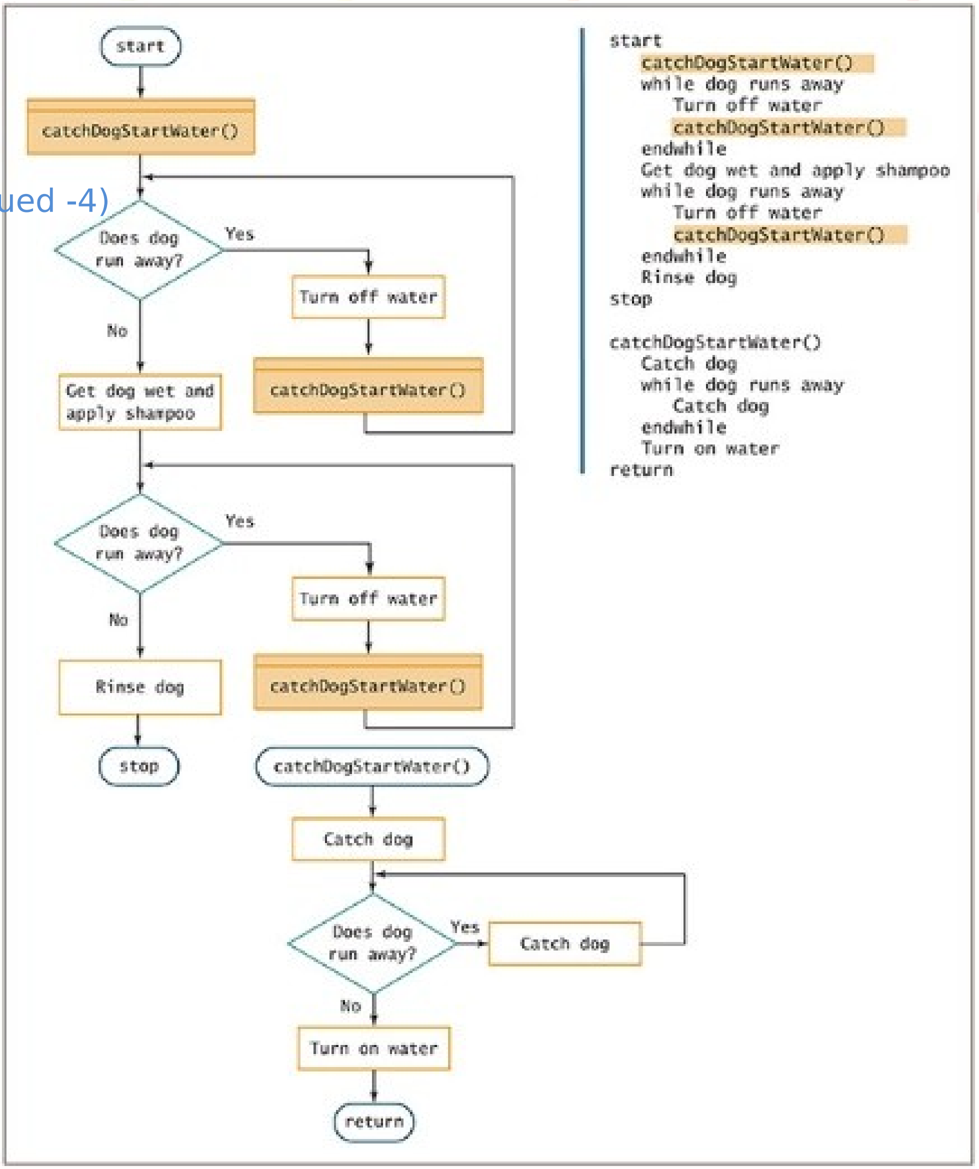
Recognizing Structure (continued 3)



```
start
   Catch dog
   while dog runs away
      Catch dog
   endwhile
   Turn on water
   while dog runs away
      Turn off water
      Catch dog
      while dog runs away
         Catch dog
      endwhile
      Turn on water
   endwhile
   Get dog wet and apply shampoo
   while dog runs away
      Turn off water
      Catch dog
      while dog runs away
         Catch dog
      endwhile
      Turn on water
   endwhile
   Rinse dog
stop
```

Figure 3-22 Structured dog-washing flowchart and pseudocode

Recognizing Structure (continued -4)



```
start
   catchDogStartWater()
   while dog runs away
      Turn off water
      catchDogStartWater()
   endwhile
   Get dog wet and apply shampoo
   while dog runs away
      Turn off water
      catchDogStartWater()
   endwhile
   Rinse dog
stop

catchDogStartWater()
   Catch dog
   while dog runs away
      Catch dog
   endwhile
   Turn on water
return
```

Figure 3-23 Modularized version of the dog-washing program

Summary

- Spaghetti code
  - Statements that do not follow rules of structured logic
- Three basic structures
  - Sequence, selection, and loop
  - Combined by stacking and nesting
- Priming input
  - Statement that reads the first input value prior to starting a structured loop

Summary (continued)

- Structured techniques promote:
  - Clarity
  - Professionalism
  - Efficiency
  - Modularity
- Flowcharts can be made structured by untangling logic
- Logical steps can be rewritten to conform to the three structures: sequence, selection, and loop