Good Program Design

Msc. Ha Van Thao
HCM University of Science

1. Tổng quan về máy tính và lập trình

- Hệ thống máy tính
- Logic của chương trình máy tính
- Chu trình phát triển phần mềm
- Mã giả (Pseudo-code) và sơ đồ luồng (flowchart)
- Sử dụng “lính gác” để kết thúc chương trình
- Sự tiến hóa của mô hình lập trình
- Tổng kết
- Bài tập

1.1 Hệ thống máy tính

- Bao gồm:
  - Hardware: phần cứng
  - Software: phần mềm
    - Application software: phần mềm ứng dụng
    - System software: phần mềm hệ thống
- Hệ thống máy tính thực hiện 3 hoạt động cơ bản trong hầu hết chương trình:
  - Input: Nhập liệu
  - Processing: Xử lí
  - Output: Xuất kết quả

1.2 Logic của chương trình máy tính

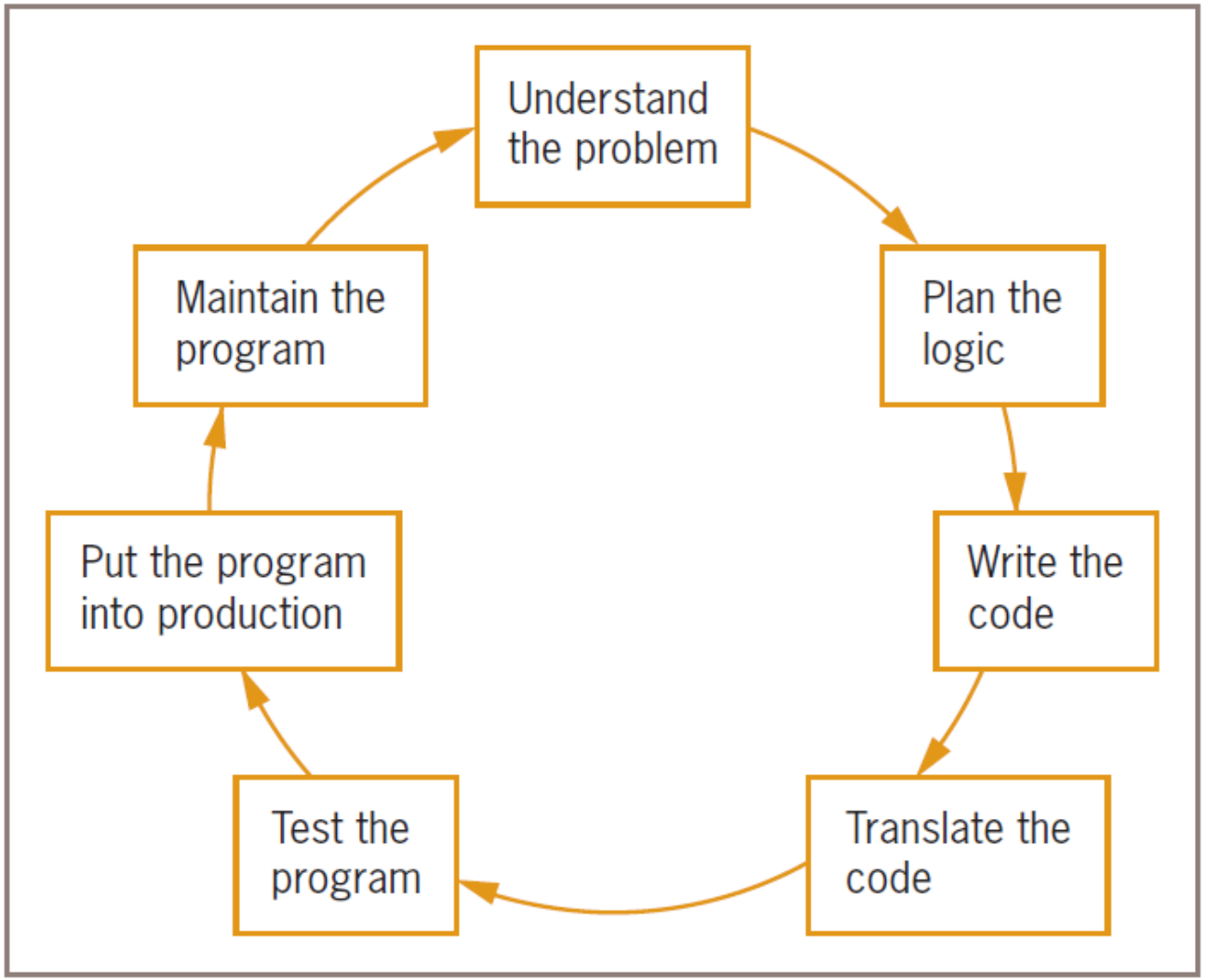
```
input myNumber
set myAnswer = myNumber * 2
output myAnswer
```

- Lệnh **input myNumber** mô tả hoạt động nhập liệu.
- Lệnh **set myAnswer = myNumber** * 2 mô tả hoạt động xử lí.
- Trong chương trình bình phương một số ở trên, lệnh **output myAnswer** mô tả hoạt động xuất kết quả.

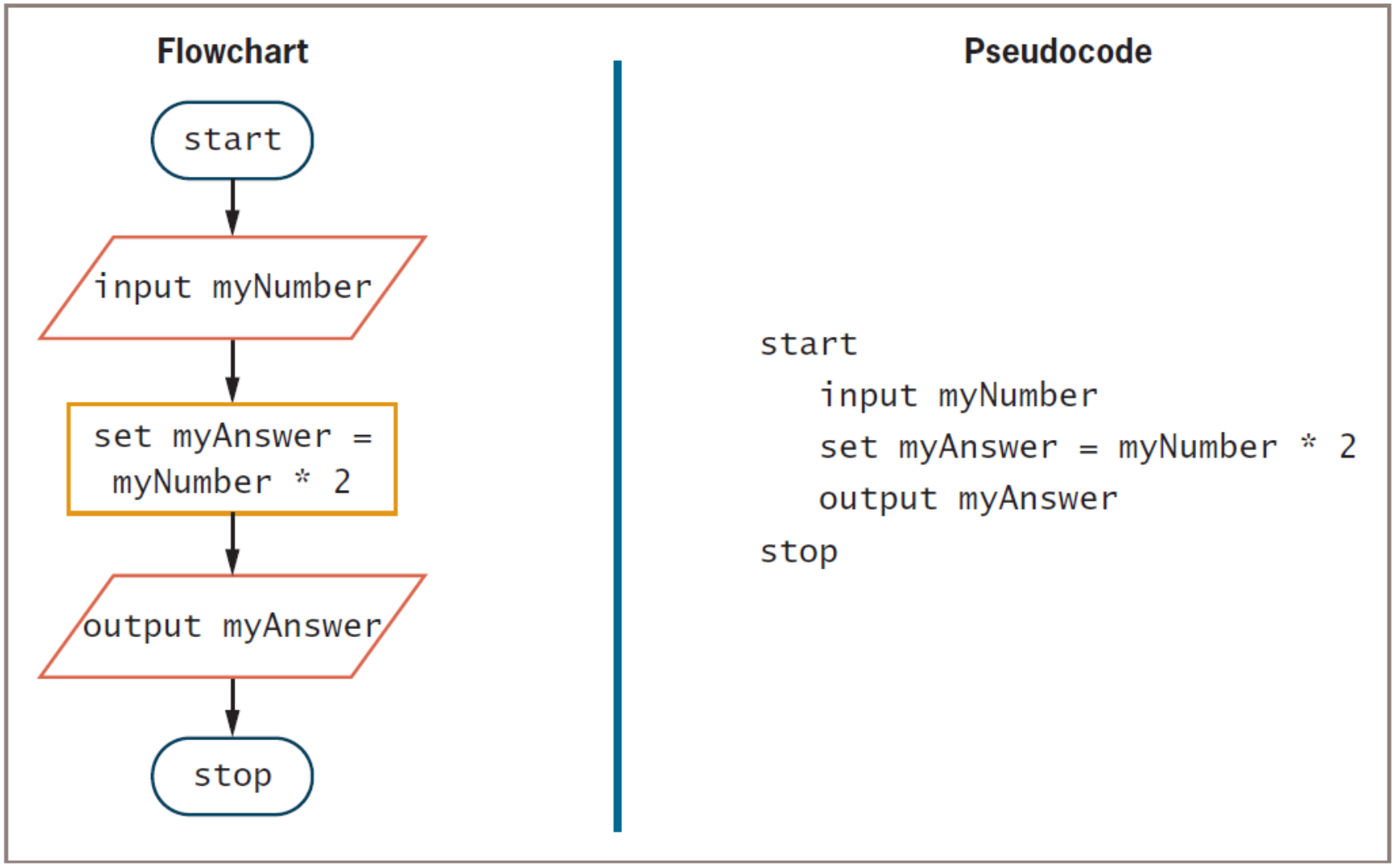
1.3 Chu trình phát triển chương trình

1. Tìm hiểu bài toán: end user, documentation
2. Xây dựng thuật toán: algorithm, desk-checking
3. Viết mã nguồn: syntax
4. Sử dụng phần mềm (compiler - biên dịch, interpreter - thông dịch) chuyển chương trình thành mã máy: high-level programming language, low-level machine language
5. Kiểm thử chương trình: debugging
6. Thành phẩm
7. Bảo trì

Chu trình phát triển chương trình



1.4 Sử dụng mã giả Pseudocode và sơ đồ luồng Flowchart



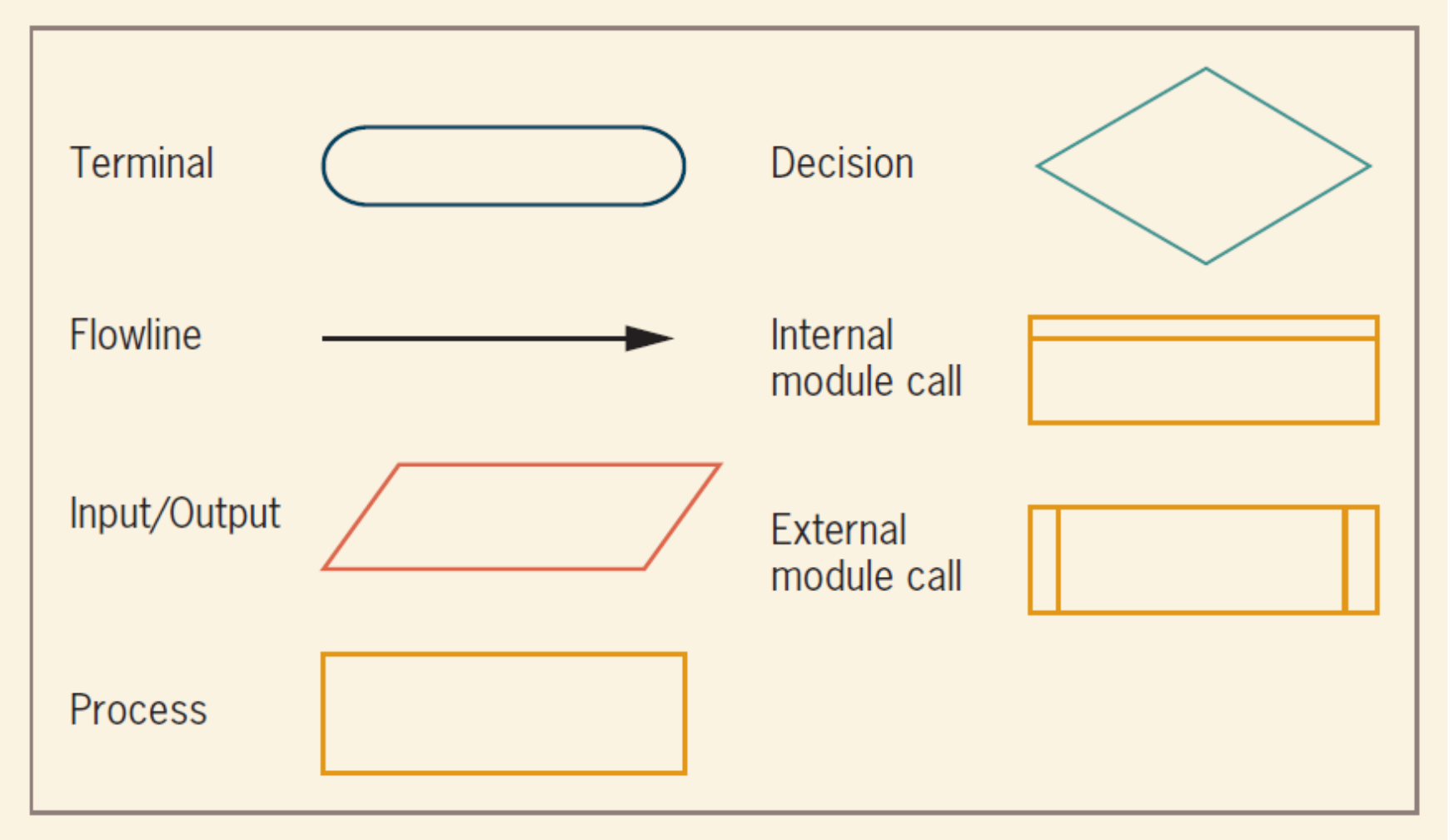
Qui tắc viết mã giả

- Chương trình bắt đầu bằng **start** và kết thúc bằng **stop**.
- Tên của module phải được theo sau bằng một cặp dấu ngoặc. Module phải được kết thúc bằng **return**.
- Mỗi lệnh chỉ thực hiện một hành động: input, processing, hay output.
- Lệnh phải được canh lề sau **start** hay sau tên của module.
- Mỗi lệnh nên xuất hiện trên một dòng nếu được. Ngược lại, dòng tiếp theo phải được canh lề.
- Lệnh bắt đầu từ kí tự viết thường.

Vẽ Flowchart

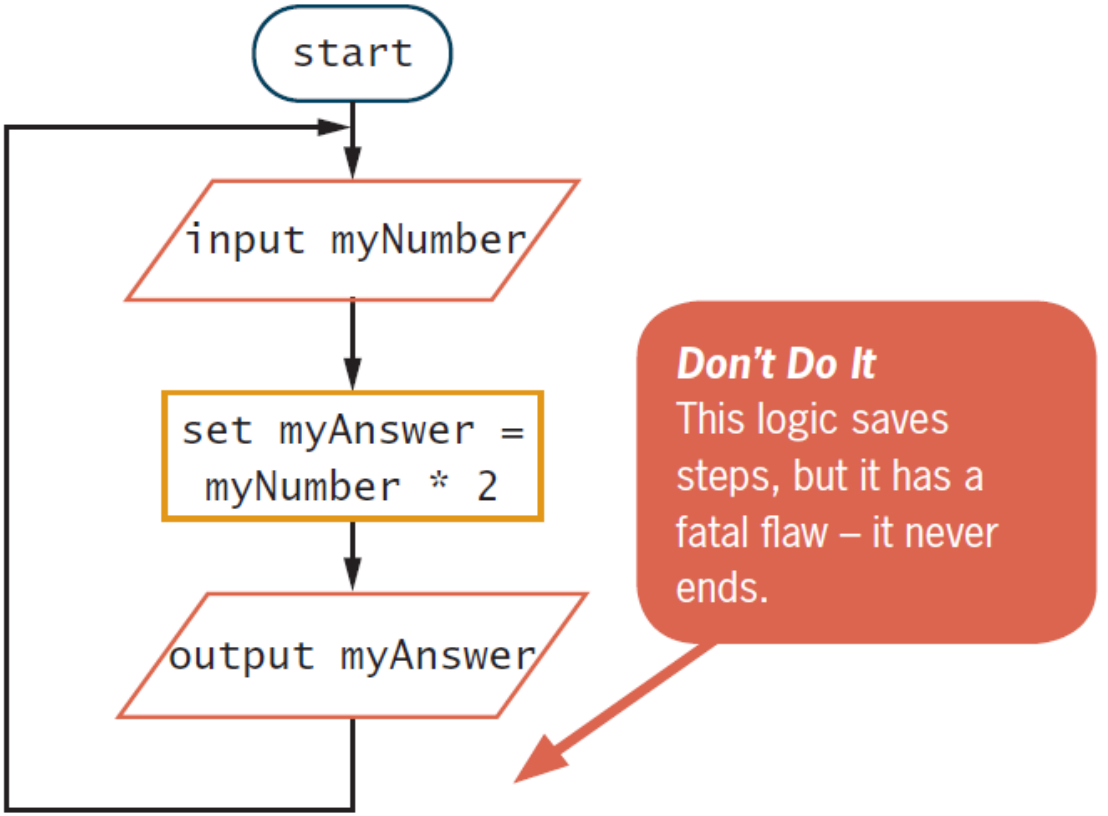
- input symbol: kí hiệu nhập
- processing symbol: kí hiệu xử lí
- output symbol: kí hiệu xuất
- flowline: kí hiệu đường dẫn
- terminal symbol: kí hiệu bắt đầu/kết thúc

Các Kí Hiệu của Flowchart



Lệnh Lặp

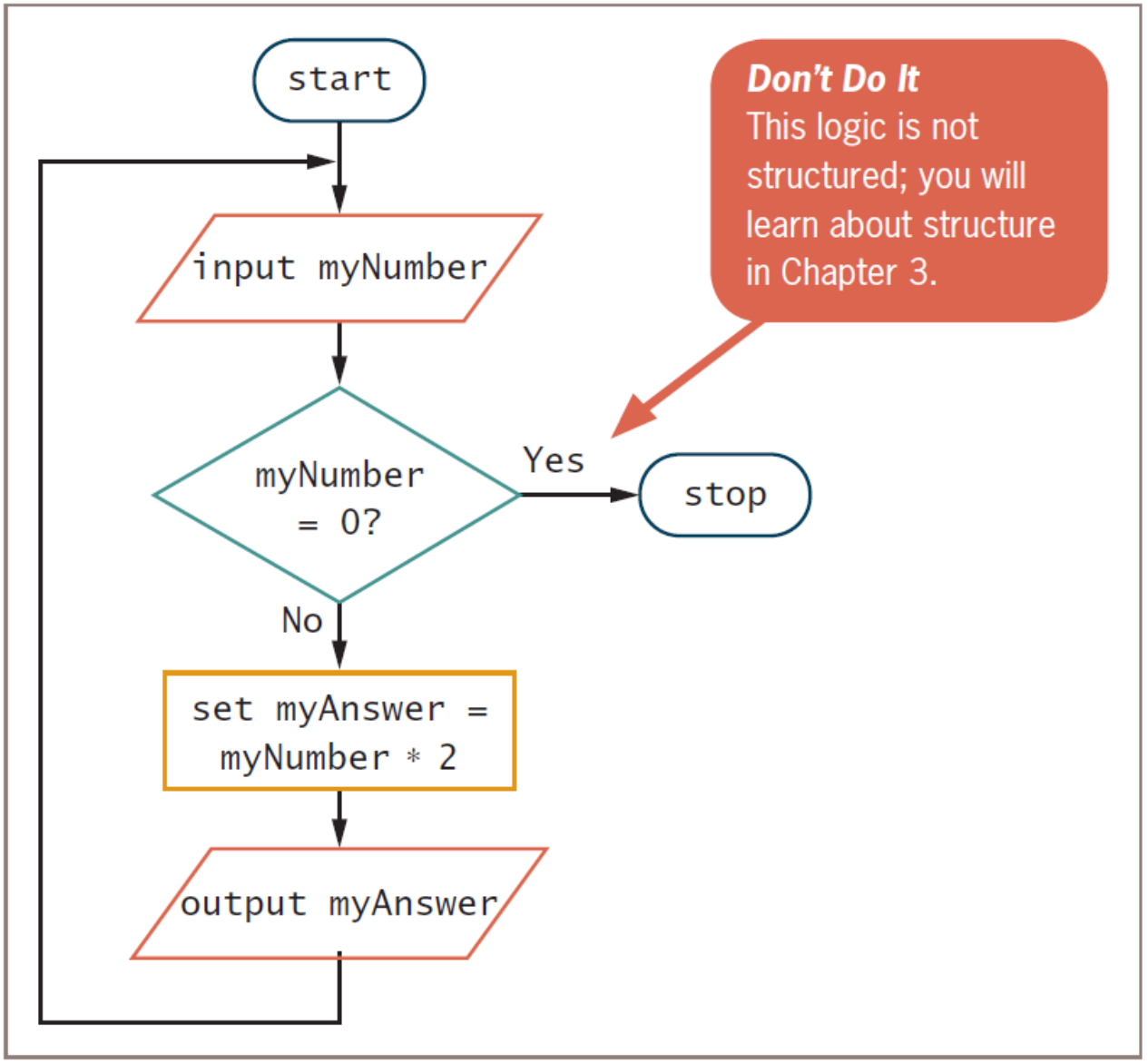
```
start
  input myNumber
  set myAnswer = myNumber * 2
  output myAnswer
  input myNumber
  set myAnswer = myNumber * 2
  output myAnswer
```



