Bài 3

Logic lập trình
Các cấu trúc điều khiển

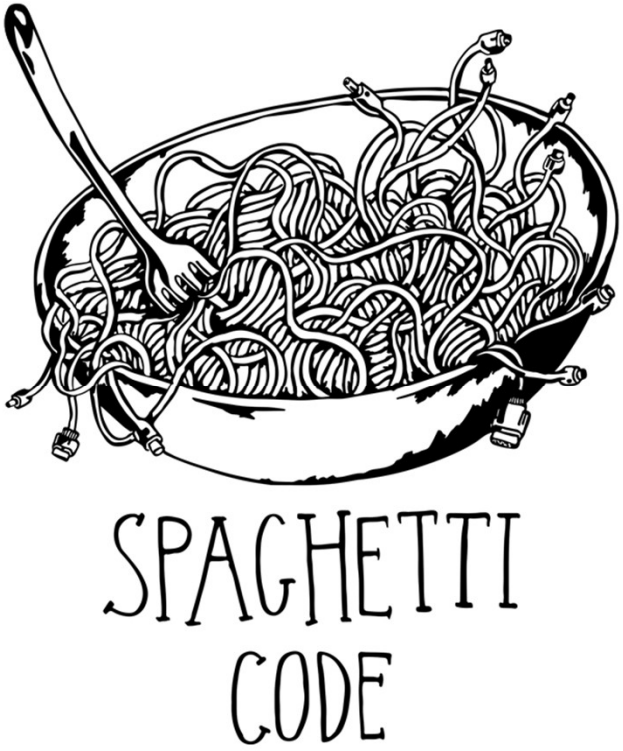
Nội dung

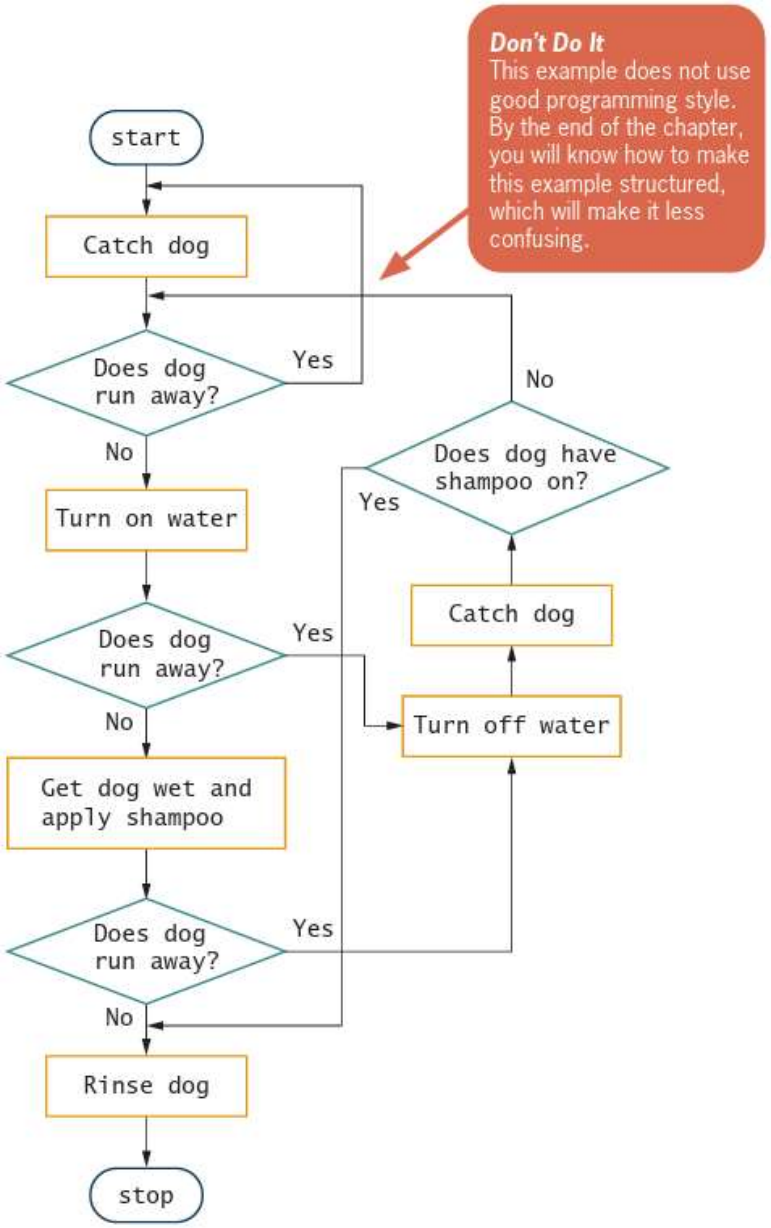
- Nhược điểm của chương trình phi cấu trúc
- Ba cấu trúc điều khiển cơ bản
- Sự cần thiết của chương trình có cấu trúc
- Nhận diện cấu trúc
- Cấu trúc hóa logic chương trình

Nhược điểm của chương trình phi cấu trúc (1)

- Mã spaghetti (spaghetti code)
  - Mô tả các câu lệnh của một chương trình có logic rối rắm.
  - Mã chương trình khó hiểu và dễ bị lỗi.
  - Chương trình vẫn làm việc nhưng khó bảo trì.
- Chương trình phi cấu trúc có logic sử dụng mã spaghetti
  - Không tuân theo các quy tắc điều khiển luồng logic.
- Chương trình có cấu trúc
  - Tuân theo các quy tắc điều khiển luồng logic.
  - Loại bỏ các vấn đề do mã spaghetti gây ra.

Nhược điểm của chương trình phi cấu trúc (2)



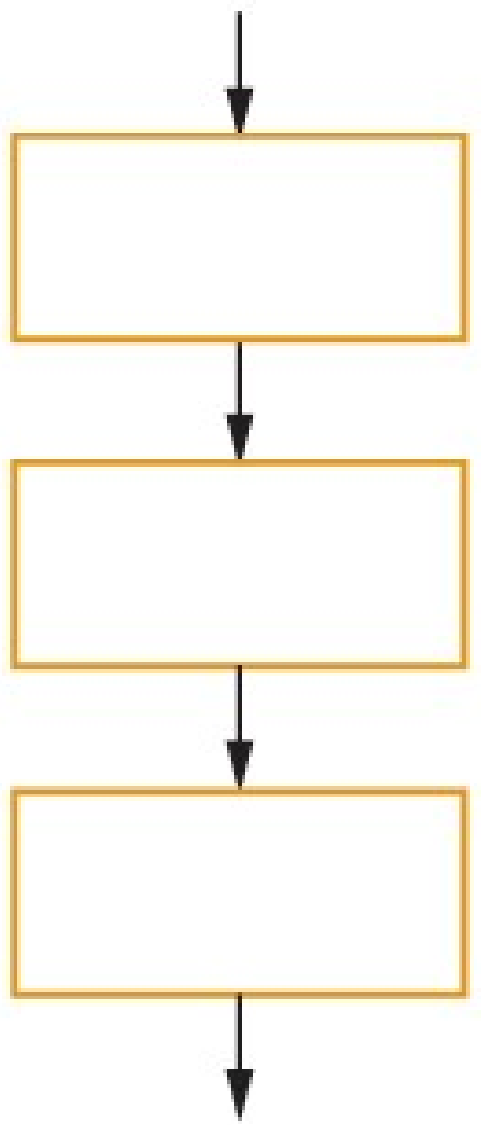


Ba cấu trúc điều khiển cơ bản

- Cấu trúc điều khiển là đơn vị cơ bản của logic lập trình.
- Ba loại cấu trúc điều khiển cơ bản:
  - Cấu trúc tuần tự (sequence structure)
  - Cấu trúc lựa chọn (selection structure)
  - Cấu trúc vòng lặp (loop structure)