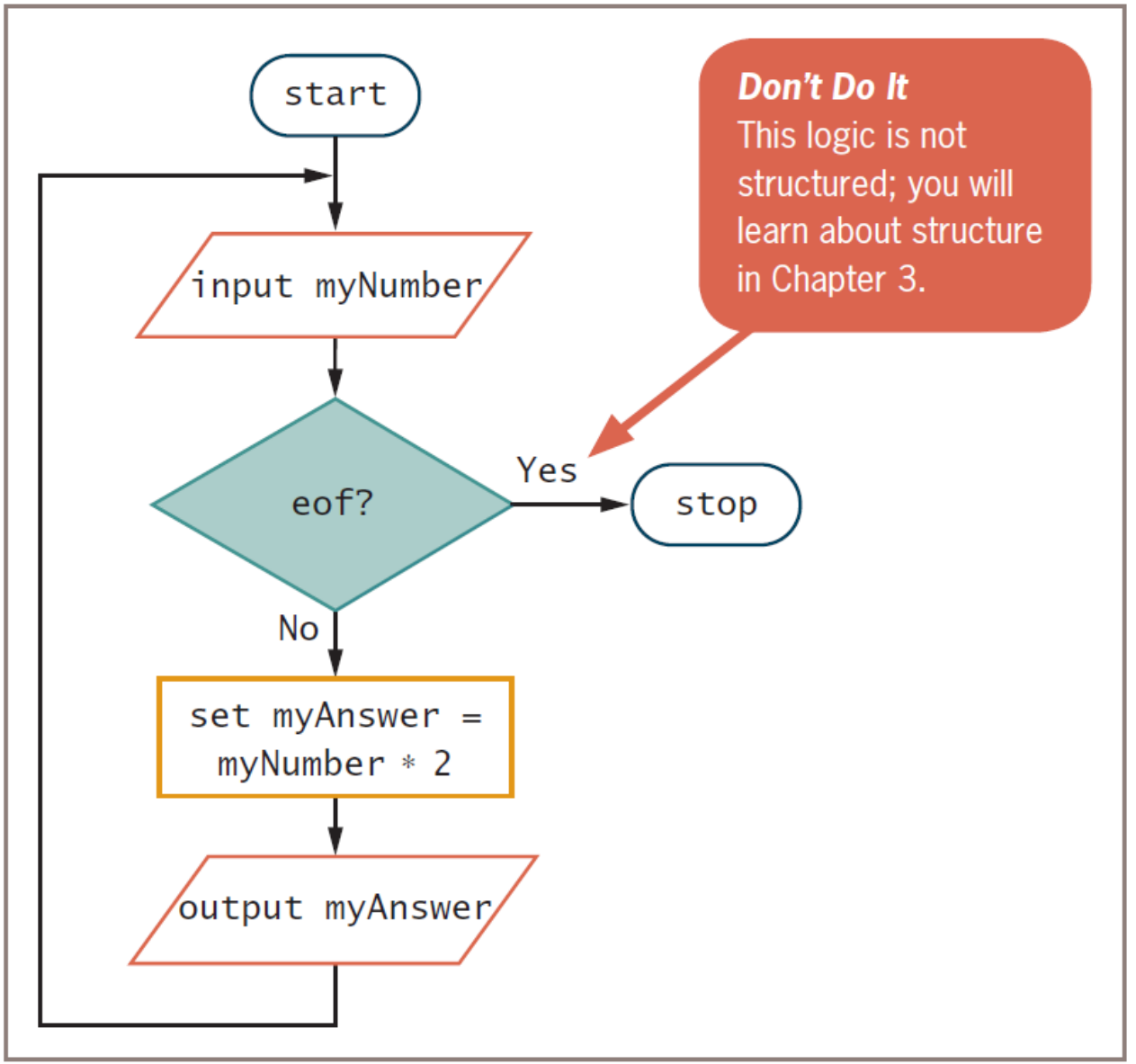
1.5 Sử dụng “lính gác” để kết thúc chương trình

- Decision symbol: kí hiệu quyết định / điều kiện
- Dummy/sentinel value: giá trị giả / lính gác
- Eof: điều kiện kết thúc

Vòng lặp sử dụng Sentinel Value = 0



Vòng lặp sử dụng eof



1.6 Sự Tiến Hóa của Mô Hình Lập Trình

- **Procedural programming:** lập trình cấu trúc / thủ tục tập trung trên các thủ tục.
- **Object-oriented programming:** lập trình hướng đối tượng tập trung trên đối tượng, và mô tả các thuộc tính và hành vi của chúng.

1.7 Tổng Kết

- Nhập, xử lí, xuất.
- Thuật toán, lỗi ngữ pháp, lỗi luận lí.
- Tìm hiểu bài toán, xây dựng thuật toán, viết chương trình, kiểm thử, thành phẩm, và bảo trì.
- Sơ đồ luồng, mã giả.
- Giá trị giả / lính gác / cờ.

1.8 Bài Tập - Xác định kí hiệu

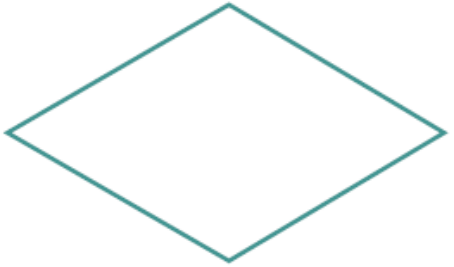
1. Input
2. Processing
3. Output
4. Decision
5. Terminal

A.

B.

C.

D.



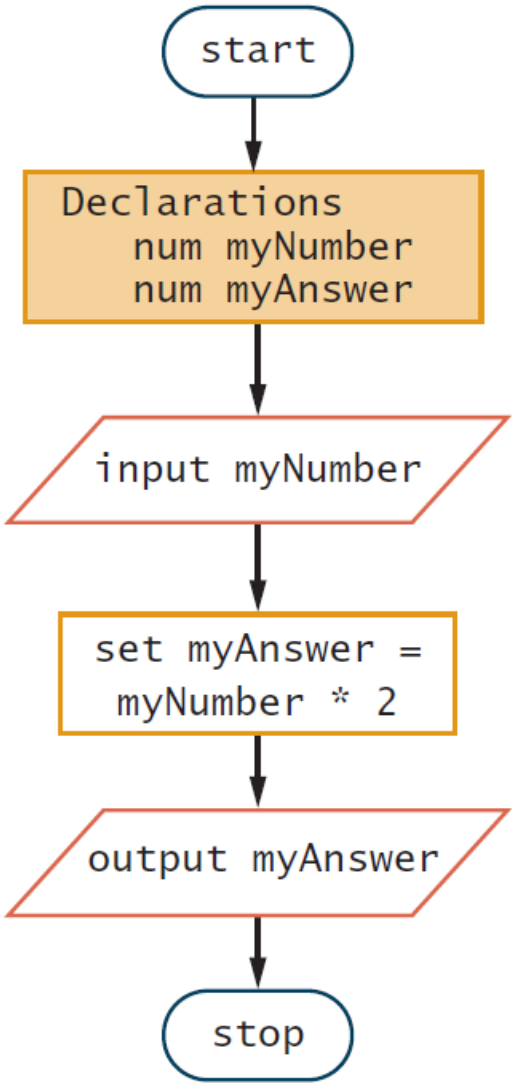
2. Các Yếu Tố của Chương Trình Chất Lượng Cao

- Khai báo, sử dụng biến và hằng
- Ưu điểm của phân đoạn chương trình
- Phân đoạn một chương trình
- Tạo sơ đồ phân cấp
- Đặc trưng của một chương trình được thiết kế tốt
- Tổng kết
- Bài tập

2.1 Khai Báo Biến và Hằng

- Numeric, string constant: hằng số, hằng chuỗi
- Variable: Biến
  - Declaration: Khai báo
    - identifier: Định danh / tên
    - data type: Kiểu dữ liệu
  - Garbage: Rác
- Các kiểu đặt tên biến:
  - Camel: **lastName**
  - Pascal: **LastName**
  - Hungarian: **stringLastName**
  - Snake: **last_name**
  - Mixed: **Last_Name**

Khai Báo Biến




```
start
  Declarations
     num myNumber
     num myAnswer
  input myNumber
  set myAnswer = myNumber * 2
  output myAnswer
stop
```

Qui Tắc Đặt Tên Biến

- Tên biến phải là một từ:
  - **interestRate**
- Tên biến phải là bắt đầu bằng một kí tự
- Tên biến phải có nghĩa:
  - **set interestEarned = initialInvestment * interestRate**

Gán Giá Trị cho Biến

- assignment statement: lệnh gán
  - **set myAnswer = myNumber** * 2
  - **set someNumber** = 2
  - **set someNumber** = 3 + 7
  - **set someOtherNumber = someNumber**
  - **set someOtherNumber = someNumber** * 5
- Trong một số ngôn ngữ lập trình:
  - **myAnswer = myNumber** * 2
  - **someNumber** = 2

Kiểu Dữ Liệu của Biến

- Giá trị số
  - **myAnswer** = **myNumber** * 2
  - **taxRate** = 2.5
- Chuỗi
  - **lastName** = "**Lincoln**"

Khai Báo Hằng

- named constant: tên hằng
  - **num SALES_TAX_RATE** = 0.06

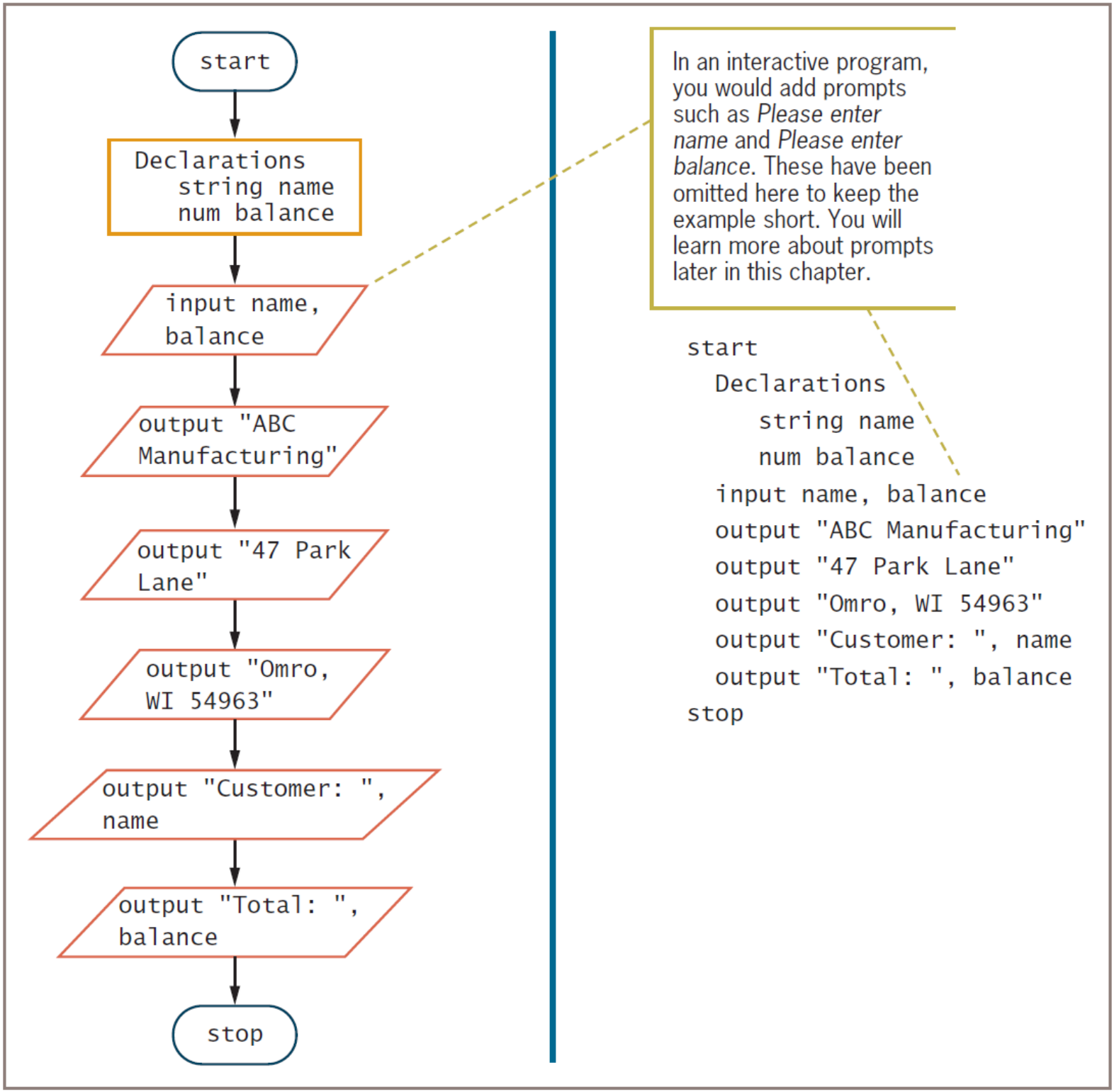
- tương đương:
  - **taxAmount** = **price** * 0.06
  - **taxAmount** = **price** * **SALES_TAX_RATE**

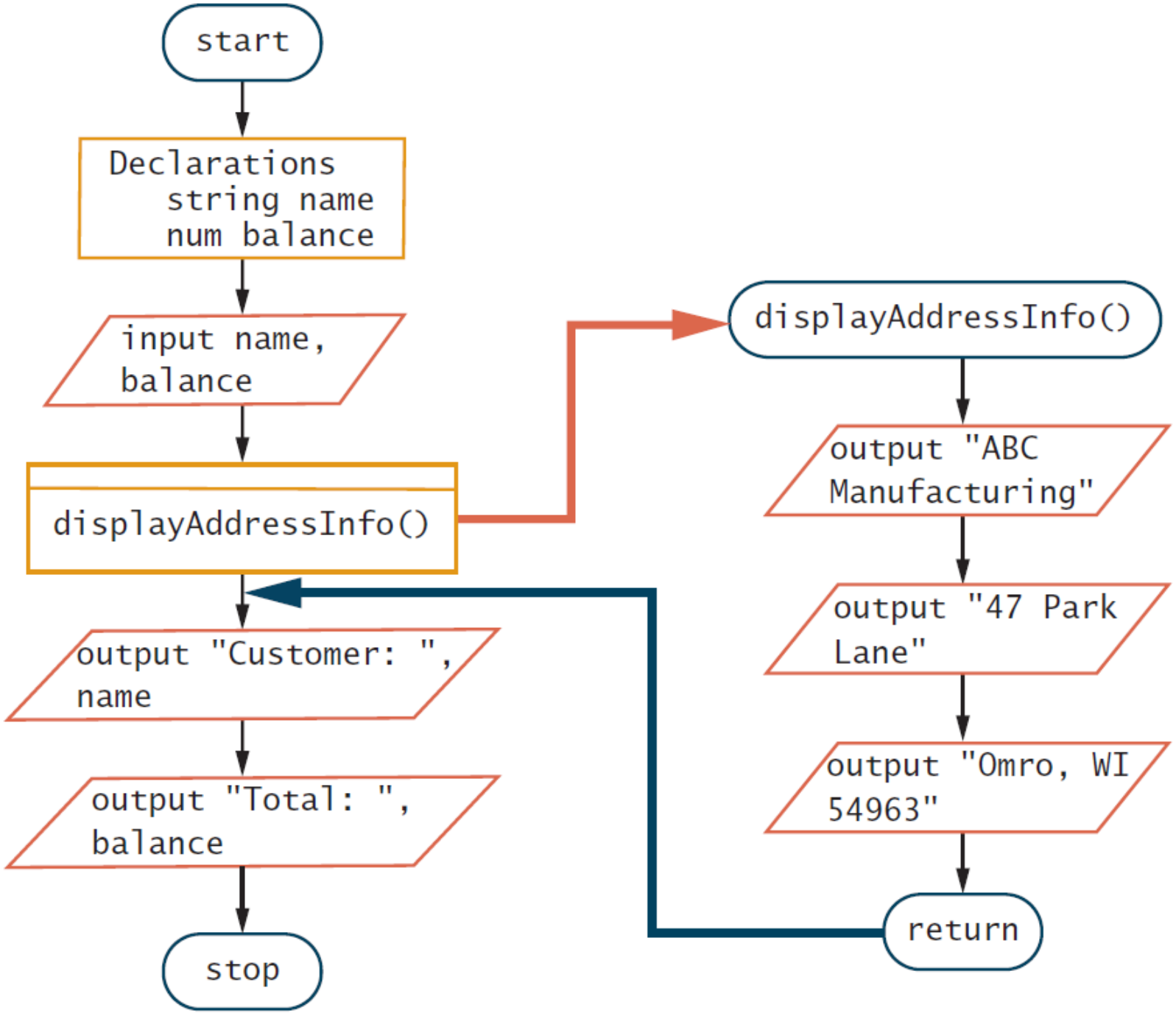
2.2 Ưu Điểm của Phân Đoạn Chương Trình

- Module, subroutine, procedure, function, method.
- Modularization, functional decomposition.
- Nguyên nhân:
  - cung cấp tính tổng quát của chương trình
  - cho phép nhiều lập trình viên cùng làm chung một bài toán
  - dễ dàng sử dụng lại mã nguồn:
    - Reusability
    - Reliability

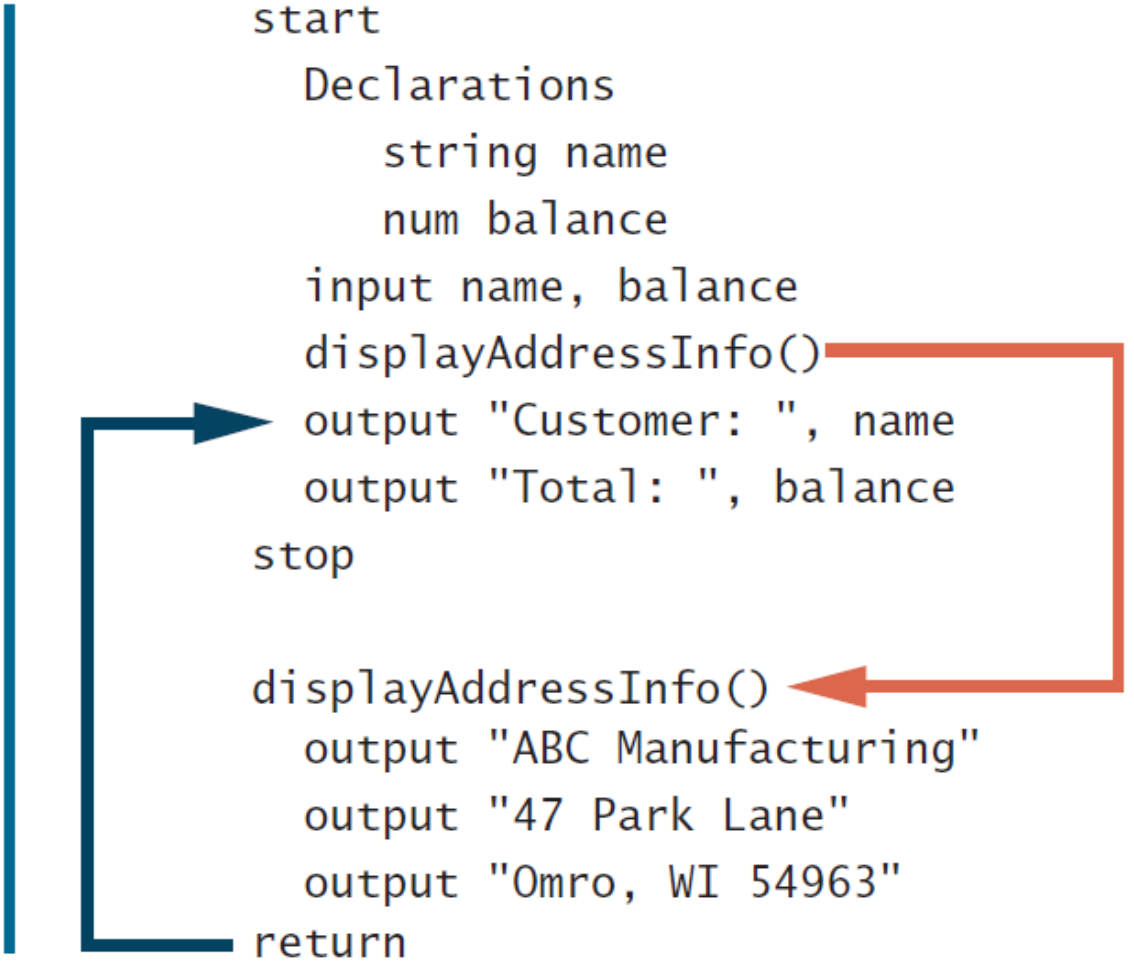
2.3 Phân Đoạn Một Chương Trình

- Chương trình chính bao gồm các bước cơ bản (mainline logic), sau đó gọi đến các phân đoạn (module) để cung cấp các bước chi tiết.
- Tạo một module:
  - Header: Phần đầu của module
  - Body: Phần thân của module
  - Lệnh **return**: Kết thúc module


start
Declarations
string name
num balance
input name, balance
output "ABC Manufacturing"
output "47 Park Lane"
output "Omro, WI 54963"
output "Customer: ", name
output "Total: ", balance
stop
In an interactive program, you would add prompts such as *Please enter name* and *Please enter balance*. These have been omitted here to keep the example short. You will learn more about prompts later in this chapter.
start
Declarations
string name
num balance
input name, balance
output "ABC Manufacturing"
output "47 Park Lane"
output "Omro, WI 54963"
output "Customer: ", name
output "Total: ", balance
stop


start
Declarations
string name
num balance
input name,
balance
displayAddressInfo()
output "Customer: ",
name
output "Total: ",
balance
stop
displayAddressInfo()
output "ABC
Manufacturing"
output "47 Park
Lane"
output "Omro, WI
54963"
return

Chương Trình Chính Gọi Module displayAddressInfo()



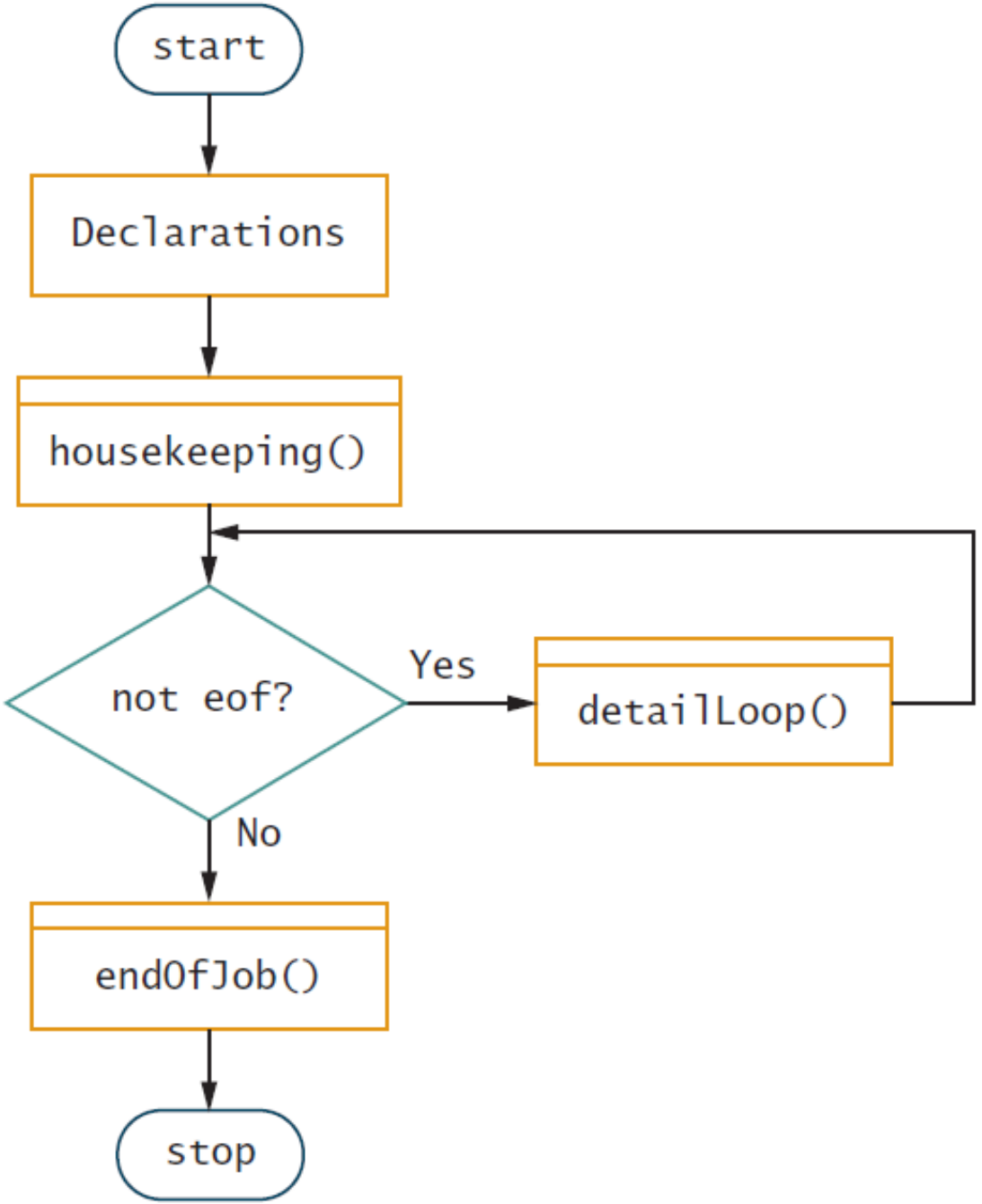
Khai Báo Biến và Hằng trong Module

```
start
  Declarations
     string name
     num balance
  input name, balance
  displayAddressInfo()
  output "Customer: ", name
  output "Total: ", balance
stop

displayAddressInfo()
  Declarations
     string LINE1 = "ABC Manufacturing"
     string LINE2 = "47 Park Lane"
     string LINE3 = "Omro, WI 54963"
  output LINE1
  output LINE2
  output LINE3
return
```

Cấu trúc thông dụng của một chương trình chính

- **Housekeeping**: thực thi khi bắt đầu chương trình
  - khai báo biến
  - hiển thị trợ giúp, hướng dẫn
  - hiển thị tiêu đề của báo cáo
  - mở tập tin
  - nhập liệu các dữ liệu đầu tiên, ..
- **Detail loop**: làm công việc chính, lặp đi lặp lại cho đến khi không còn dữ liệu để xử lí.
- **End-of-job**: hiển thị tổng kết hay thông điệp kết thúc và đóng các tập tin đã mở.




```
start
  Declarations
  housekeeping()
  while not eof
     detailLoop()
  endwhile
  endOfJob()
stop
```

Ví dụ: Bảng kê lương

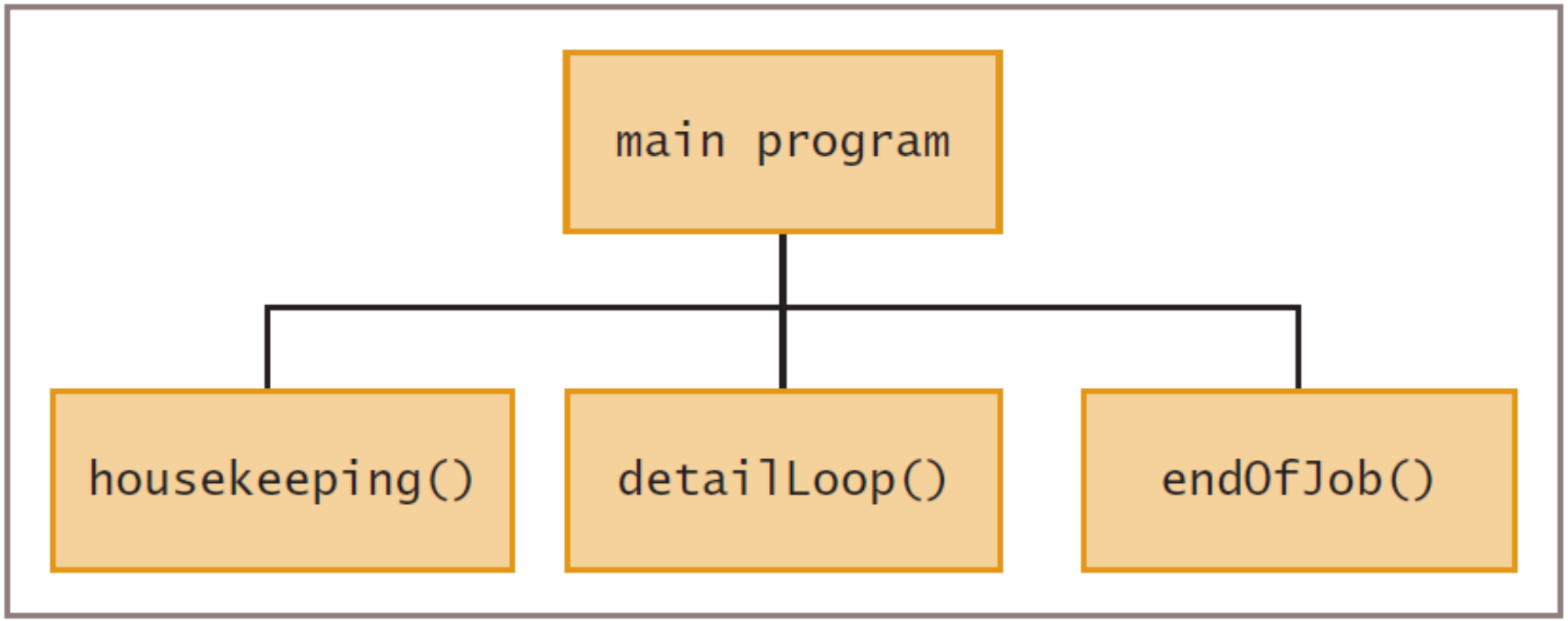
Payroll Report

Name	Gross	Deductions	Net
Andrews	1000.00	250.00	750.00
Brown	1400.00	350.00	1050.00
Carter	1275.00	318.75	956.25
Young	1100.00	275.00	825.00

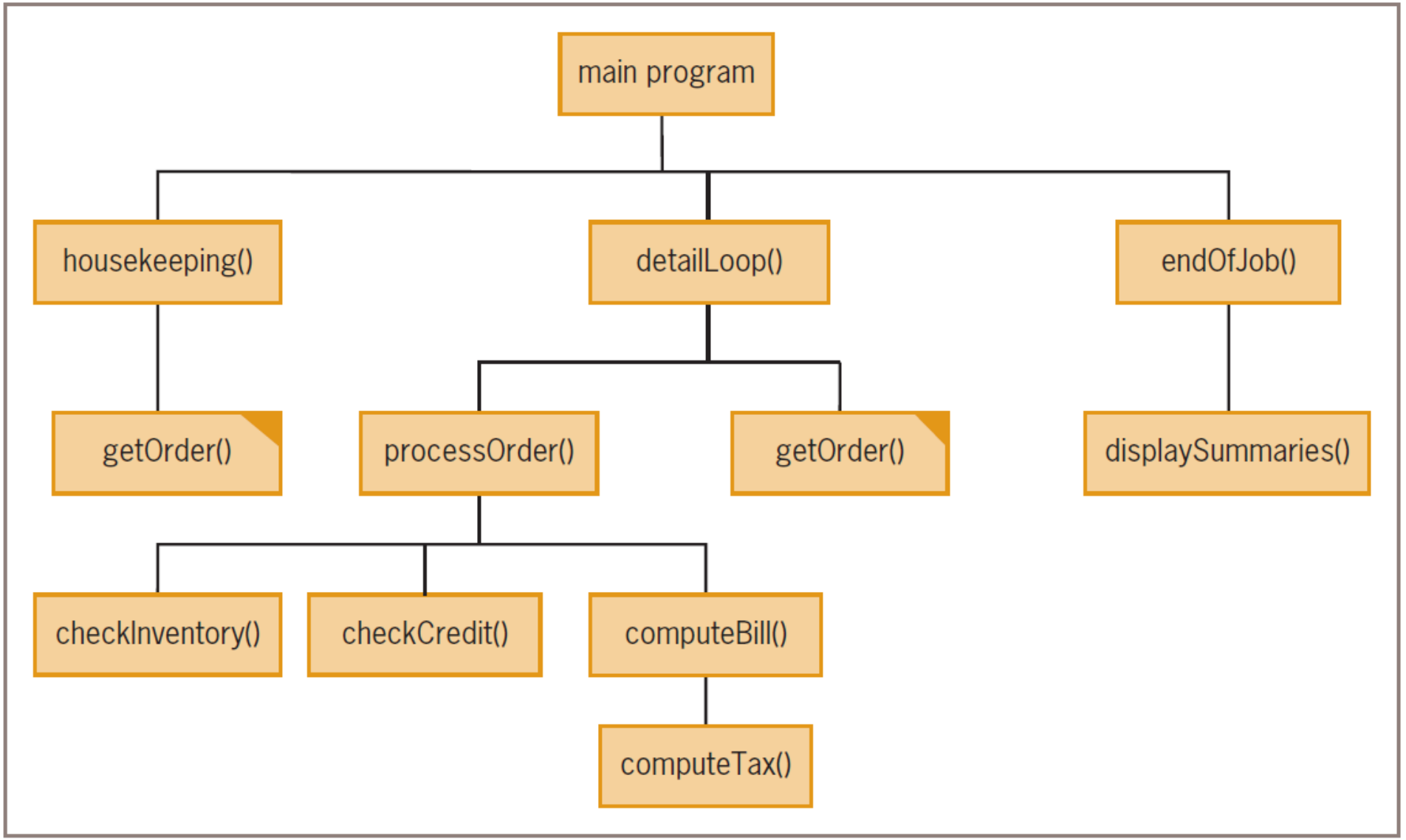
***End of report

2.4 Tạo Sơ Đồ Phân Cấp

- Sơ đồ phân cấp mô tả module nào tồn tại trong một chương trình và module nào gọi module khác.



Sơ đồ phân cấp của chương trình tính hóa đơn



2.5 Đặc Trưng của một Chương Trình Được Thiết Kế Tốt

- Cung cấp chú thích (comment)
- Chọn định danh kĩ càng
- Thiết kế lệnh rõ ràng trong chương trình và module
- Viết tường minh lời hướng dẫn khi nhập liệu
- Tiếp tục duy trì thói quen lập trình tốt đồng thời khi phát triển kĩ năng lập trình.

2.6 Tổng Kết

- Dữ liệu trong chương trình có 3 dạng: literal, variable, và named constant.
- Lập trình viên chia chương trình thành nhiều đơn vị nhỏ: module, subroutine, procedure, function, hay method.
- Một module bao gồm: header, body, và lệnh **`return`** .
- Sơ đồ phân cấp mô tả module và mối liên hệ giữa chúng.
- Khi chương trình trở nên lớn và phức tạp hơn, yêu cầu thiết kế một chương trình tốt là cần thiết.

2.7 Bài Tập

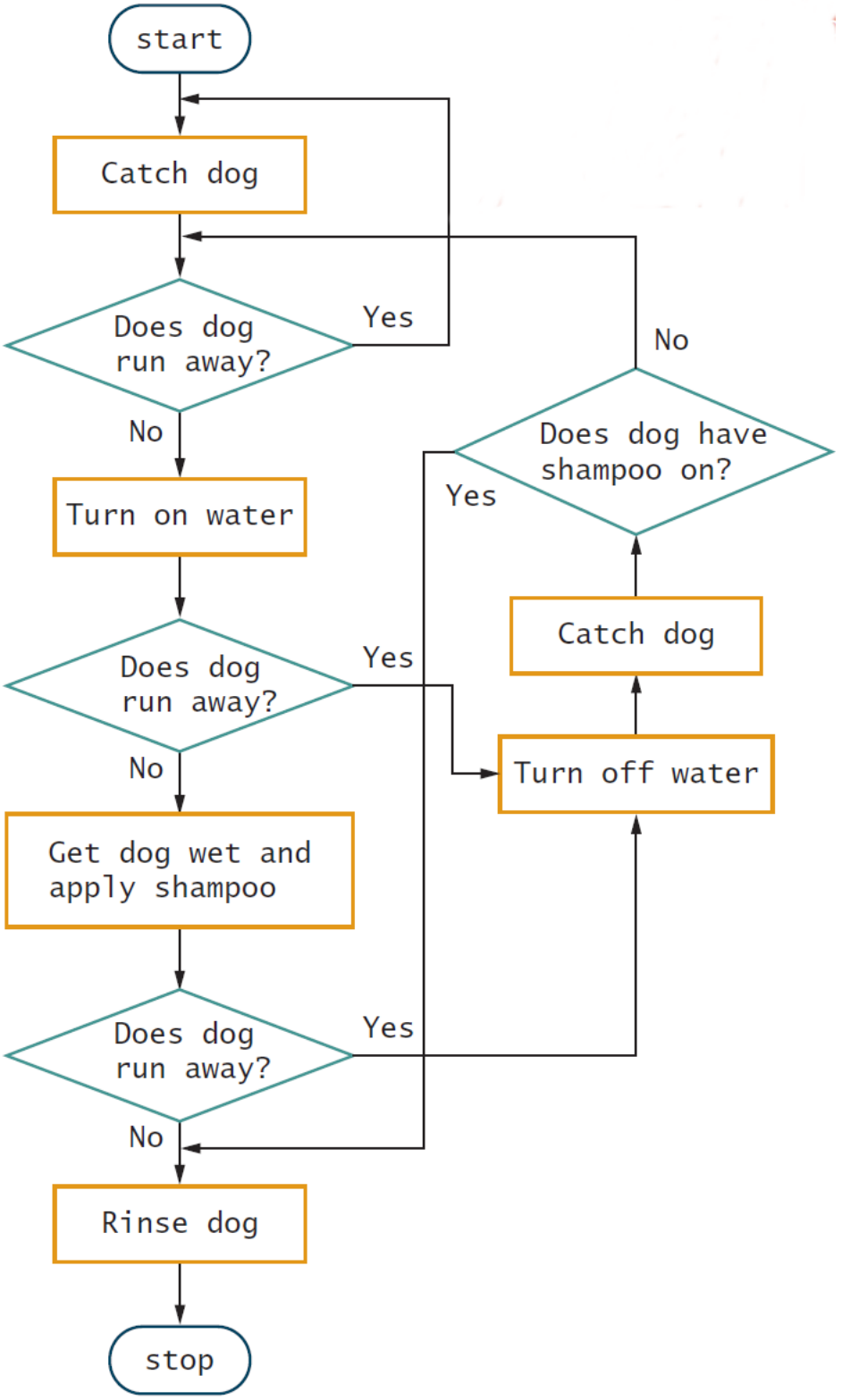
- Vẽ sơ đồ phân cấp và xây dựng logic cho một chương trình tính toán chỉ số BMI của một người:
  - BMI so sánh cân nặng và chiều cao
  - Sử dụng 3 module:
    - Module đầu tiên cho phép nhập chiều cao ở dạng inch
    - Module thứ hai cho phép nhập cân nặng ở dạng pound, rồi chuyển cân nặng và chiều cao sang dạng kilogram và meter. Sau đó, tính BMI và hiển thị kết quả.
    - Module cuối cùng hiển thị thông điệp **End of Job**.
- Thay đổi chương trình xác định BMI ở trên cho phép thực thi liên tục cho đến khi người dùng nhập vào chiều cao 0.