Cấu trúc tuần tự

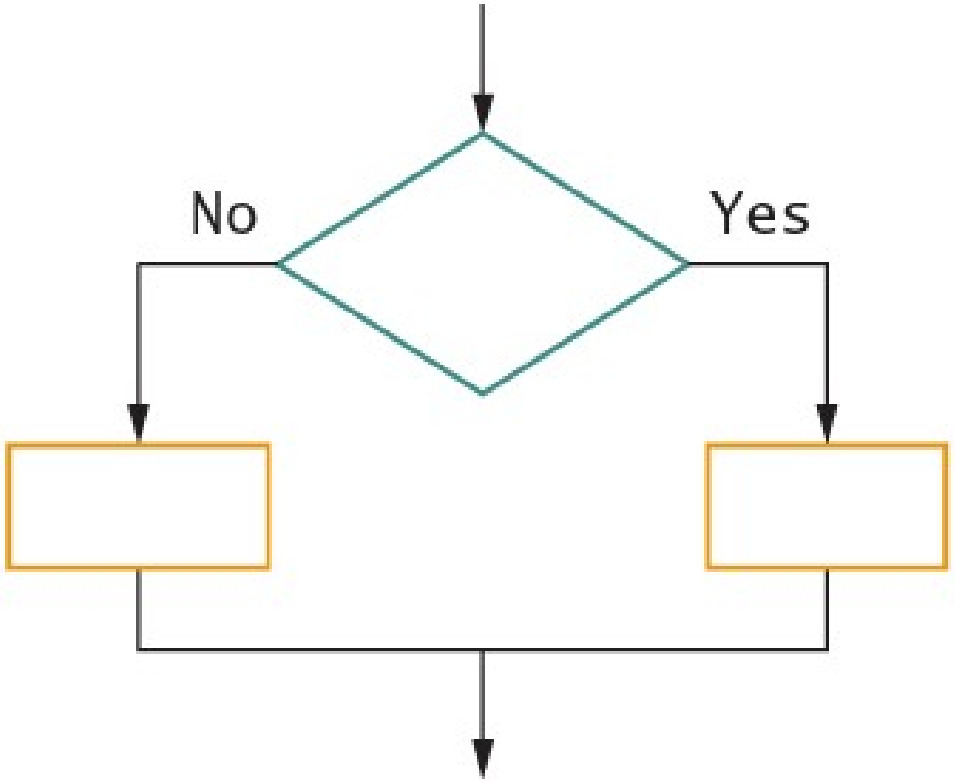
- Thực hiện các hành động hay tác vụ theo thứ tự lần lượt.
- Một cấu trúc có thể bao gồm số lượng nào đó các hành động, nhưng không có tùy chọn để phân nhánh hay bỏ qua hành động nào.



Cấu trúc lựa chọn (1)

- Thực hiện một trong hai hành động dựa trên câu trả lời cho một câu hỏi.
- Cấu trúc lựa chọn kép có hai hành động để lựa chọn thực hiện.
- Biểu diễn
  - Lưu đồ bắt đầu bằng ký hiệu quyết định. Mỗi nhánh của quyết định có một hành động.
  - Mã giả bắt đầu bằng lệnh **if-then-else** và kết thúc bằng lệnh **endif**.

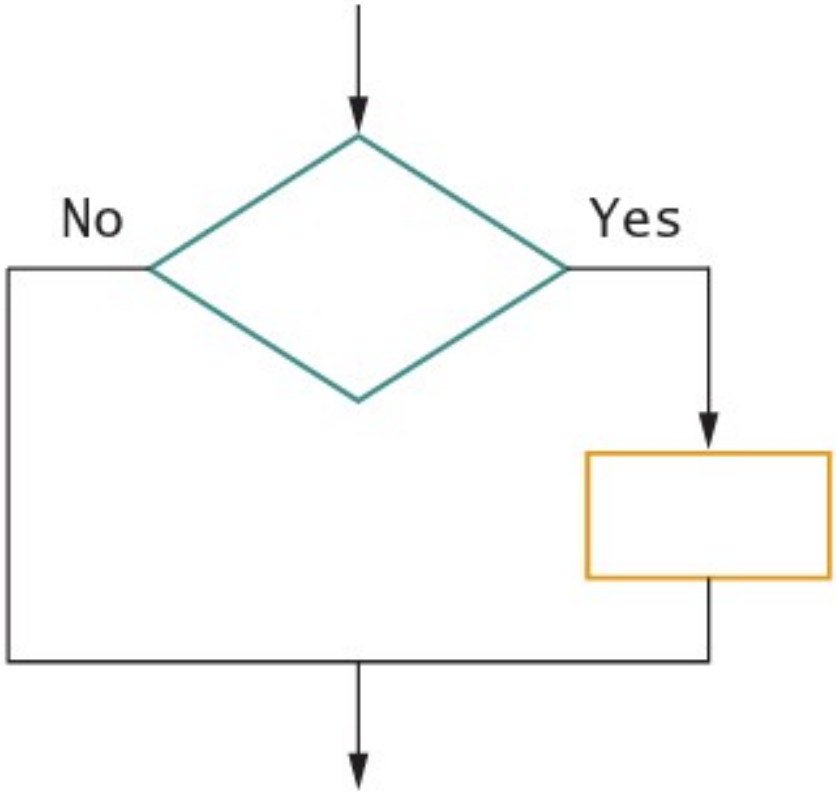
```
if someCondition is true then
   do oneProcess
else
   do theOtherProcess
endif
```



Cấu trúc lựa chọn (2)

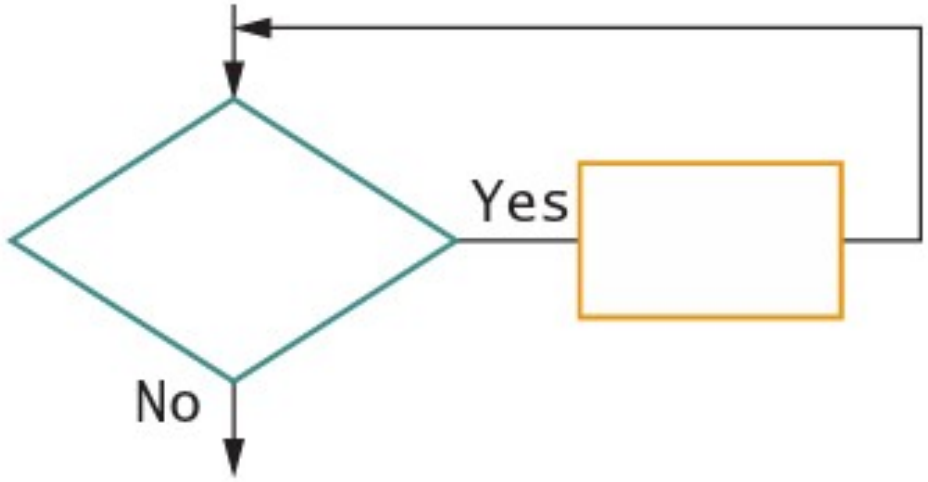
- Cấu trúc lựa chọn đơn có một hành động để lựa chọn thực hiện.
- Biểu diễn
  - Lưu đồ chỉ có một nhánh của quyết định chứa hành động, nhánh còn lại rỗng.
  - Mã giả bắt đầu bằng lệnh **if-then** (không chứa mệnh đề **else**) và kết thúc bằng lệnh **endif**.

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



Cấu trúc vòng lặp (1)

- Thực hiện lặp lại một hay nhiều hành động trong khi một điều kiện vẫn còn đúng.
  - Các hành động xảy ra trong vòng lặp là thân vòng lặp.
- Kiểu vòng lặp phổ biết nhất được gọi là vòng lặp while hoặc while .. do.
- Biểu diễn
  - Lưu đồ bắt đầu bằng ký hiệu quyết định có nhánh quay về vị trí trước quyết định.
  - Mã giả bắt đầu bằng lệnh **while** hoặc **while do** và kết thúc bằng lệnh **endwhile**.