3. Lập trình cấu trúc

- Nhược điểm của lập trình không cấu trúc
- Ba cấu trúc cơ bản
- Sử dụng kĩ thuật Priming Input
- Lý do để cấu trúc chương trình
- Nhận diện cấu trúc
- Cấu trúc và phân đoạn chương trình
- Tổng kết
- Bài tập

3.1 Nhược điểm của mã không cấu trúc

- Mã không cấu trúc: unstructured spaghetti code.
- có thể tạo kết quả sai
- khó hiểu và bảo trì
- thuật toán khó hiểu
- dễ sai
- khó tái sử dụng

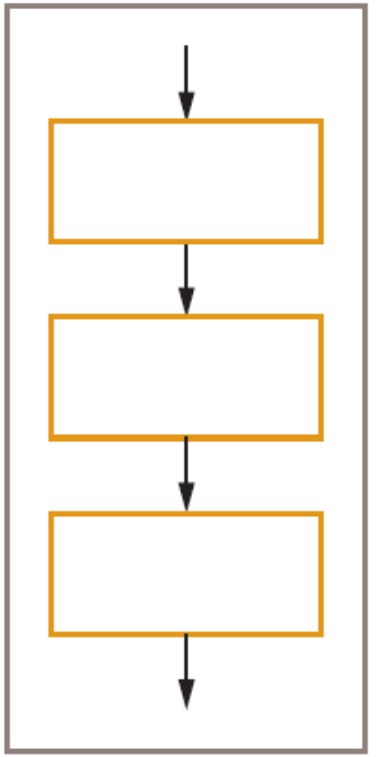

start
Catch dog
Does dog run away?
Yes
No
Turn on water
Does dog run away?
Yes
No
Get dog wet and apply shampoo
Does dog run away?
Yes
No
Rinse dog
stop
Does dog have shampoo on?
No
Yes
Catch dog
Turn off water

3.2 Ba cấu trúc cơ bản

- Bất cứ chương trình đều có thể xây dựng từ việc sử dụng một hay nhiều các cấu trúc sau đây:
  - tuần tự - sequence
  - rẽ nhánh - selection
  - lặp - loop
- Cấu trúc là một đơn vị cơ bản của thuật toán chương trình.

Cấu trúc tuần tự - Sequence Structure

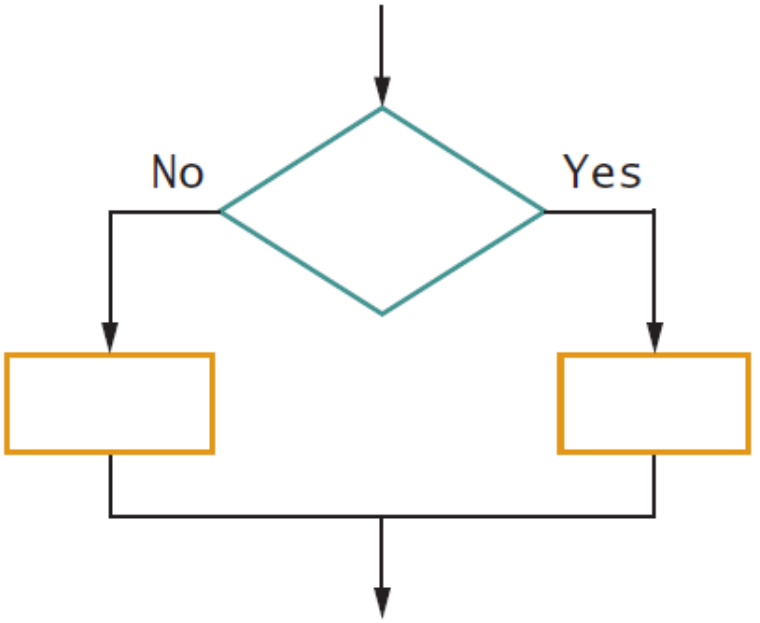
- Cấu trúc tuần tự có thể có nhiều hành động, nhưng không thể rẽ nhánh hay bỏ qua bất cứ hành động nào.



Cấu trúc rẽ nhánh - Selection Structure

- Một trong 2 nhánh hành động dựa trên câu hỏi điều kiện.
- Sơ đồ mô tả cấu trúc rẽ nhánh bắt đầu biểu tượng hình kim cương và các nhánh phải kết hợp ở cuối cấu trúc.

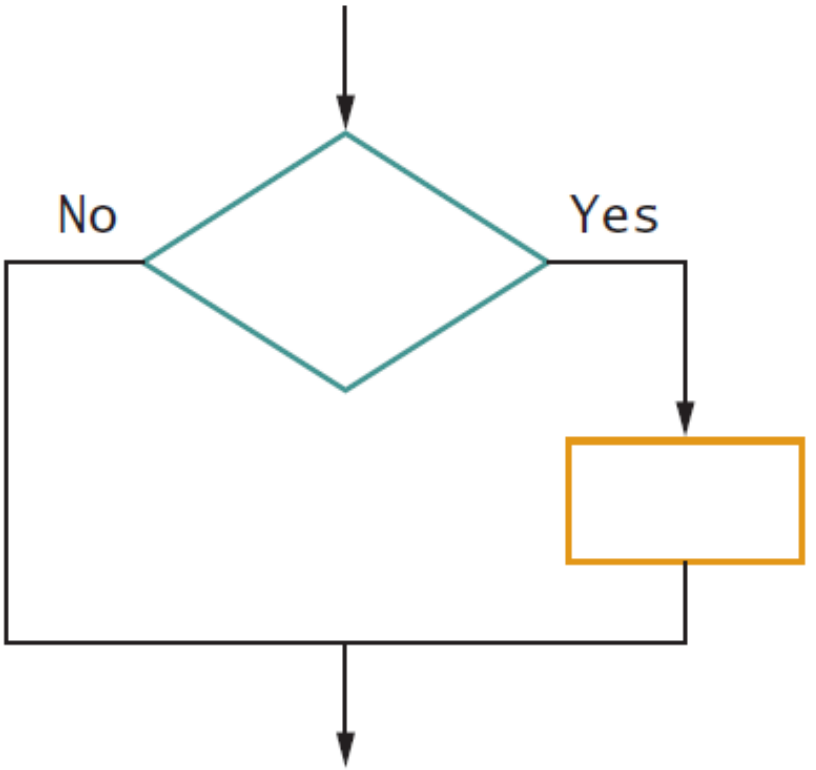
```
if someCondition is true then
   do oneProcess
else
   do theOtherProcess
endif
```



Cấu trúc rẽ 1 nhánh

- Một nhánh có thể không làm hành động gì.

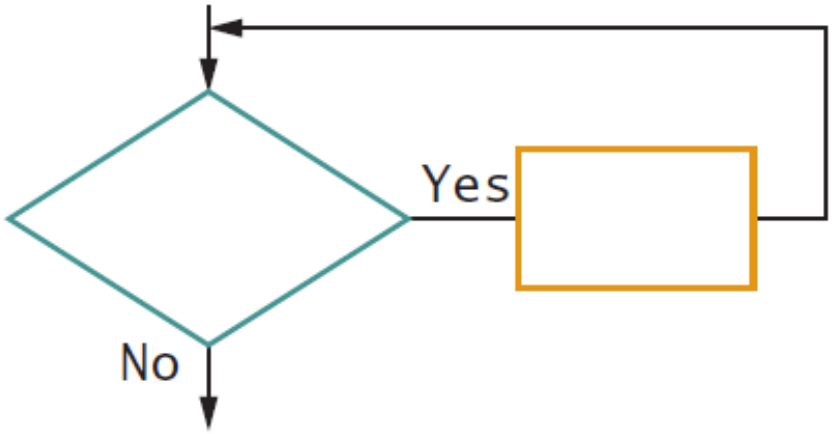
```
if it is raining then
   take an umbrella
endif
```



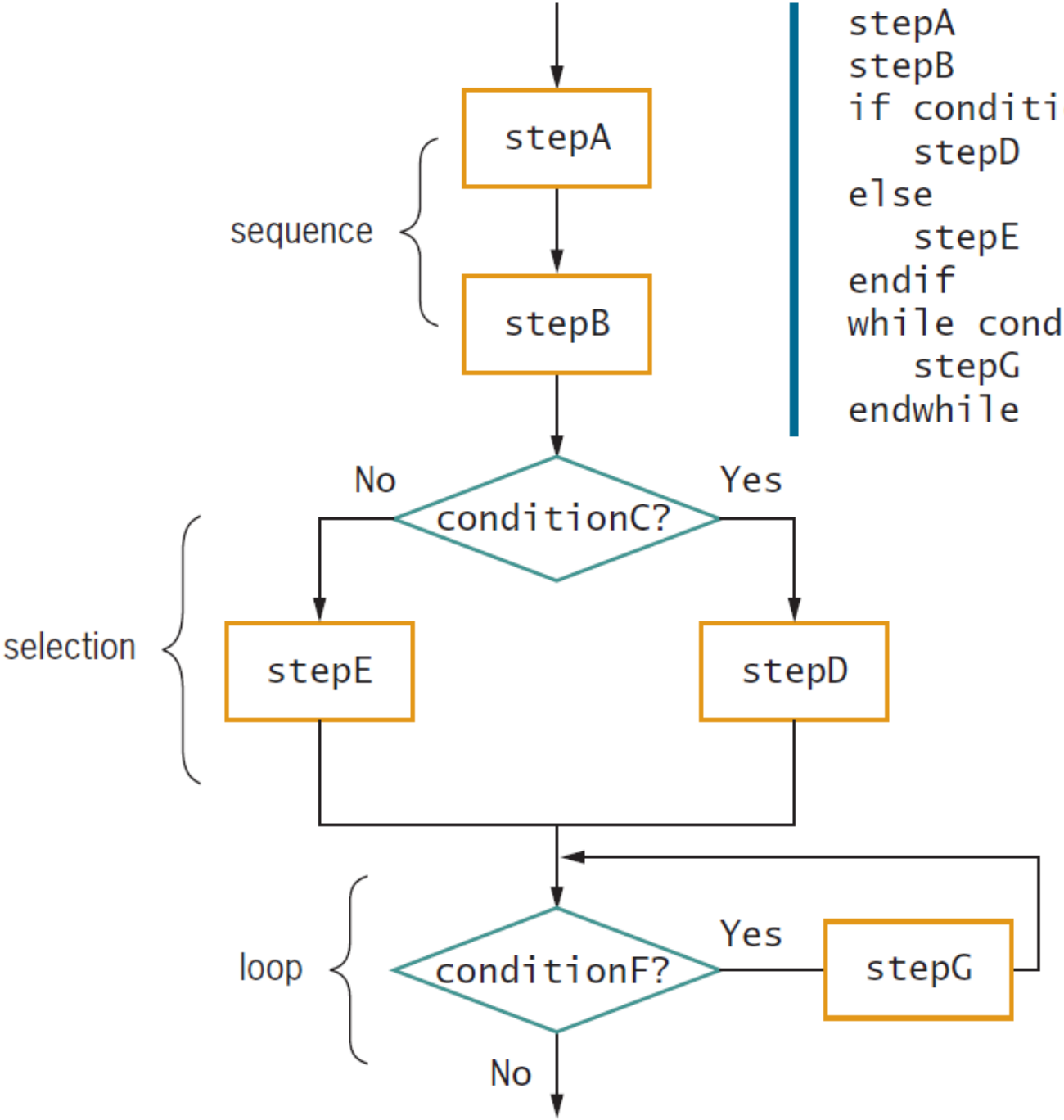
Cấu trúc lặp - Loop Structure

- Vòng lặp tiếp tục thực hiện hành động khi điều kiện đúng.

```
while testCondition continues to be true do
    someProcess
endwhile
```

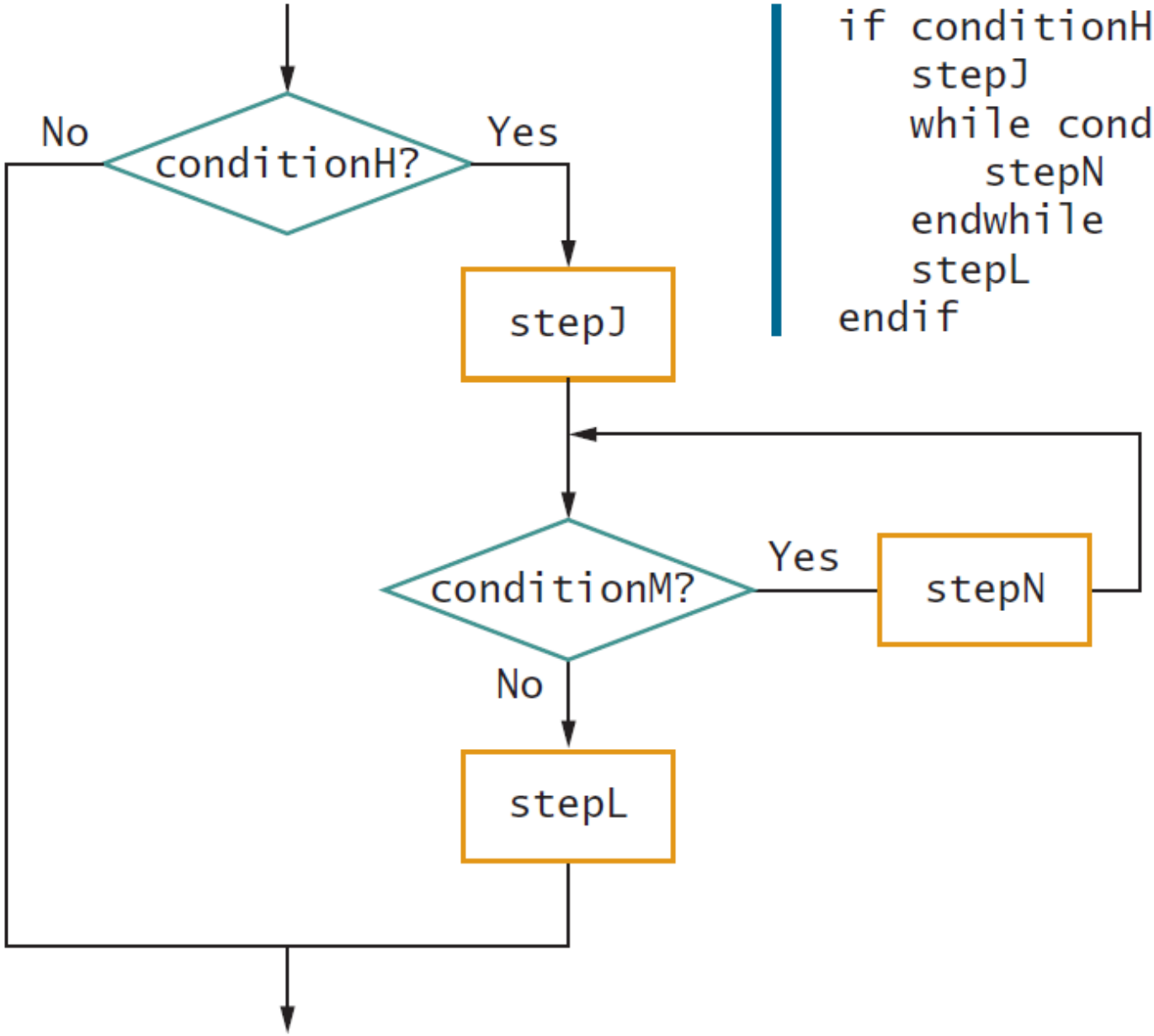


Kết hợp cấu trúc - Cấu trúc chồng nhau (stacking structures)

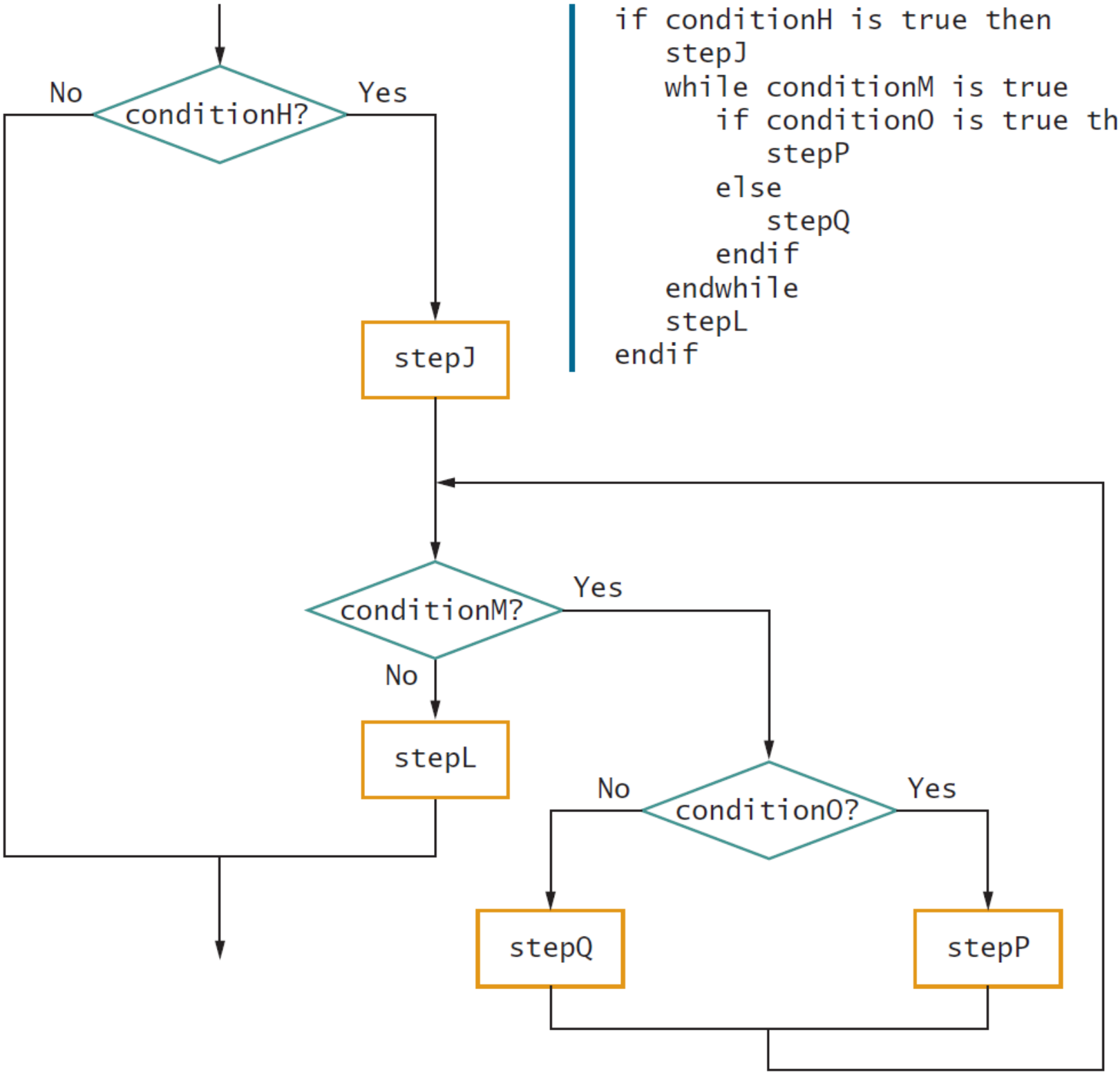



```
stepA
stepB
if conditionC is true then
   stepD
else
   stepE
endif
while conditionF is true
   stepG
endwhile
```

Kết hợp cấu trúc - Cấu trúc lồng nhau (nesting structures)