Cấu trúc vòng lặp (2)

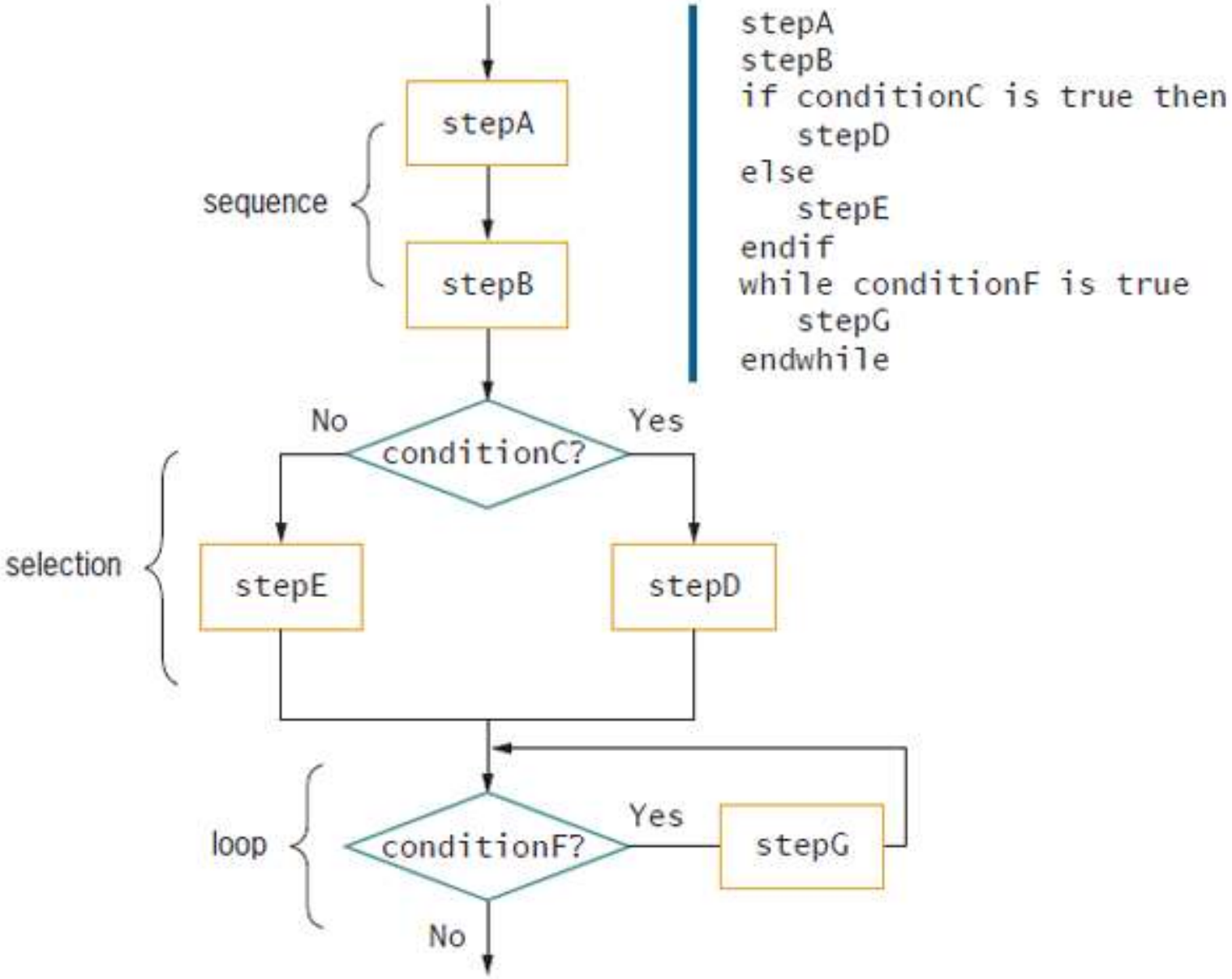
```
while testCondition continues to be true do
    someProcess
endwhile
```

```
while you continue to be hungry
    take another bite of food
    determine whether you still feel hungry
endwhile
```

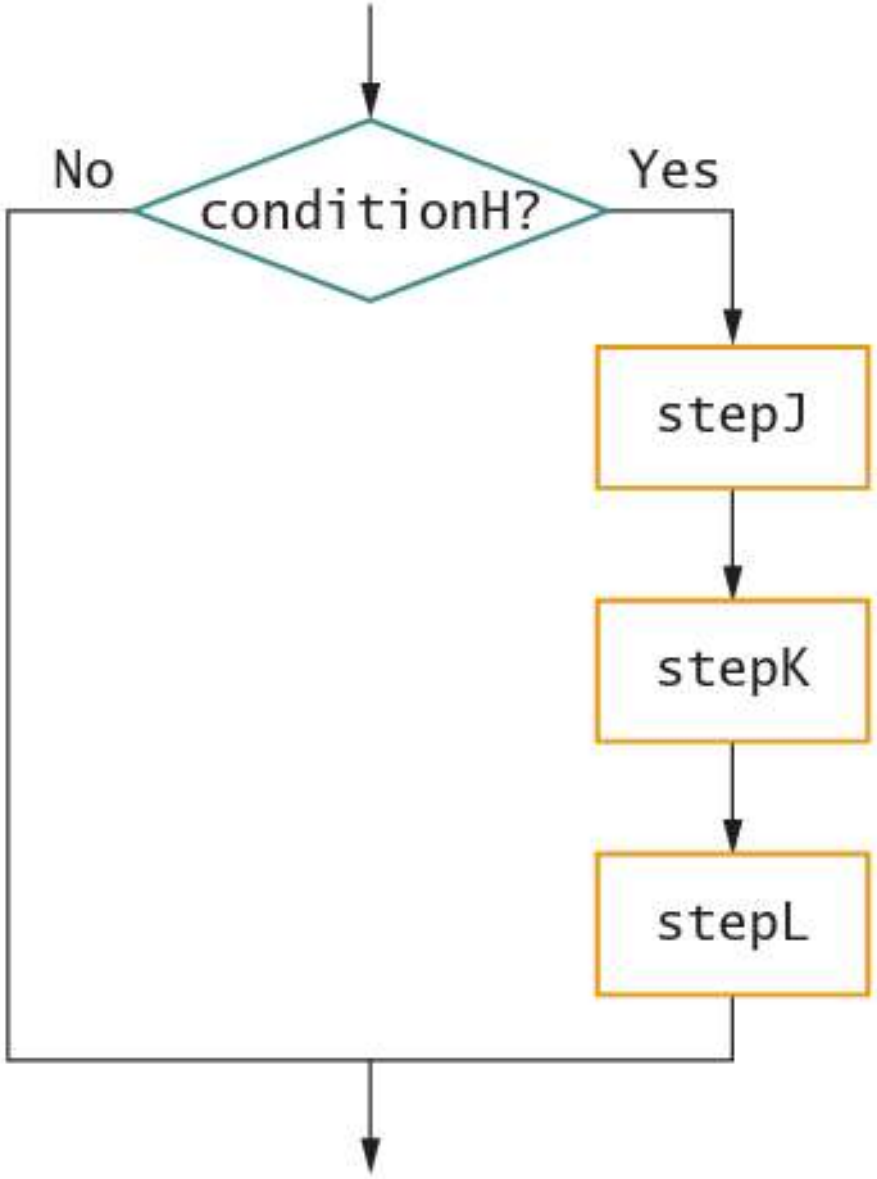
Kết hợp các cấu trúc

- Các cấu trúc có thể được kết hợp theo vô số cách.
- Kết hợp kiểu xếp chồng
  - Các cấu trúc được gắn với nhau theo quy tắc đầu cuối.
- Kết hợp kiểu lồng nhau
  - Một cấu trúc được đặt bên trong một cấu trúc khác khi ta thay thế một hành động trong một cấu trúc bởi một cấu trúc khác.
- Không giới hạn số lượng và cấp độ khi xếp chồng và lồng các cấu trúc.

Kết hợp kiểu xếp chồng

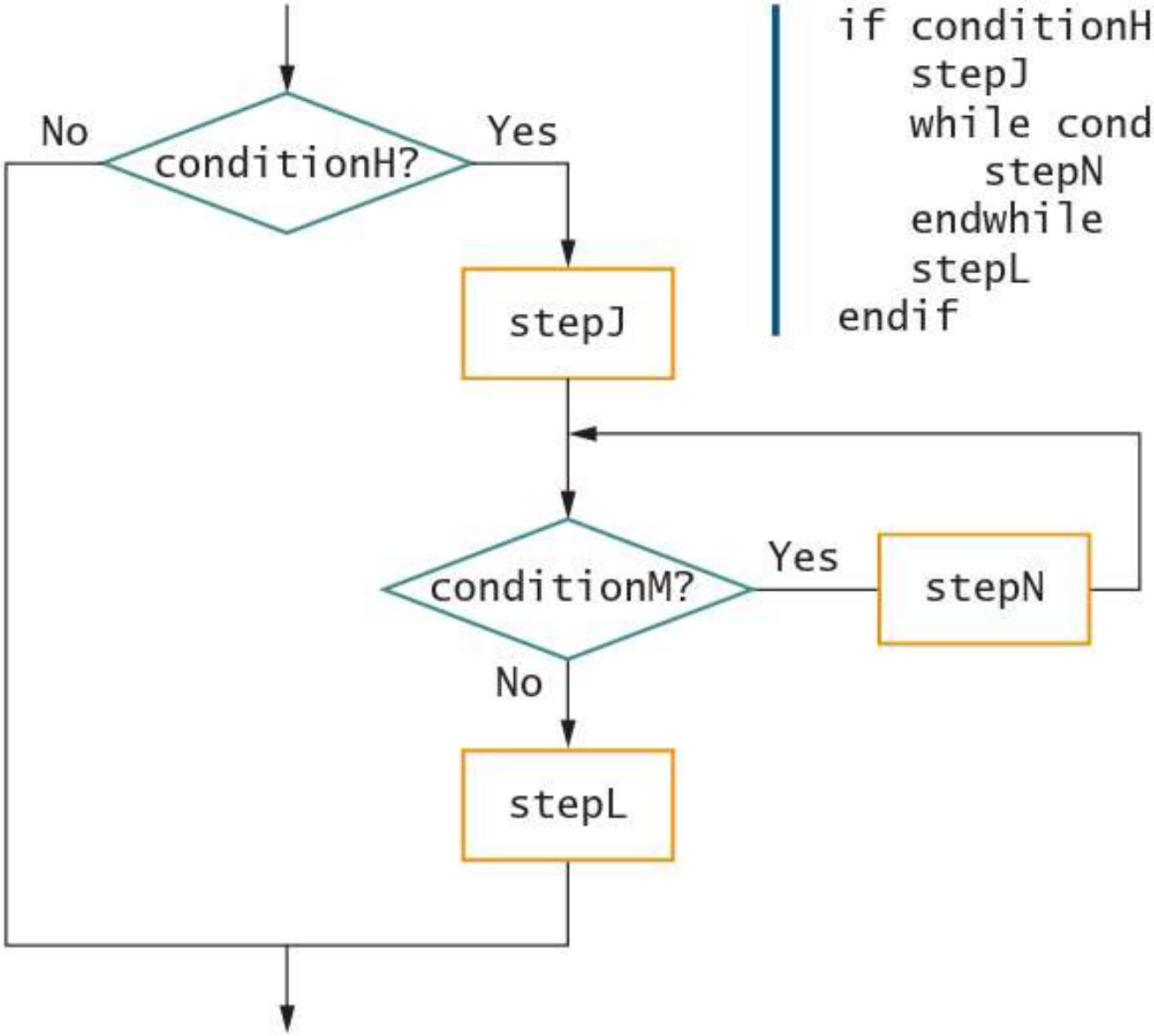


Kết hợp kiểu lồng nhau (1)




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

Kết hợp kiểu lồng nhau (2)




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

Đặc điểm của chương trình có cấu trúc

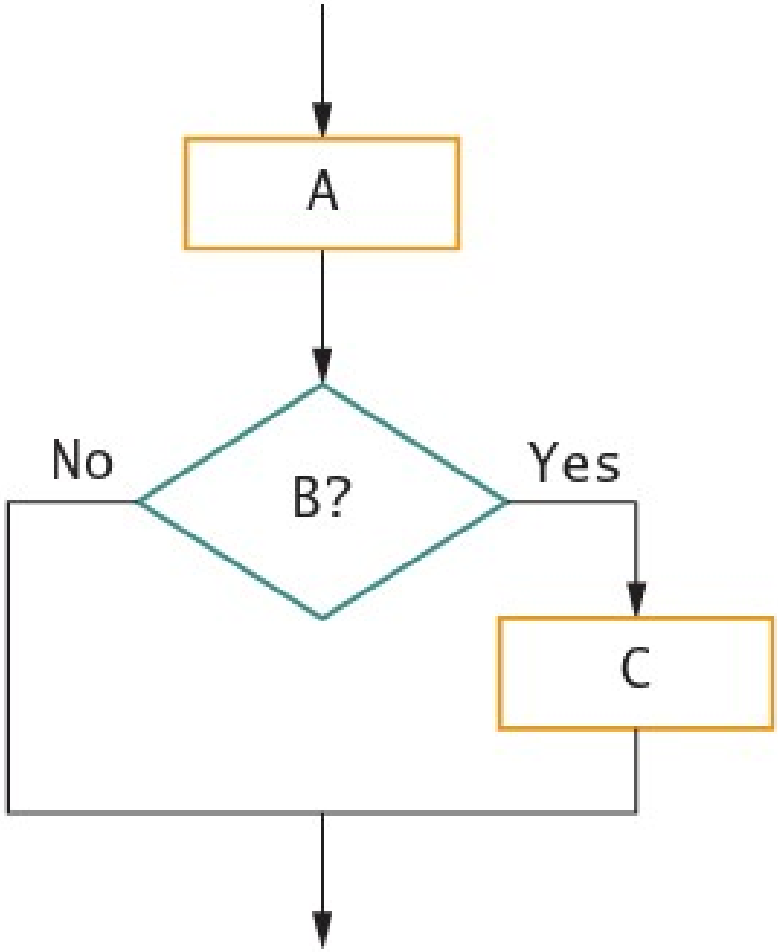
- Chỉ bao gồm sự kết hợp của ba cấu trúc: tuần tự, lựa chọn và vòng lặp.
- Mỗi cấu trúc có duy nhất một điểm bắt đầu và một điểm kết thúc.
- Các cấu trúc chỉ có thể kết hợp theo kiểu xếp chồng hoặc lồng nhau.