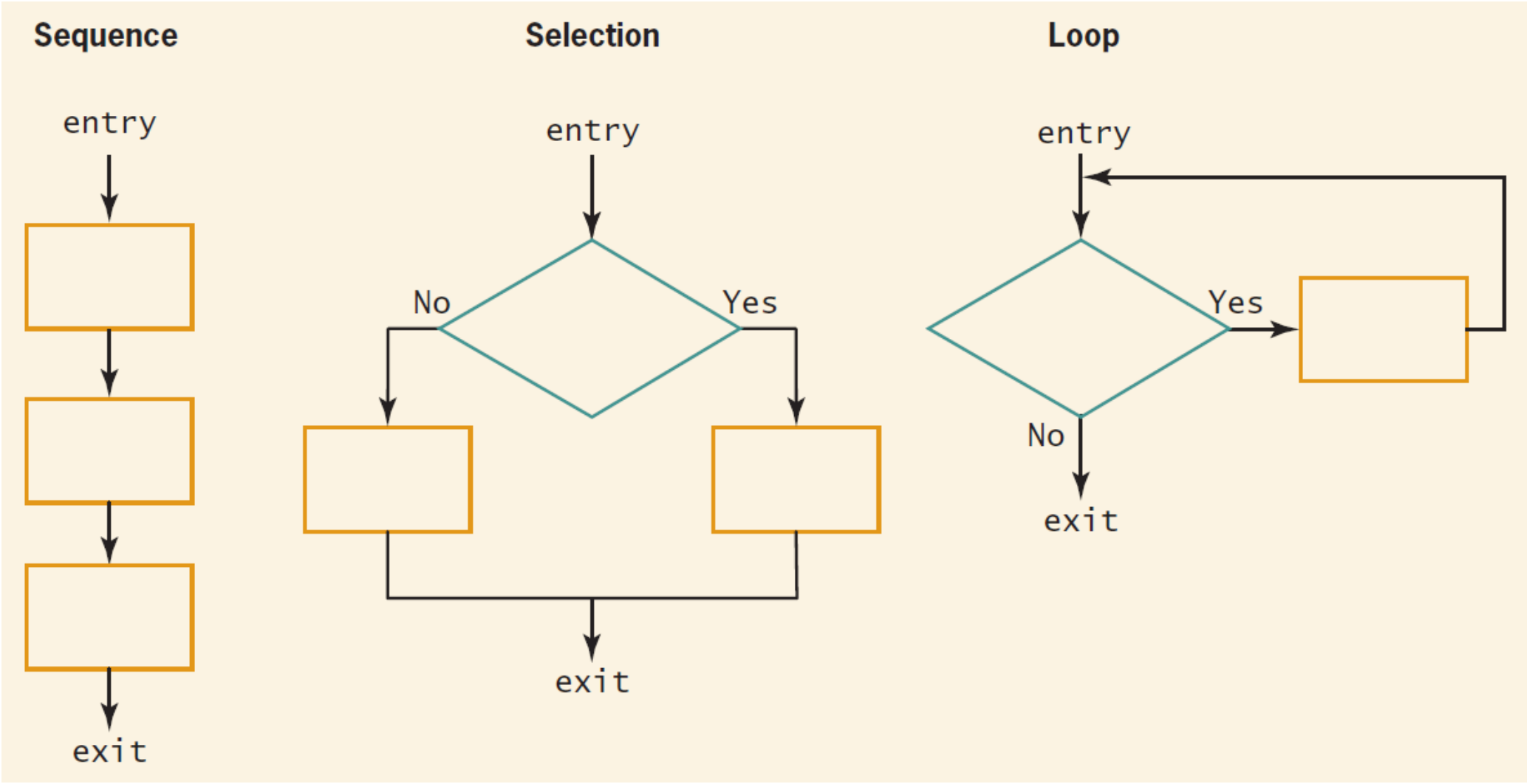



```
if conditionH is true then
   stepJ
   while conditionM is true
      stepN
   endwhile
   stepL
endif
```

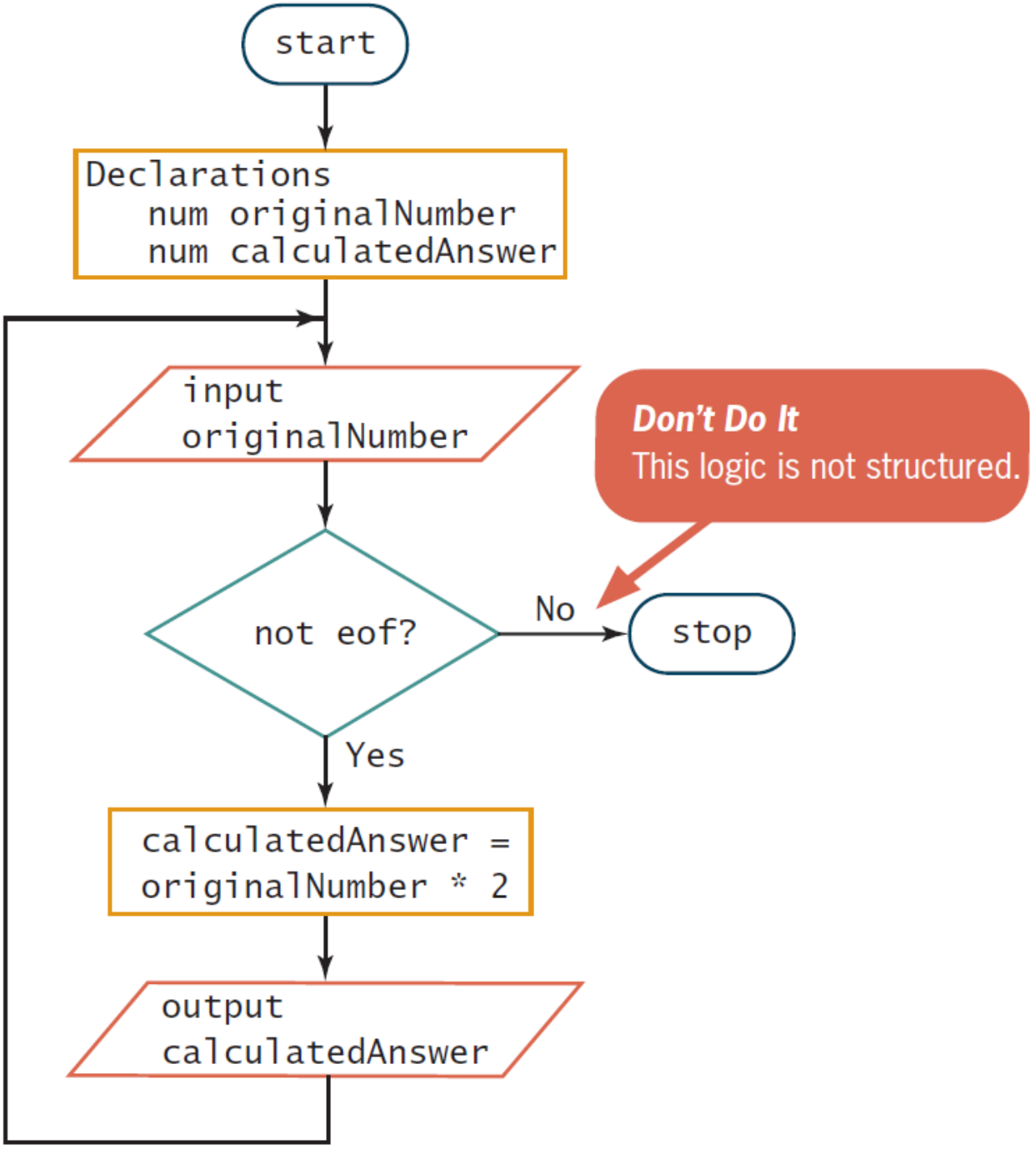



```
if conditionH is true then
   stepJ
   while conditionM is true
      if conditionO is true then
         stepP
      else
         stepQ
      endif
   endwhile
   stepL
endif
```

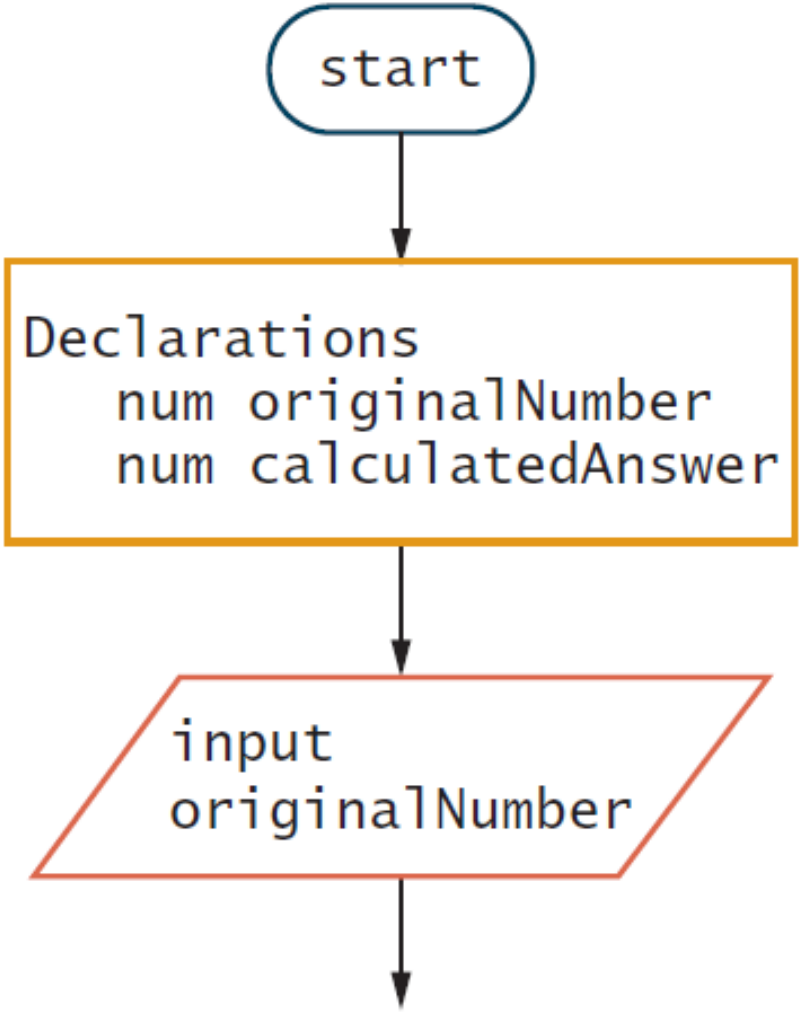
Tham chiếu nhanh 3 cấu trúc



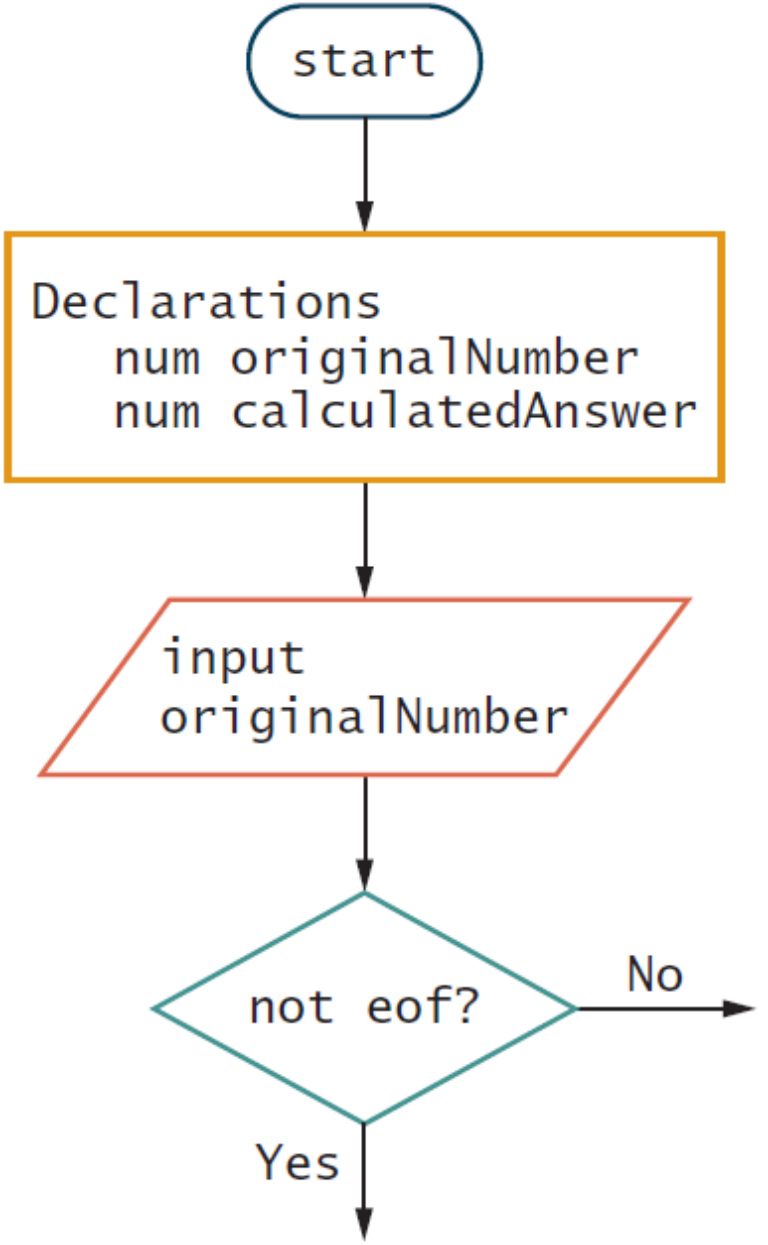
3.3 Dùng kĩ thuật Priming Input để cấu trúc chương trình

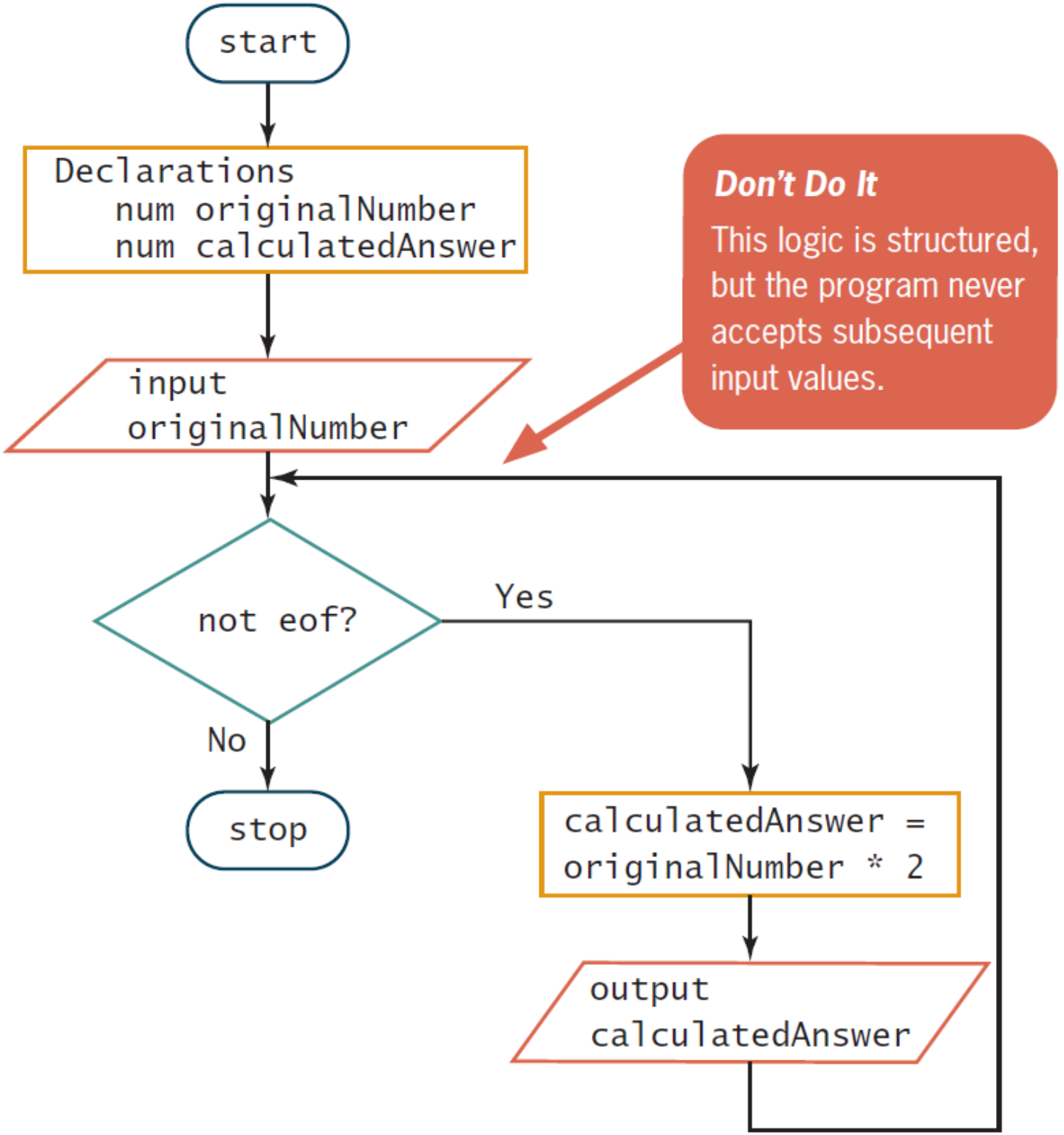


- Phần sơ đồ sau có cấu trúc hay không?



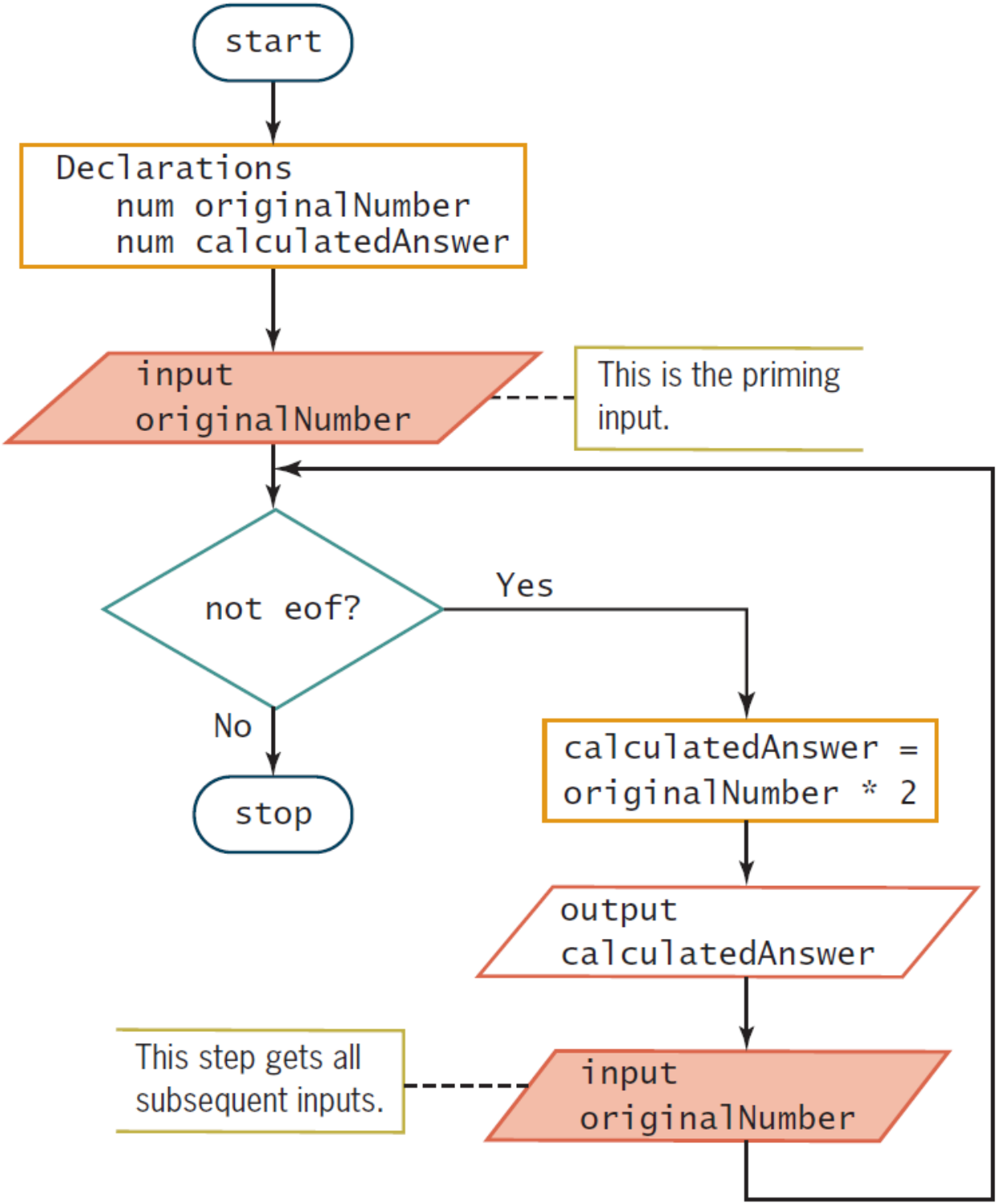
- Trong lệnh lặp có cấu trúc, qui luật thứ tự như sau:
  - Câu hỏi điều kiện
  - Nếu câu hỏi xác định đúng thì thực thi phần thân vòng lặp.
  - Sau khi thực thi phần thân, phải lập tức quay lại câu hỏi lần nữa.