Sự cần thiết của chương trình có cấu trúc

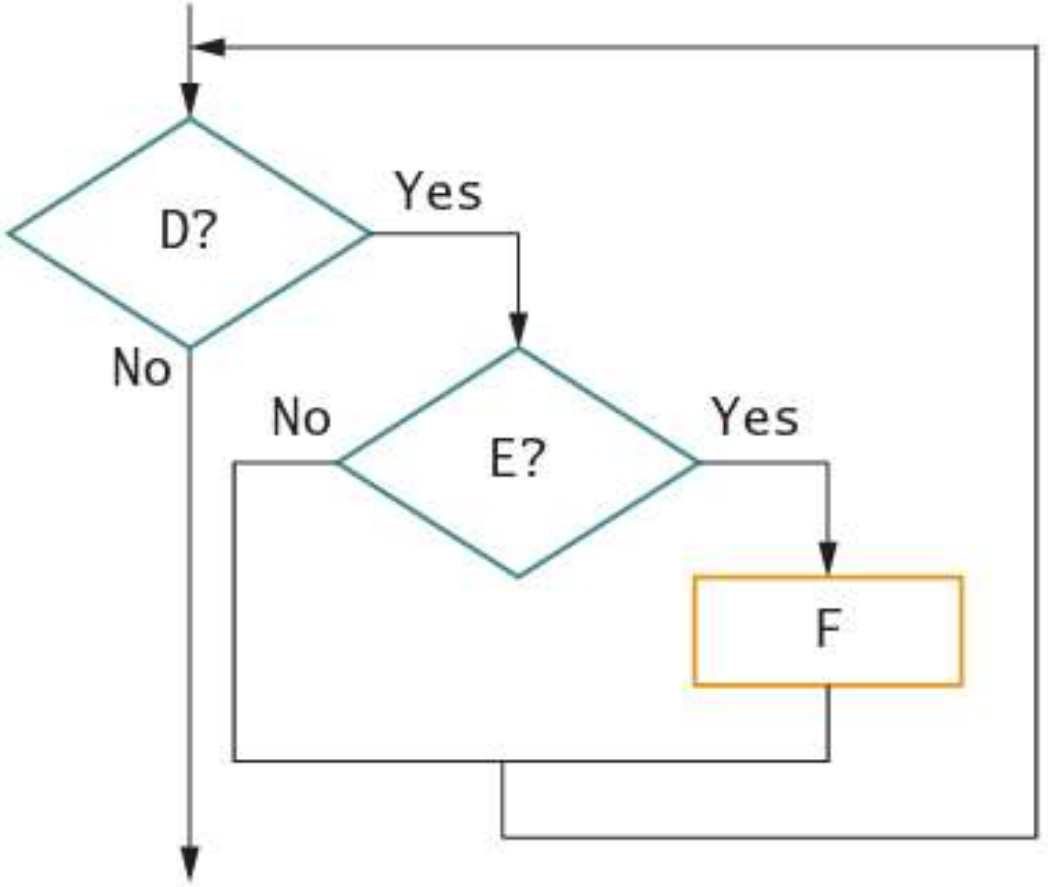
- Sự rõ ràng – chương trình càng lớn sẽ trở nên khó hiểu hơn nếu nó không có cấu trúc.
- Sự chuyên nghiệp – thể hiện chuyên môn nghiệp vụ cao của người lập trình.
- Sự hiệu quả - các ngôn ngữ lập trình mới đều hổ trợ lệnh cho các cấu trúc.
- Dễ bảo trì – các lập trình viên sẽ thấy dễ đọc và dễ chỉnh sửa, bổ sung mã nguồn.
- Tính mô-đun – dễ dàng chia thành các mô-đun và giao cho các lập trình viên.

Nhận diện cấu trúc (1)

- Các đoạn lưu đồ sau có cấu trúc



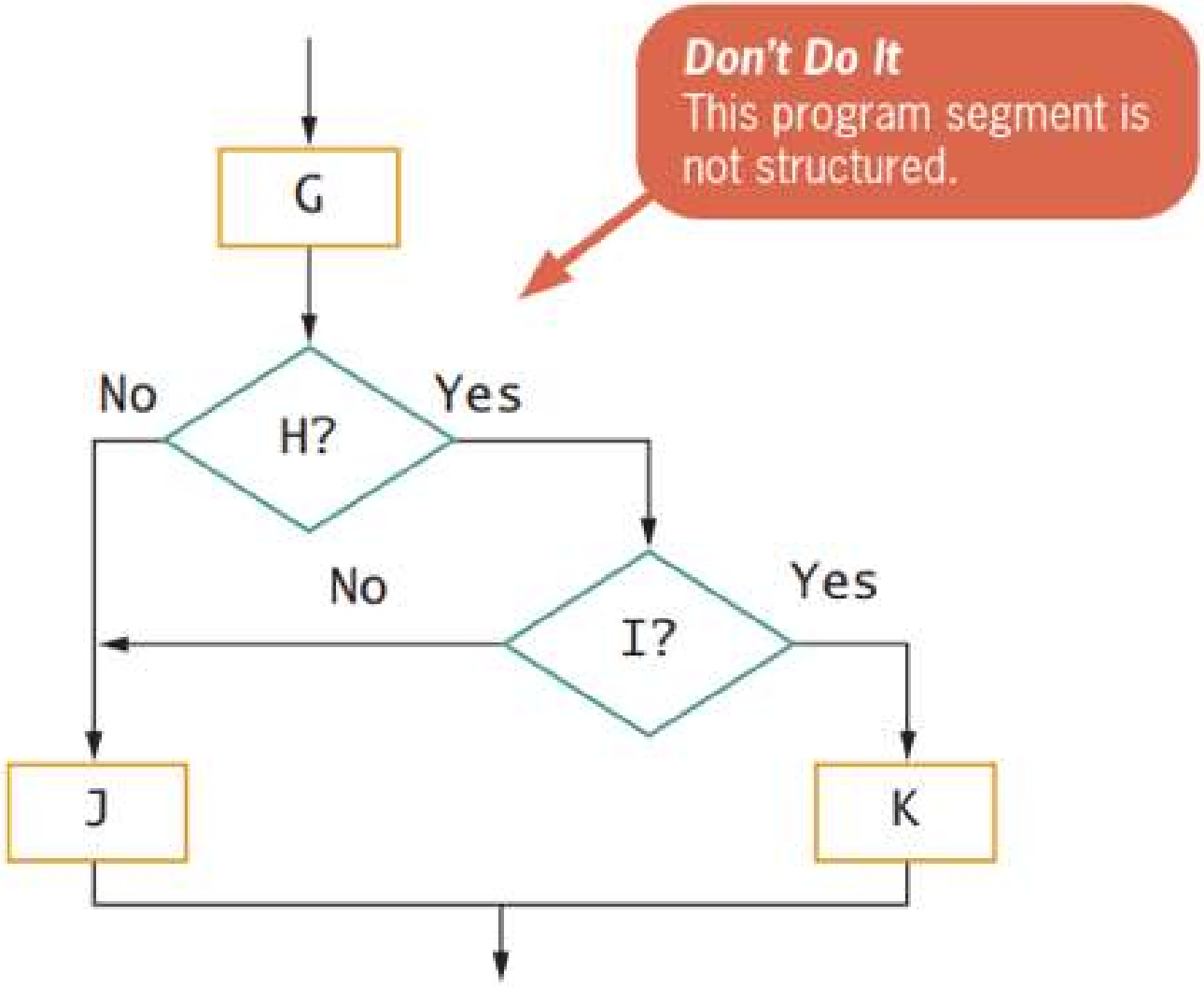
Cấu trúc tuần tự và cấu trúc chọn.



Cấu trúc vòng lặp và cấu chọn bên trong vòng lặp.

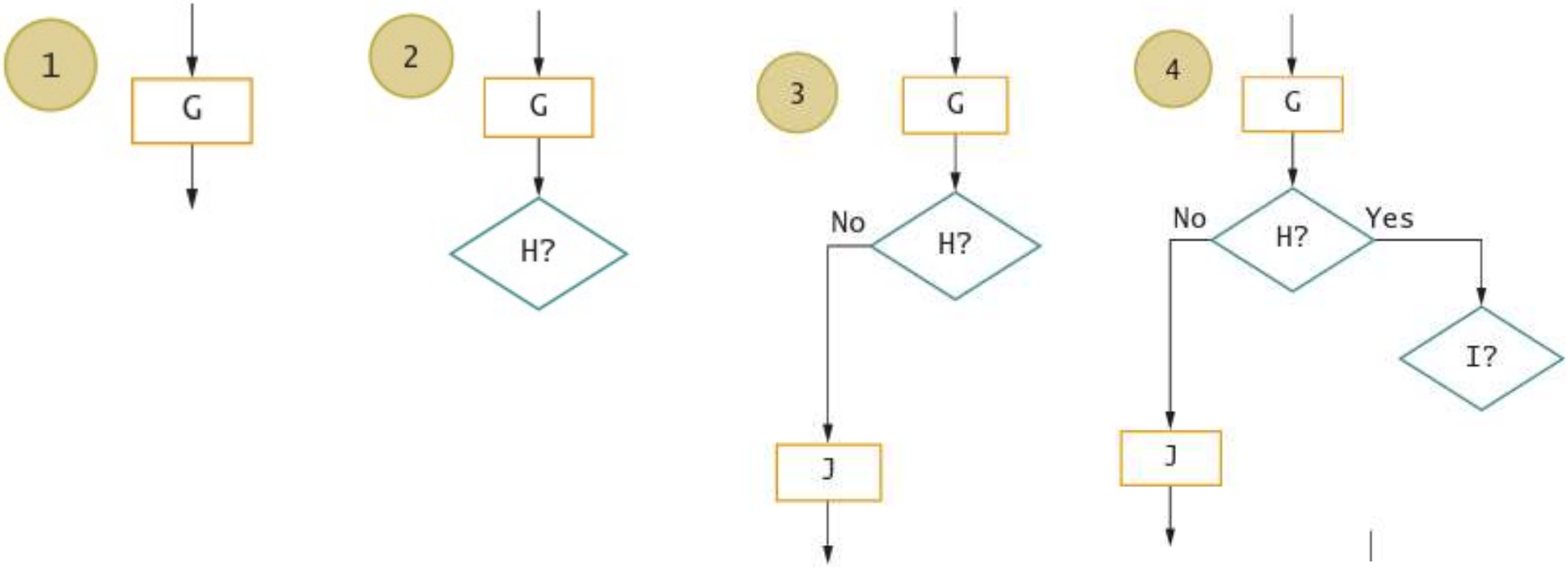
Nhận diện cấu trúc (2)

- Đoạn lưu đồ sau phi cấu trúc

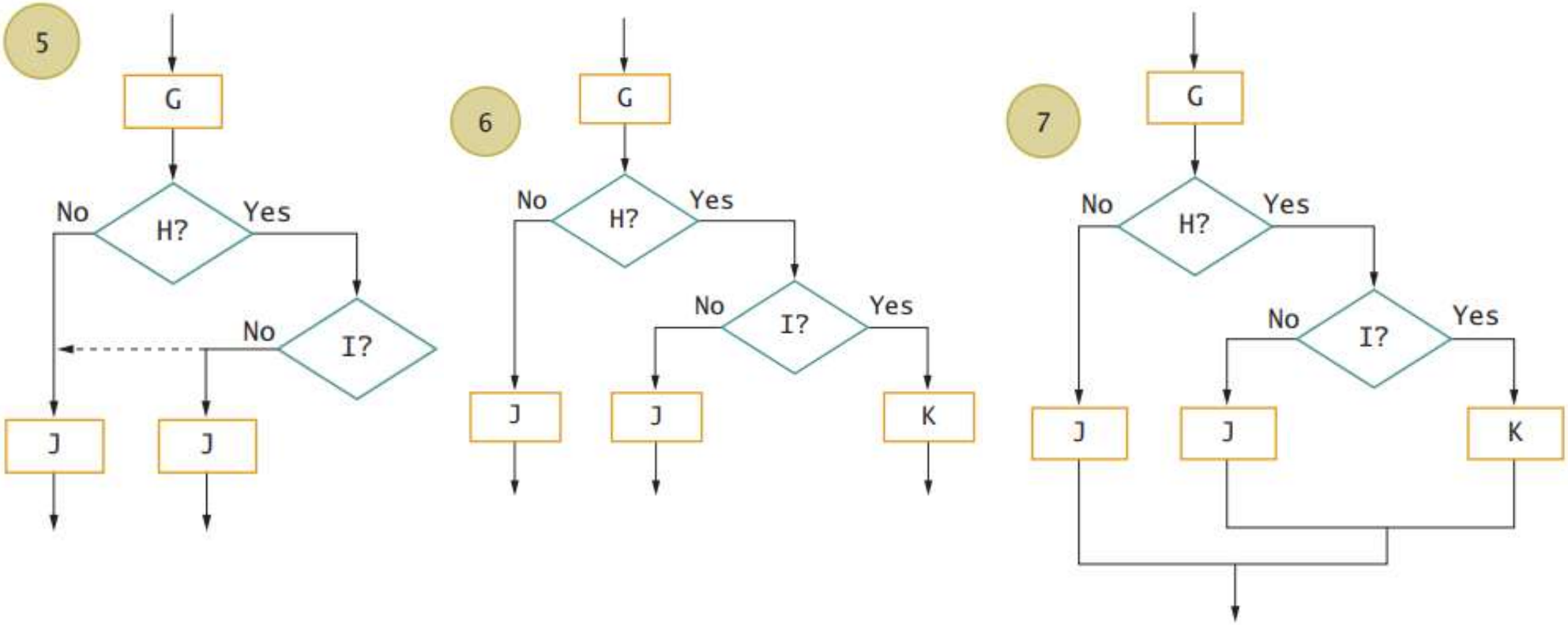


Cấu trúc hóa logic chương trình (1)

- Gỡ rối luồng logic

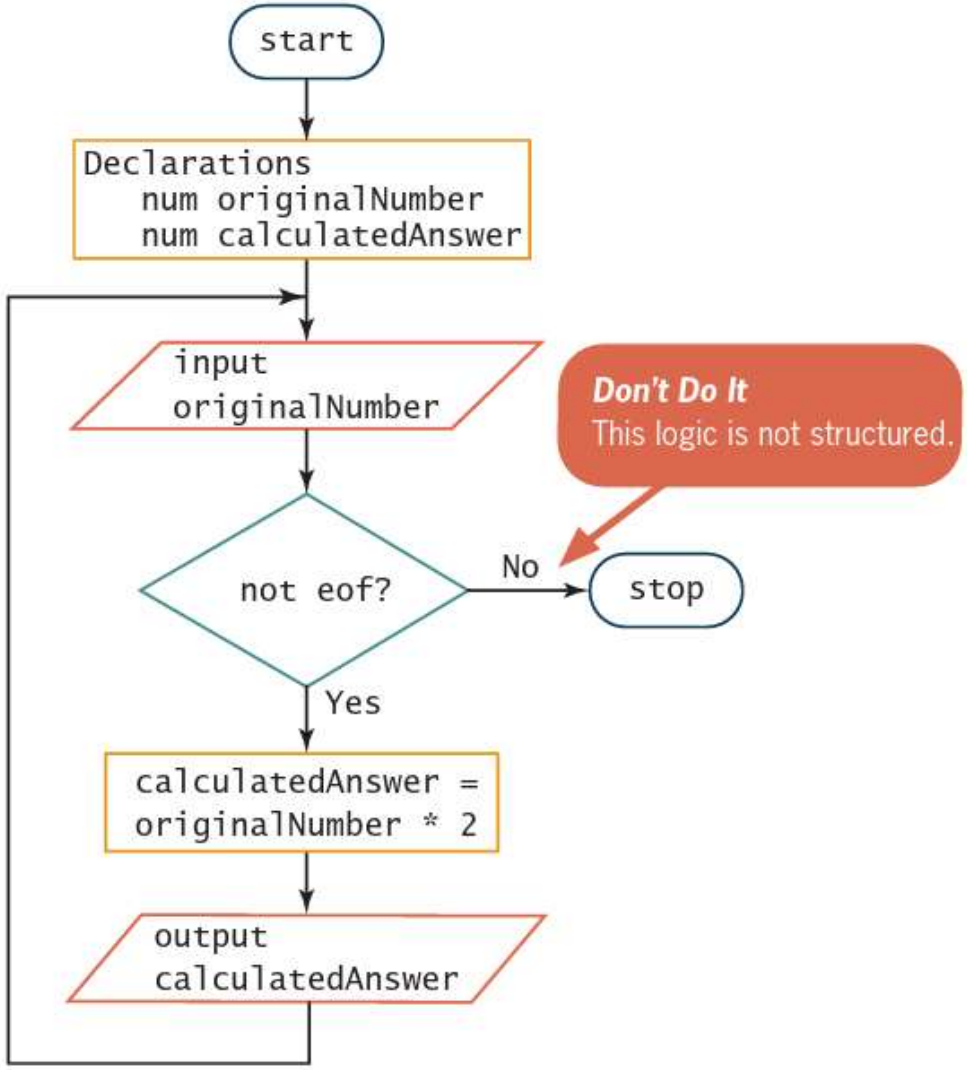


Cấu trúc hóa logic chương trình (2)