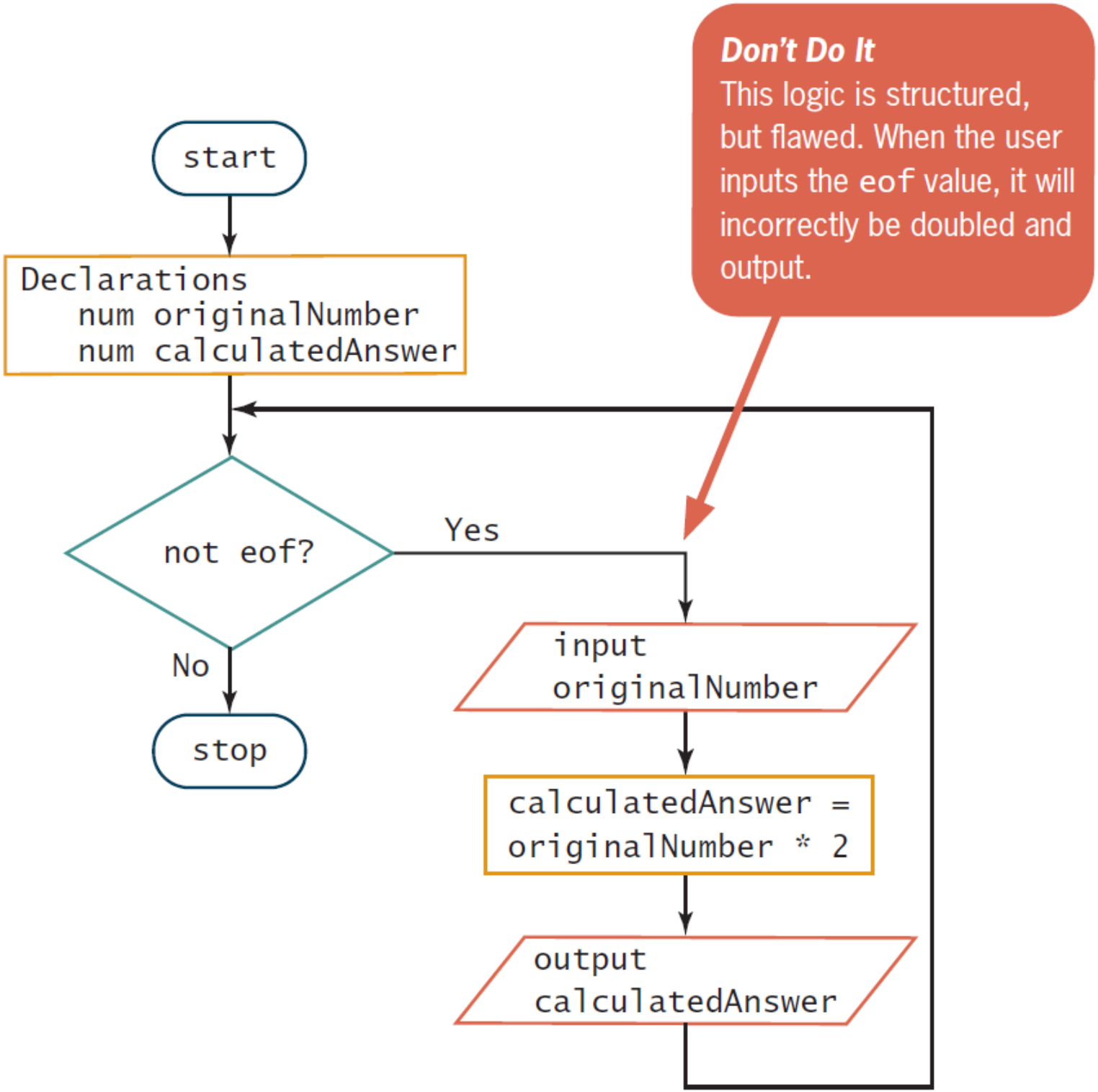

start
Declarations
num originalNumber
num calculatedAnswer
input
originalNumber
not eof?
Yes
No
stop
calculatedAnswer =
originalNumber * 2
output
calculatedAnswer
Don't Do It
This logic is structured, but the program never accepts subsequent input values.

có cấu trúc và đúng?

- **priming input (priming read)** là một lệnh thêm vào để nhập giá trị lần đầu tiên.



Có cấu trúc nhưng sai

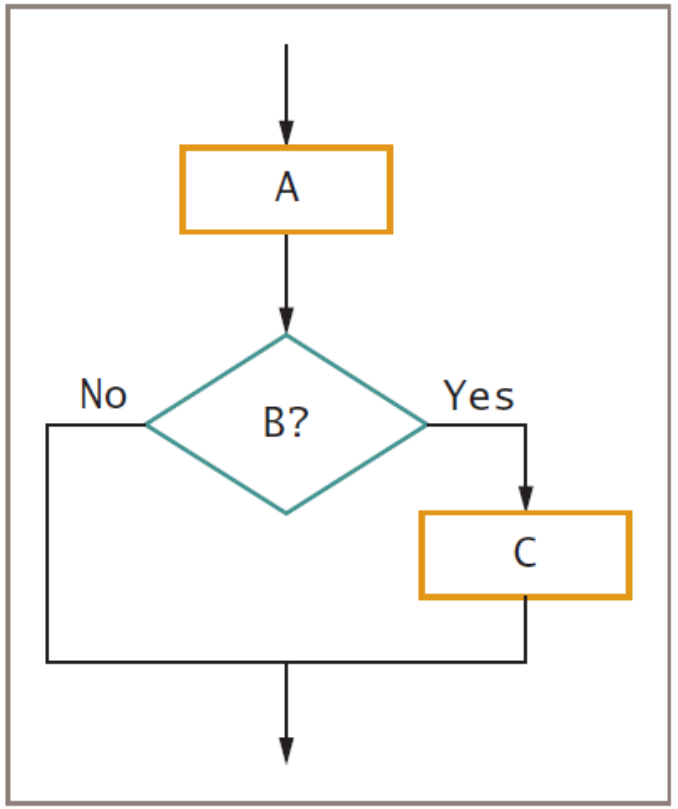


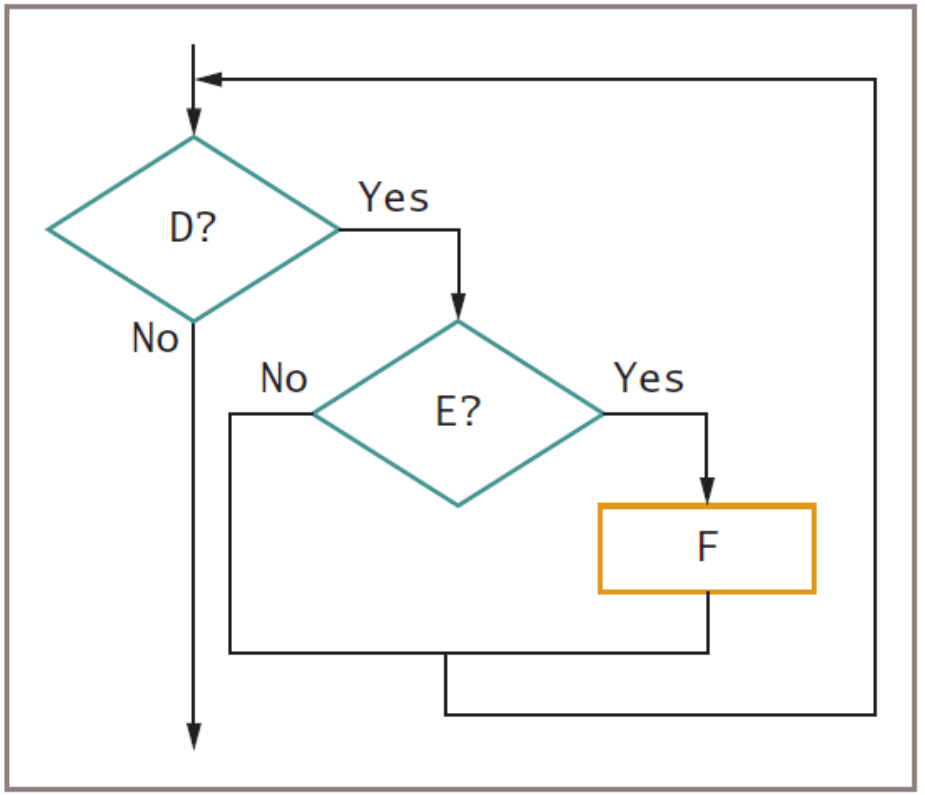
3.4 Các nguyên do của việc cấu trúc chương trình

- Clarity—Rõ ràng: khi chương trình phát triển lớn hơn, nếu nó không được cấu trúc thì chúng ta dễ dàng bị nhầm lẫn.
- Professionalism—Chuyên nghiệp: Tất cả lập trình viên đều đòi hỏi chương trình phải được cấu trúc.
- Efficiency—Hiệu quả: Hầu hết ngôn ngữ lập trình hỗ trợ và sử dụng cấu trúc.
- Maintenance—Bảo trì: Bạn và các lập trình viên khác dễ dàng thay đổi và bảo trì chương trình khi có sự thay đổi trong tương lai.
- Modularity—Phân đoạn: Chương trình có cấu trúc dễ dàng tách thành các thủ tục và gán cho từng lập trình viên hiện thực.

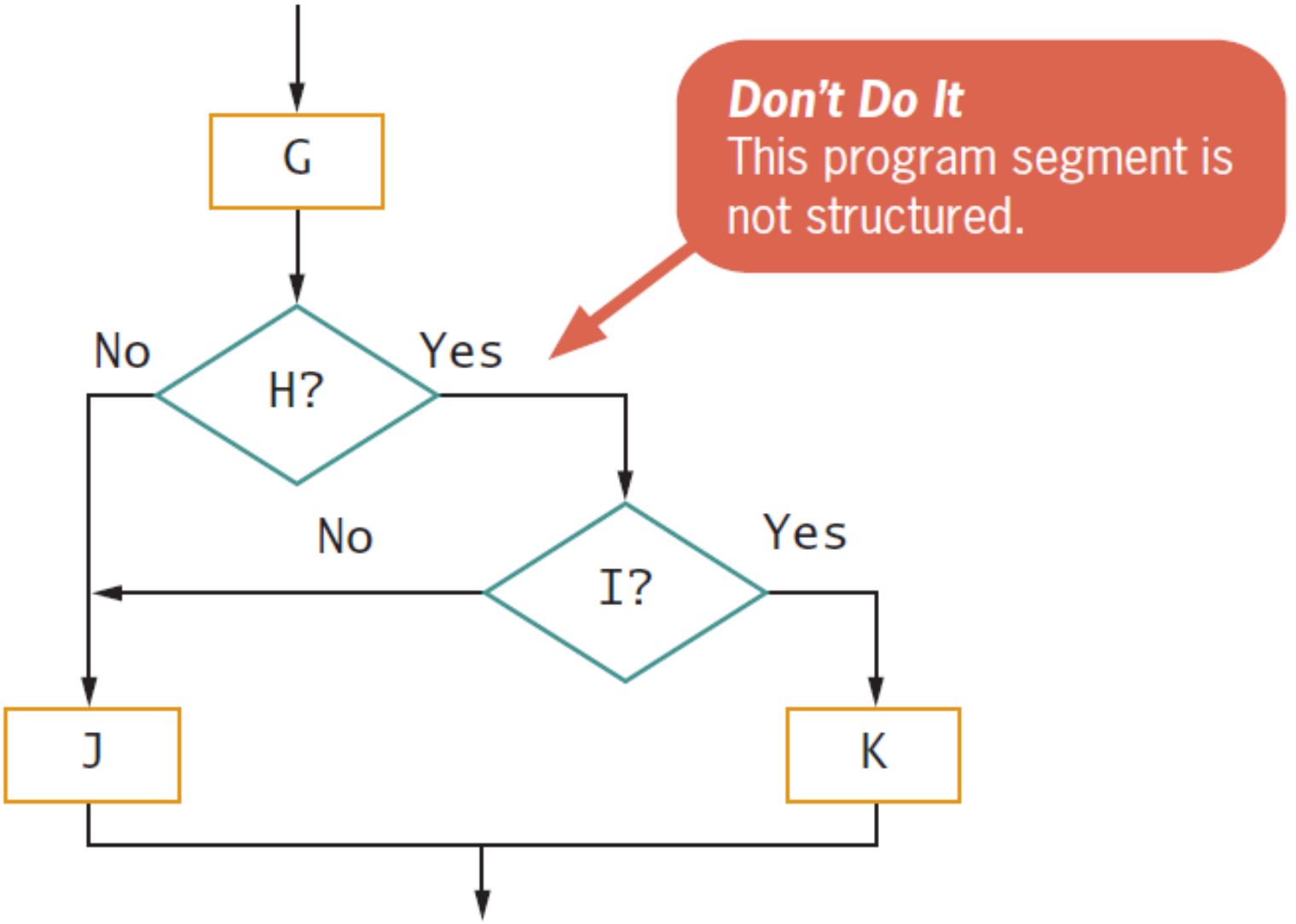
3.5 Nhận diện cấu trúc

- Rất khó để xác định sơ đồ của một chương trình có cấu trúc hay không.

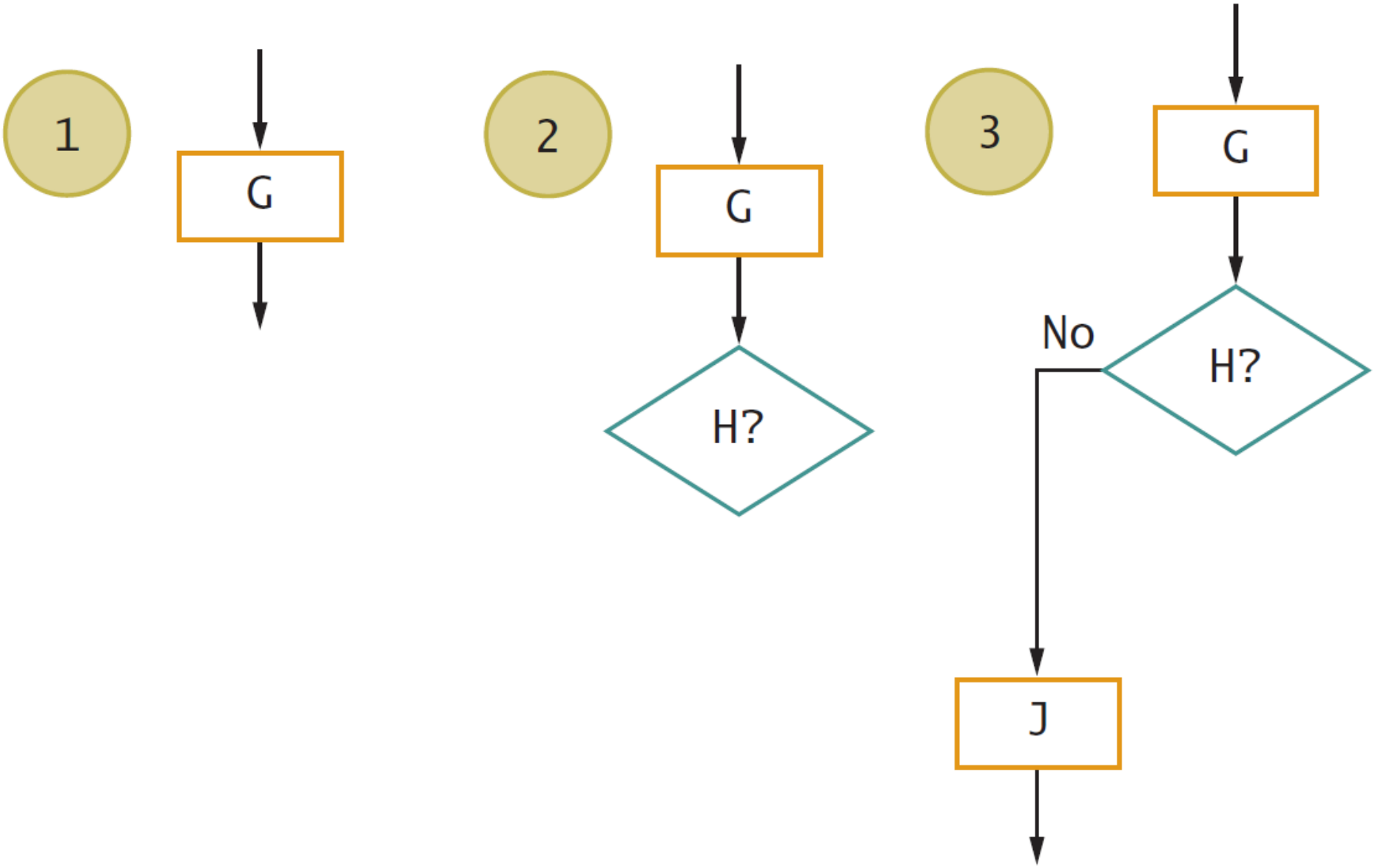




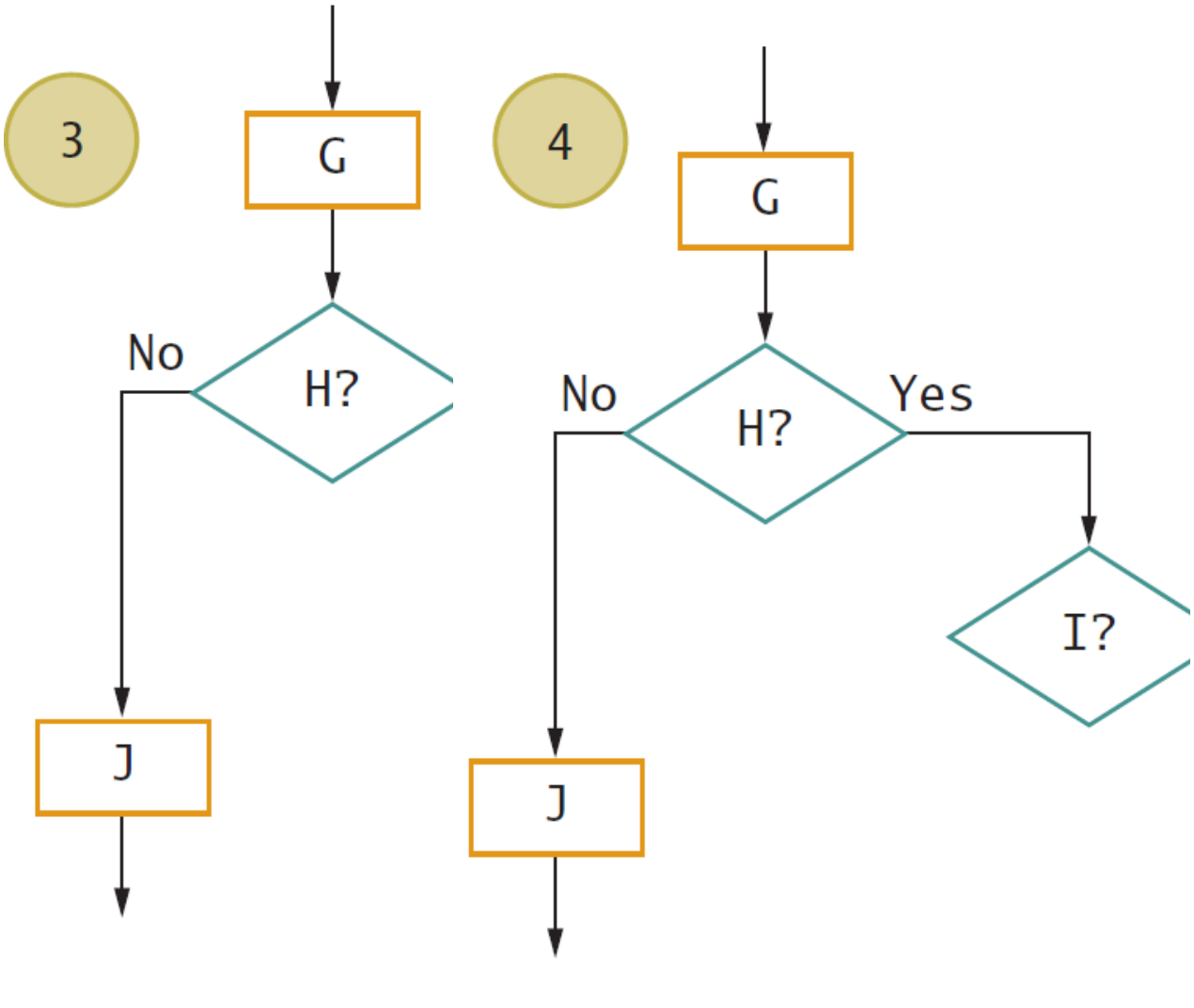
Phương pháp Spaghetti Bowl

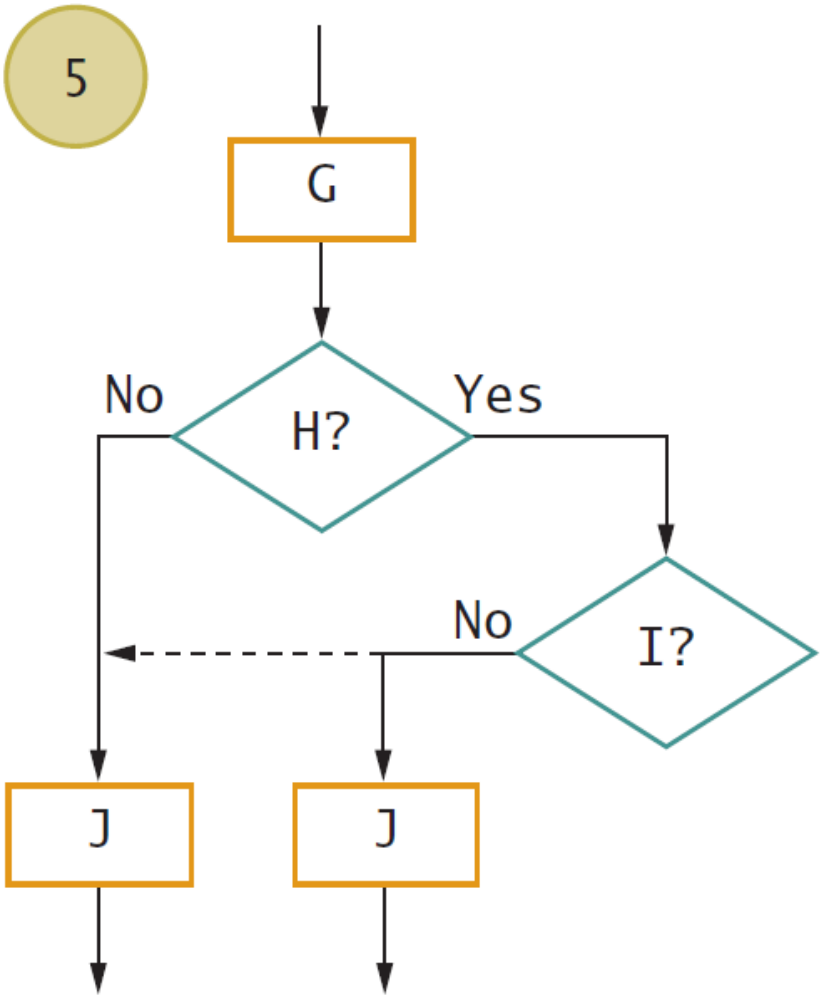


Phương pháp Spaghetti Bowl

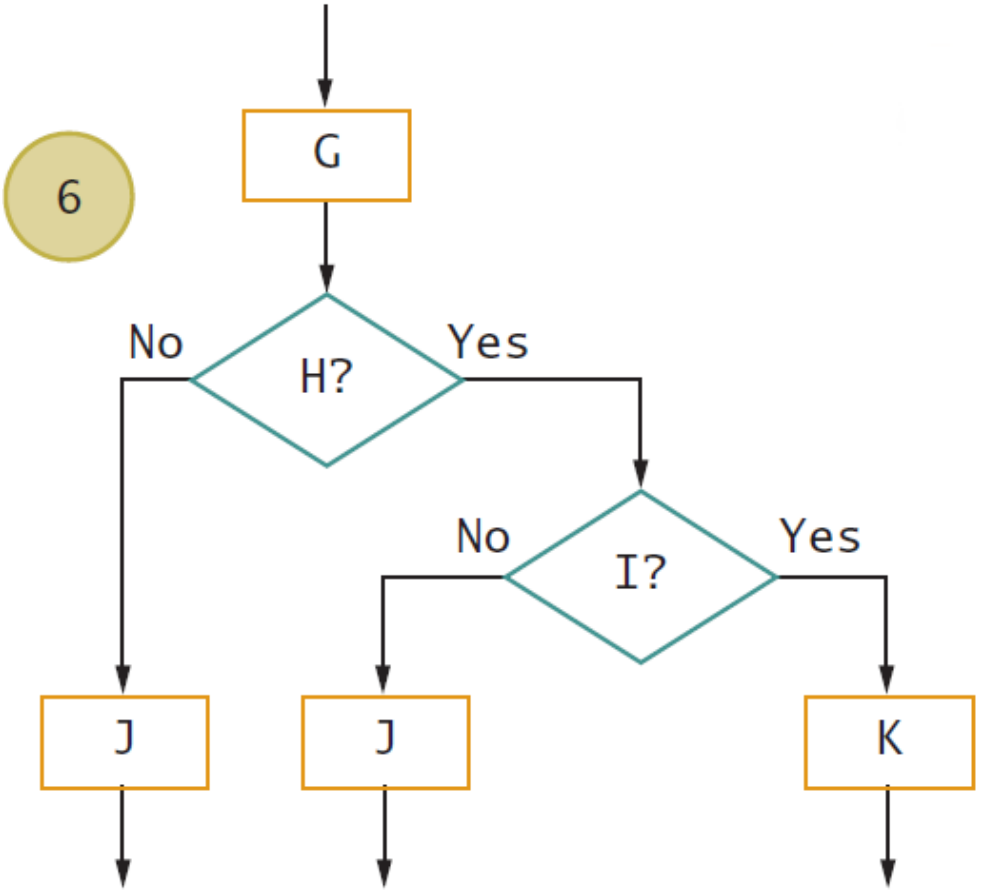


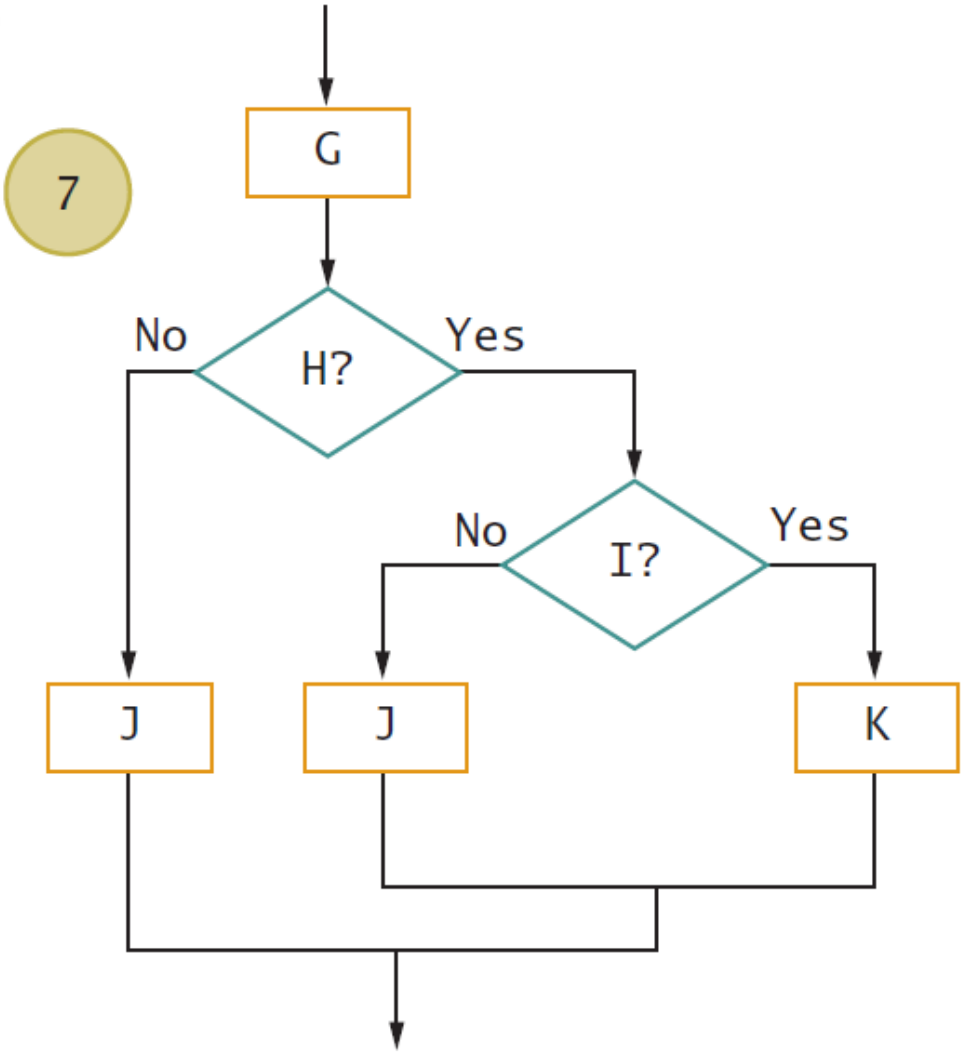
Phương pháp Spaghetti Bowl



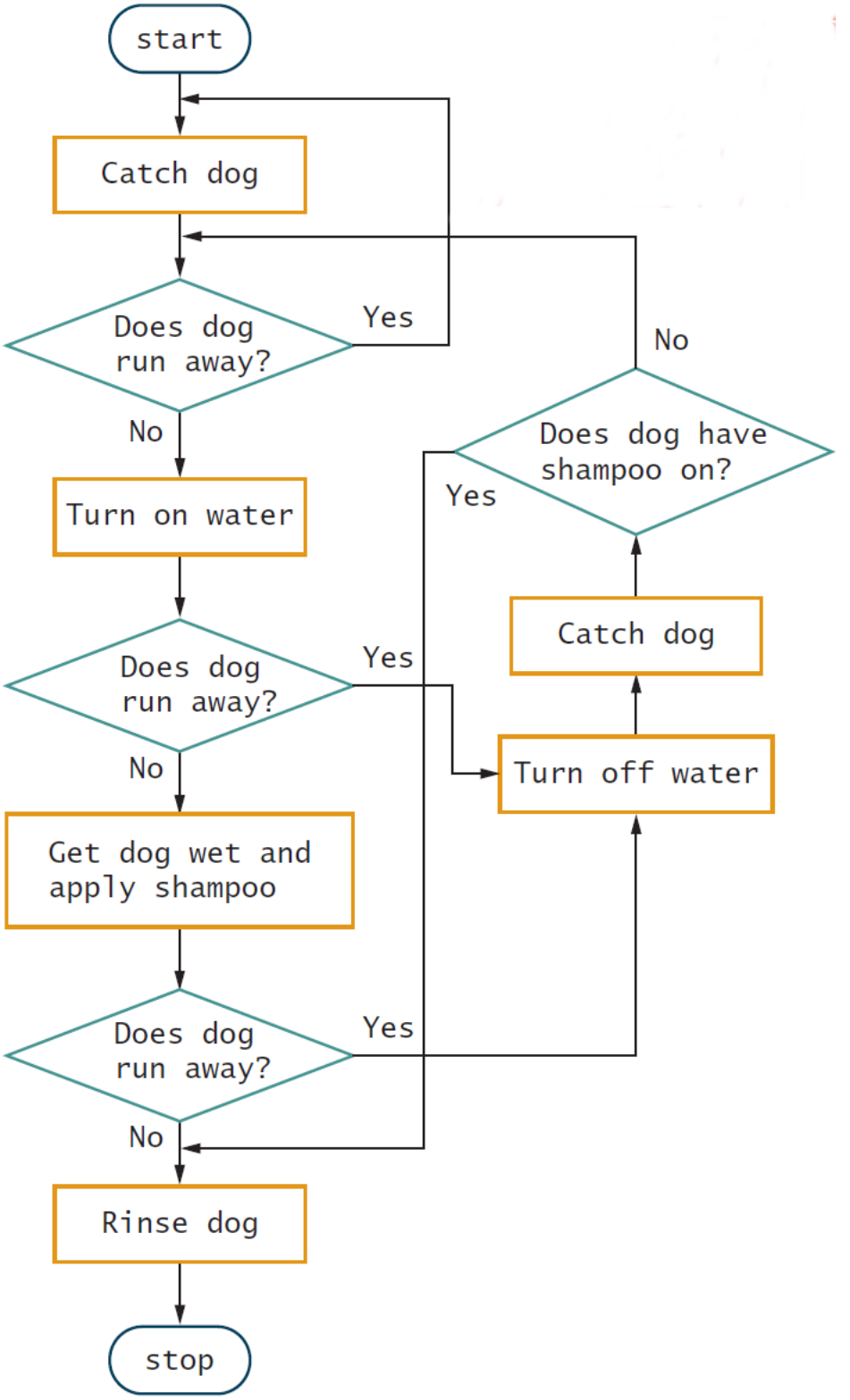


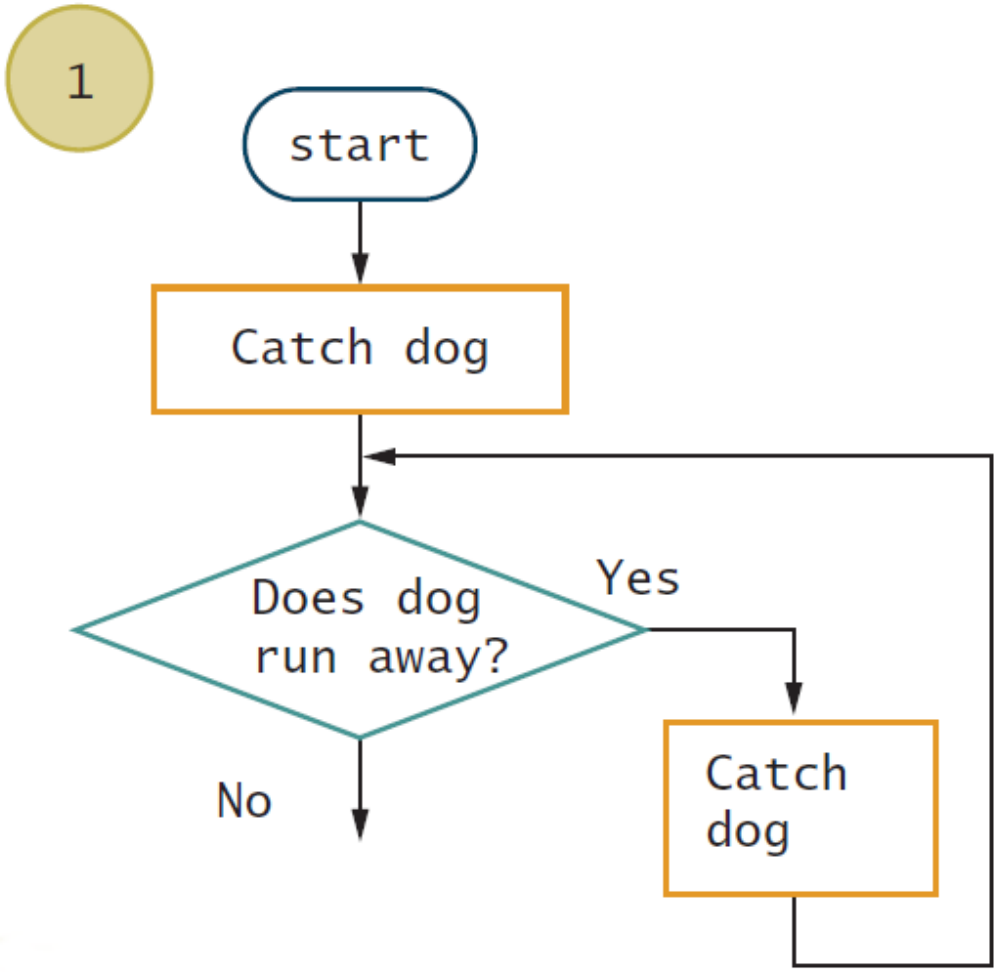
Phương pháp Spaghetti Bowl

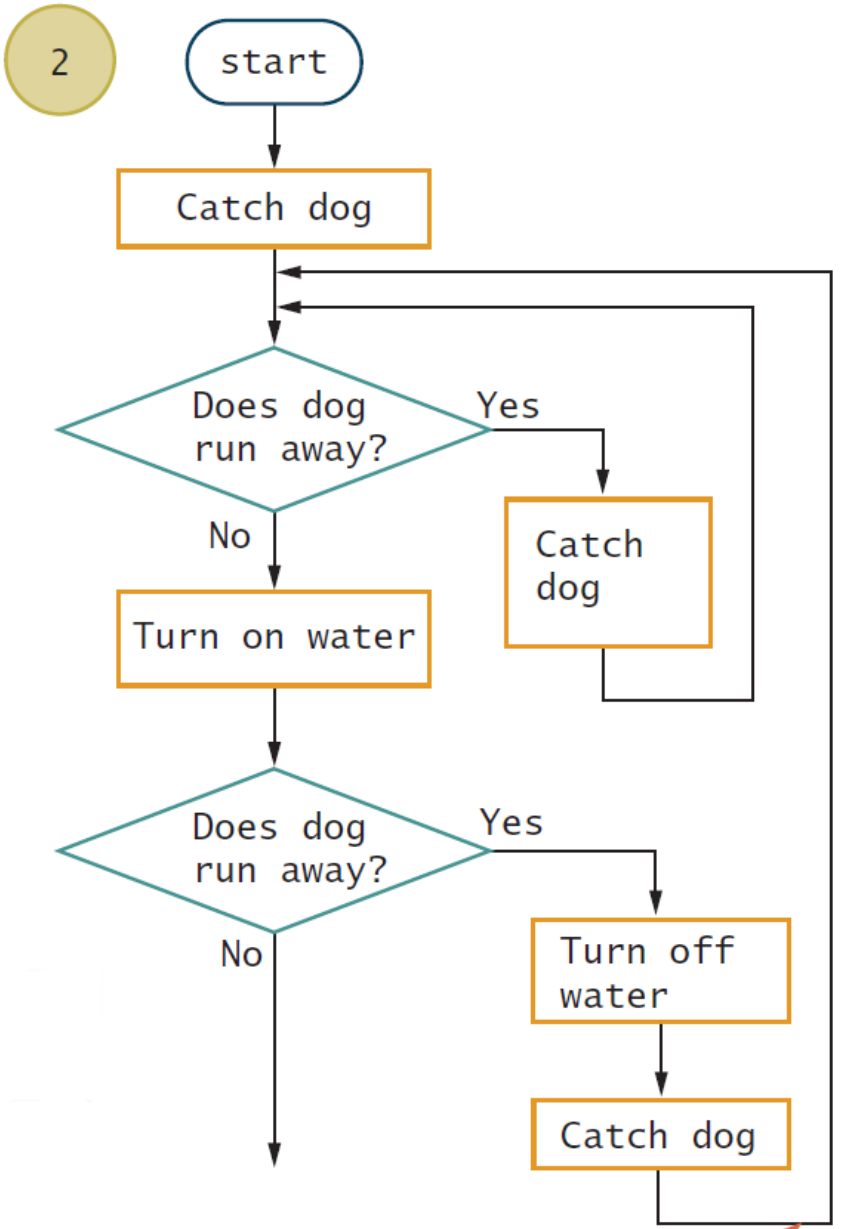




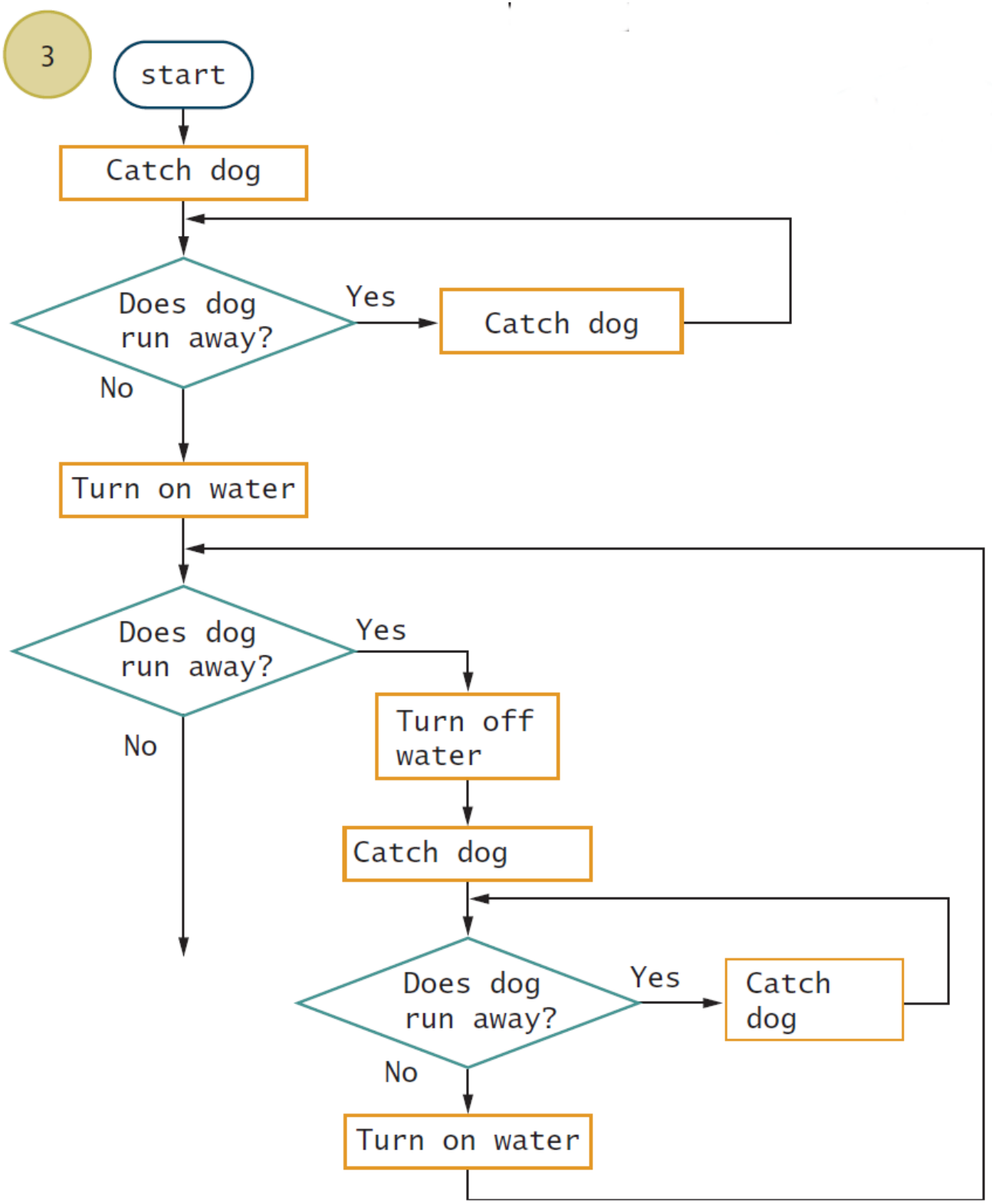
3.6 Cấu trúc và phân đoạn chương trình

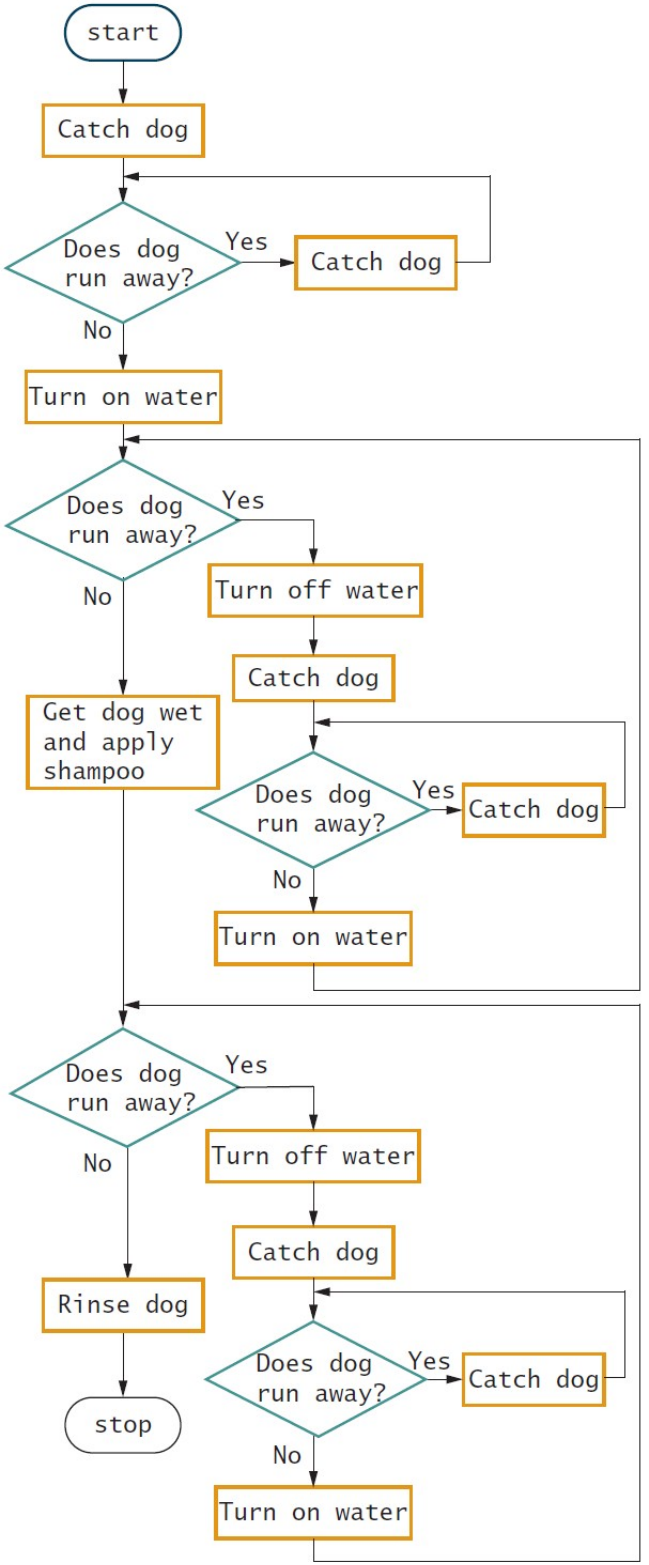




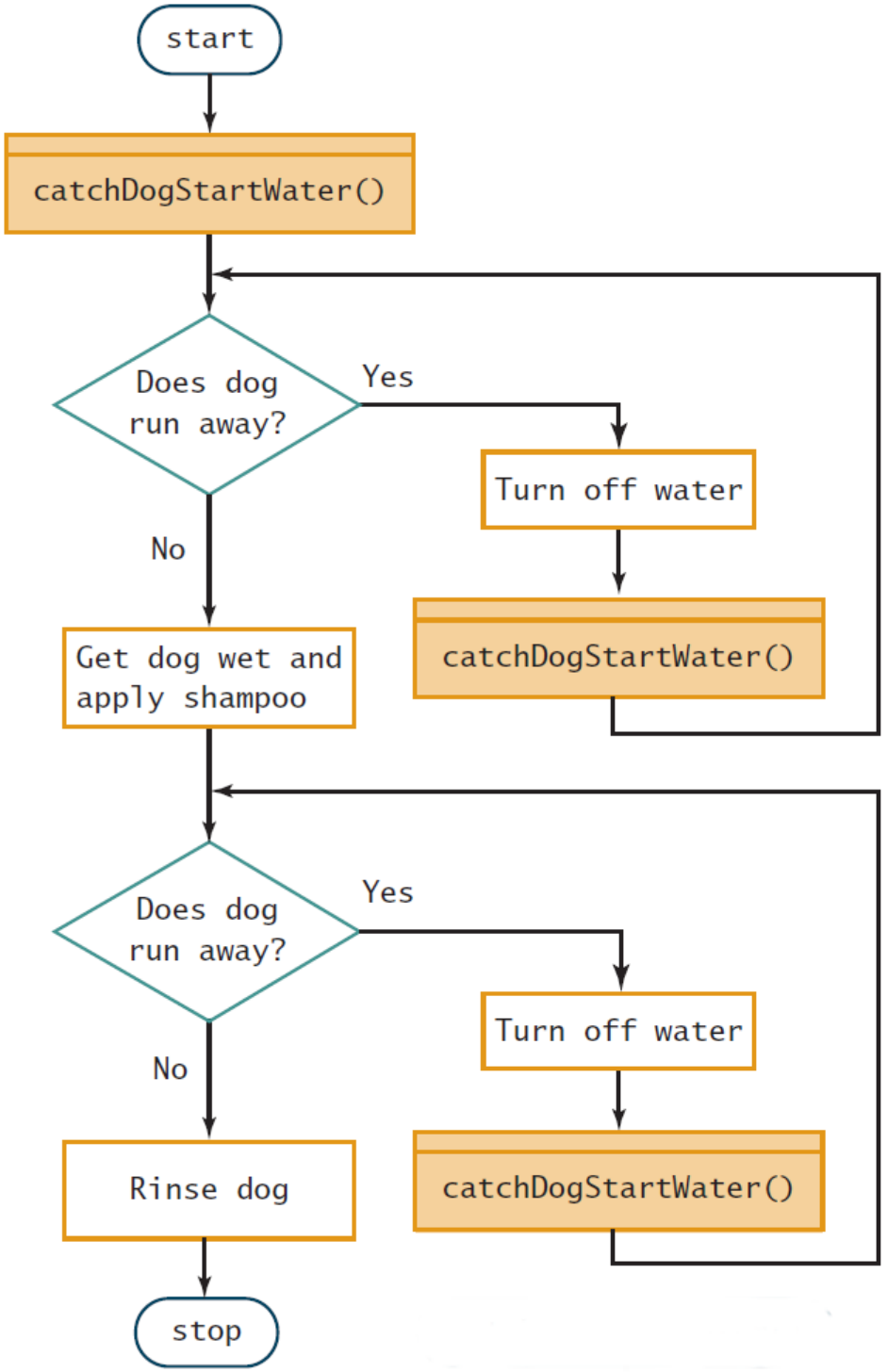


Don't Do It
This loop is not structured because its logic does not return to the question after its body executes.


3
start
Catch dog
Does dog run away?
Yes
Catch dog
No
Turn on water
Does dog run away?
Yes
Turn off water
No
Catch dog
Does dog run away?
Yes
Catch dog
No
Turn on water

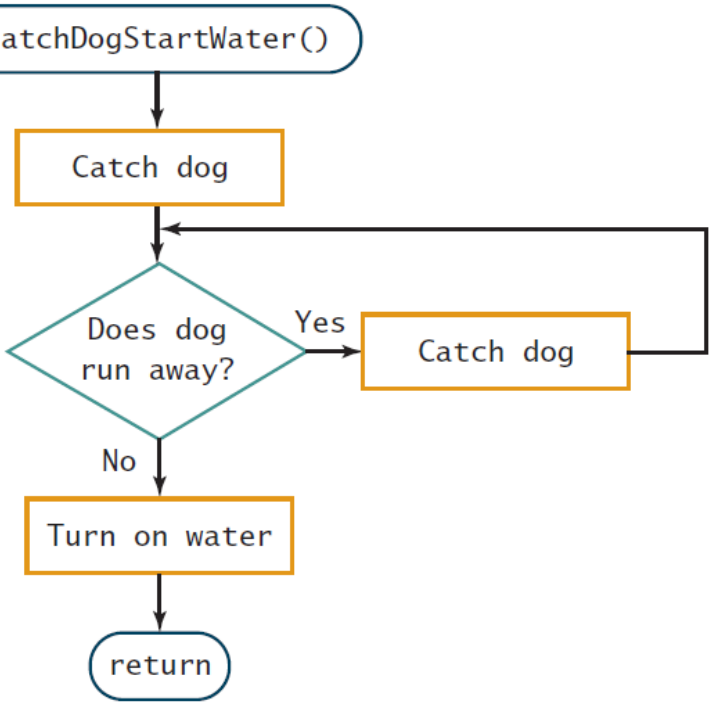



```