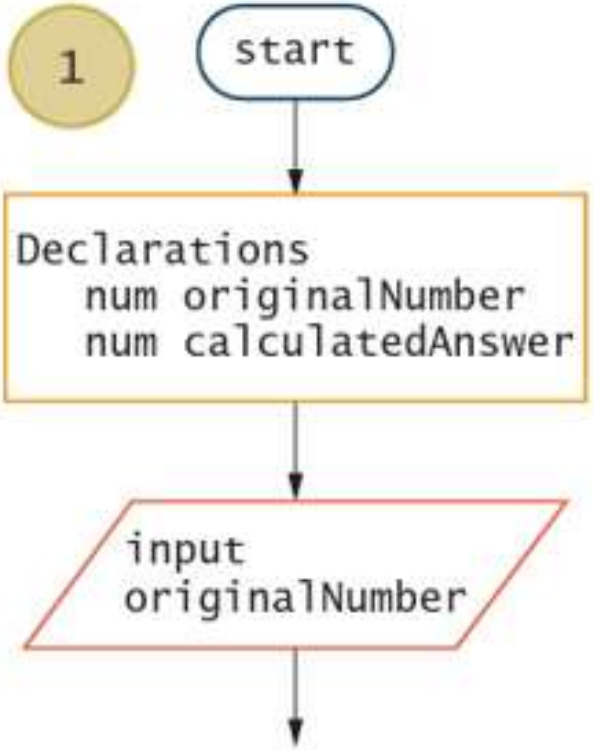


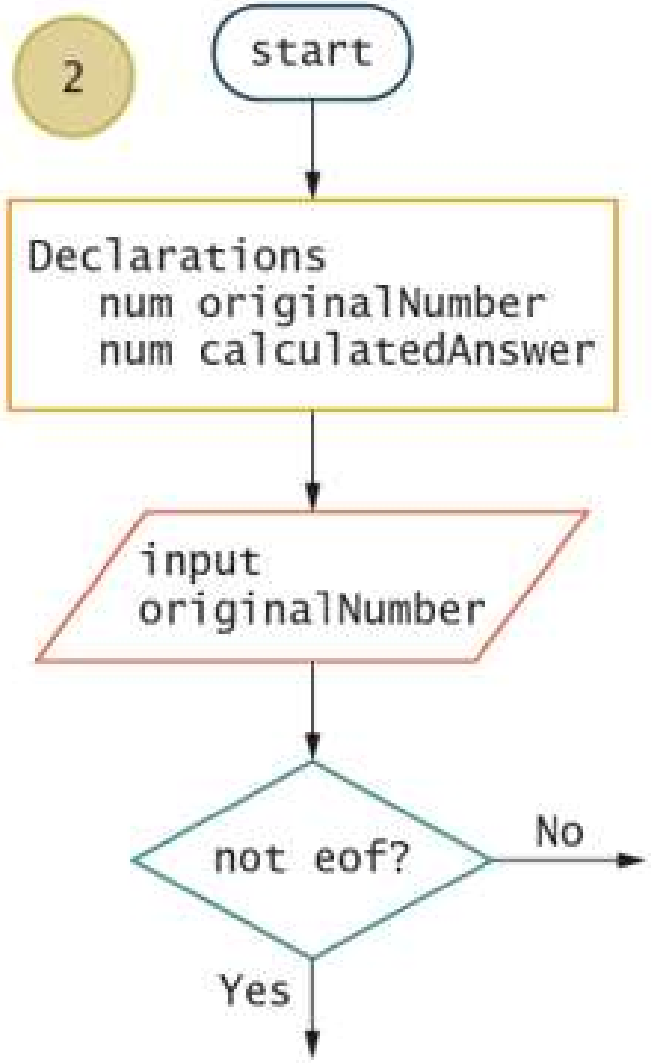
Cấu trúc hóa logic chương trình (3)

- Sử dụng đầu vào mồi (priming input)

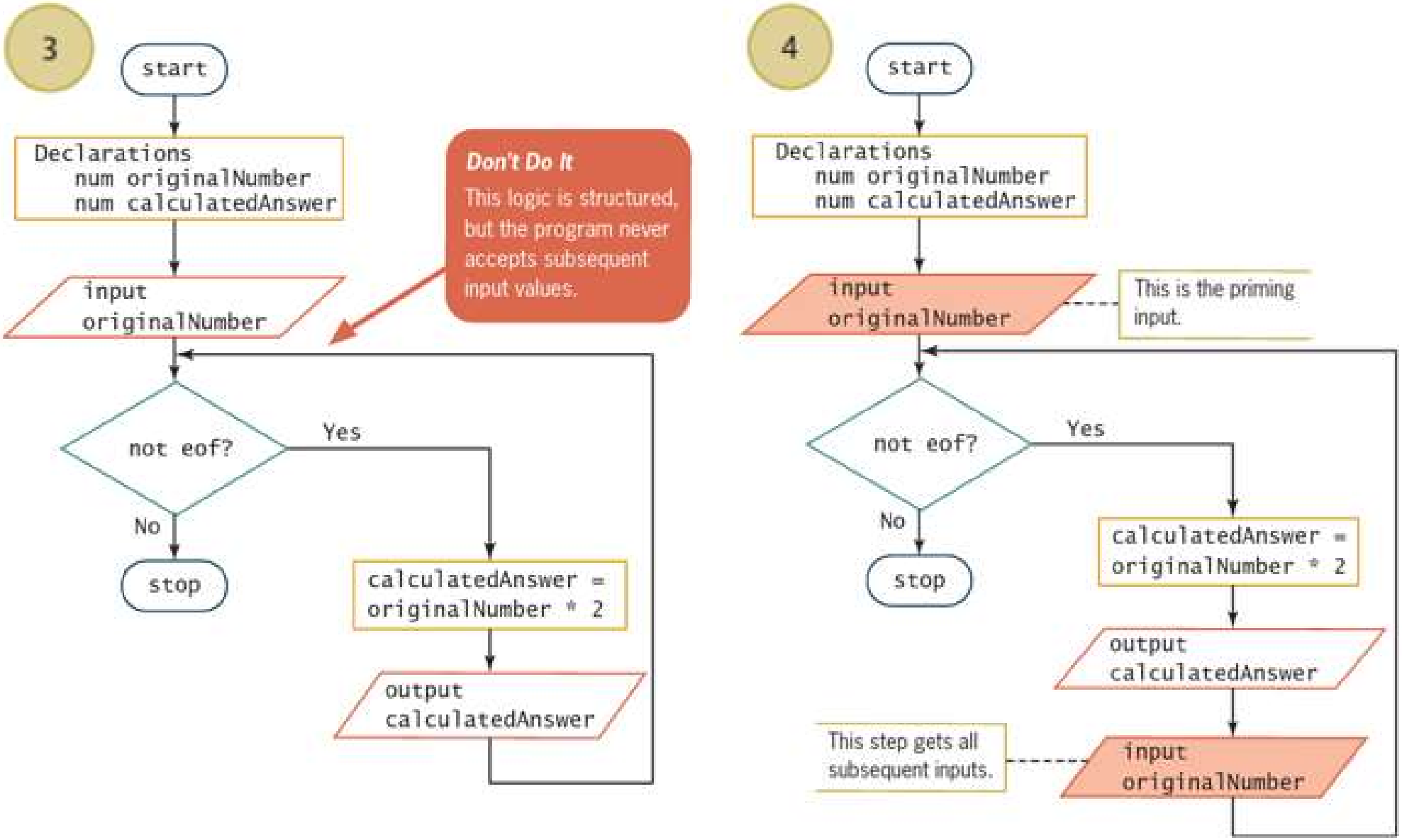


Cấu trúc hóa logic chương trình (2)