start
   Catch dog
   while dog runs away
      Catch dog
   endwhile
   Turn on water
   while dog runs away
      Turn off water
      Catch dog
      while dog runs away
         Catch dog
      endwhile
      Turn on water
   endwhile
   Get dog wet and apply shampoo
   while dog runs away
 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




```
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
```



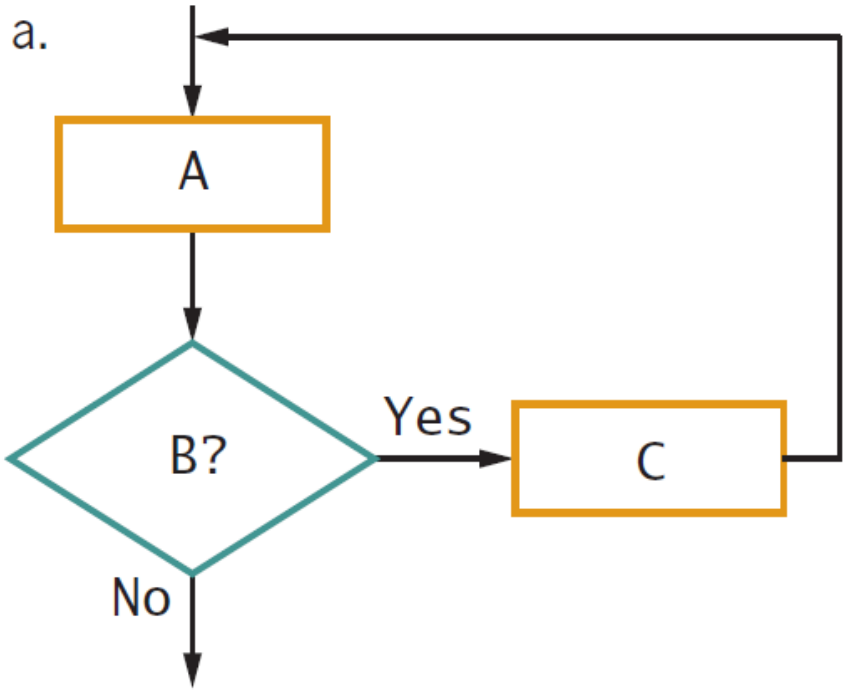
3.7 Tổng kết

- Spaghetti code: đoạn mã không có cấu trúc.
- Chương trình có thể chỉ sử dụng 3 cấu trúc cơ bản: tuần tự, chọn (rẽ nhánh), và lặp.
- Priming input: lệnh nhập liệu để bắt đầu một vòng lặp.
- Lập trình viên sử dụng kĩ thuật cấu trúc để đạt được sự rõ ràng, chuyên nghiệp, thuận tiện, và phân đoạn chương trình.
- Spaghetti bowl: Phương pháp để tạo cấu trúc cho một chương trình.

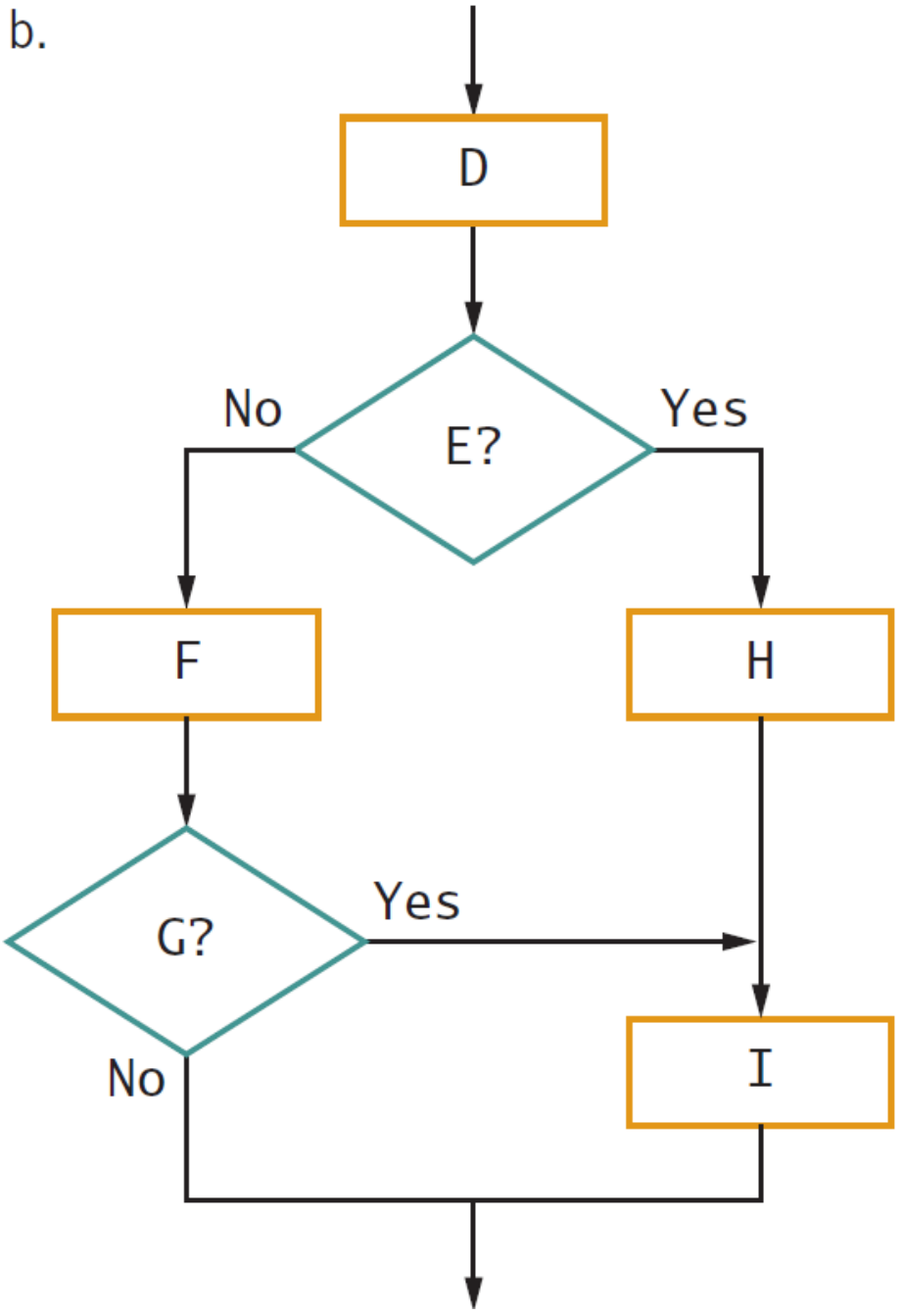
3.8 Bài tập

- Vẽ lại các sơ đồ sau theo cấu trúc.

a.



b.



c.