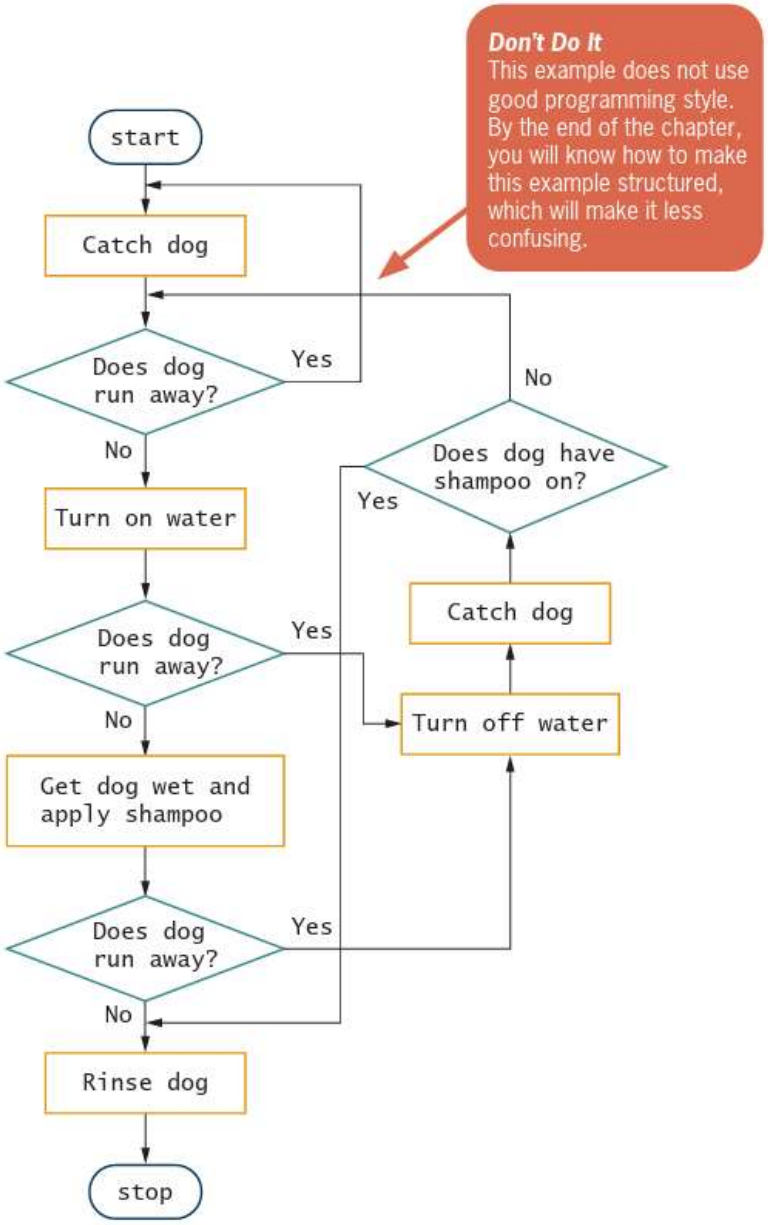


Cấu trúc hóa logic chương trình (3)



Cấu trúc hóa logic chương trình (4)

- Yêu cầu về nhà:
  - Hãy cấu trúc hóa logic chương trình bên.
  - Tham khảo trang 110 – 113 sách Programming Logic and Design, 8th Edition.



Bài tập lập trình (1)

- Dãy Fibonaci là dãy vô hạn các số tự nhiên bắt đầu bằng hai phần tử 0 và 1 hoặc 1 và 1, các phần tử kế tiếp được thiết lập theo qui tắc mỗi phần tử bằng tổng của 2 phần tử trước nó. Công thức truy hồi của dãy Fibonaci
  - F(1) = F(2) = 1
  - F(n) = F(n-1) + F(n-2), với n > 2
- Lập logic cho chương trình nhập vào số tự nhiên n > 0 và xuất ra dãy n số Fibonaci đầu tiên. Ví dụ, nhập n = 4 thì xuất ra 1, 1, 2, 3.

Bài tập lập trình (2)

- Logic bằng ngôn ngữ tự nhiên

Bước 0: Bắt đầu
Bước 1: Nhập vào n
Bước 2: Đặt các biến t = 0, f = 1, i = 1
Bước 3: Xuất ra f // xuất ra F(1)
Bước 4: Lặp lại trong khi i ≠ n
    Bước 4.1: t = f - t
    Bước 4.2: i = i + 1
    Bước 4.3: Xuất ra f // xuất ra F(i)
    Bước 4.4: f = f + t
Bước 5: Kết thúc

Hình A.1

Bài tập lập trình (3)

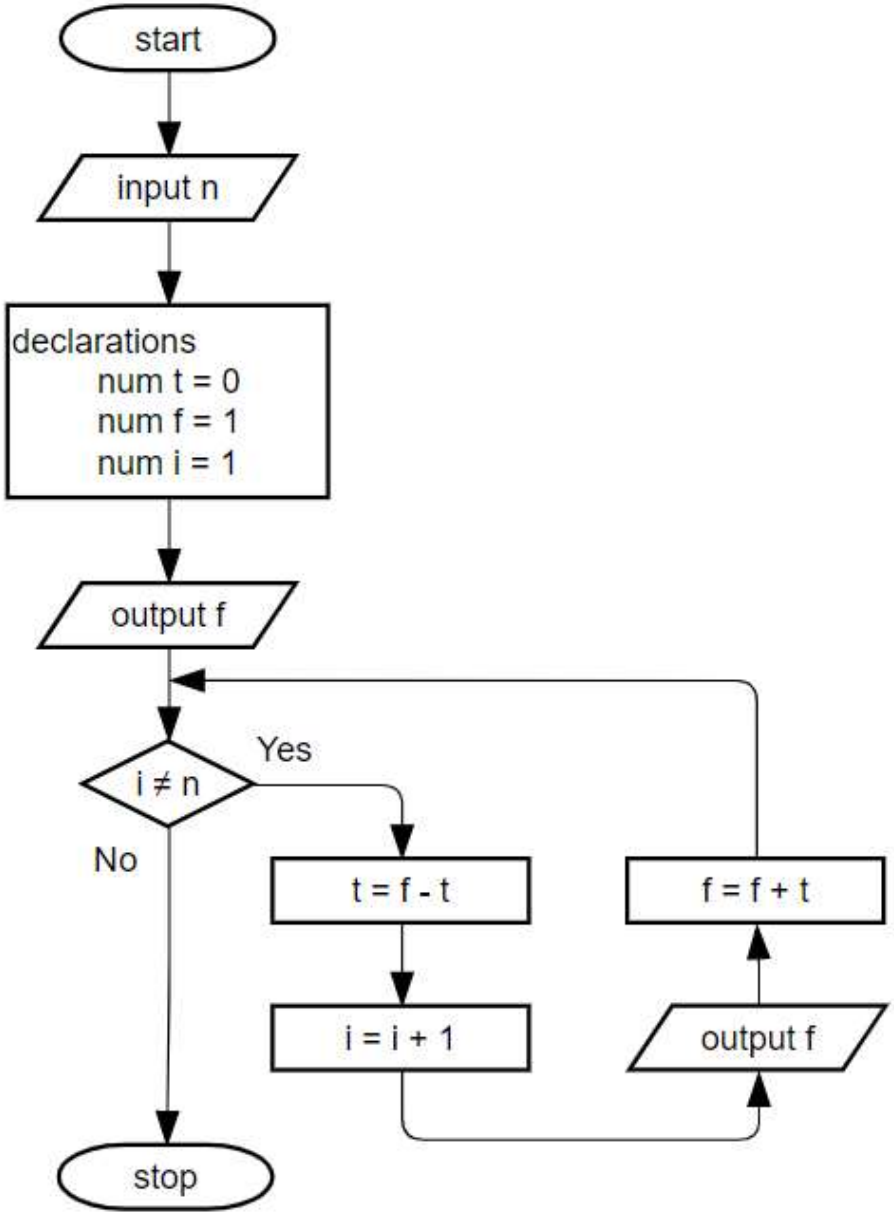
Bước 0:	Bắt đầu	
Bước 1:	Nhập vào n	
Bước 2:	Đặt các biến t = 0, f = 1, i = 1	
Bước 3:	Xuất ra f	// xuất ra F(1)
Bước 4:	Nếu i = n thì thực hiện *Bước 10*	
Bước 5:	t = f - t	
Bước 6:	i = i + 1	
Bước 7:	Xuất ra f	// xuất ra F(i)
Bước 8:	f = f + t	
Bước 9:	Quay lại *Bước 4*	
Bước 10:	Kết thúc	

Hình A.2

Bài tập lập trình (3)

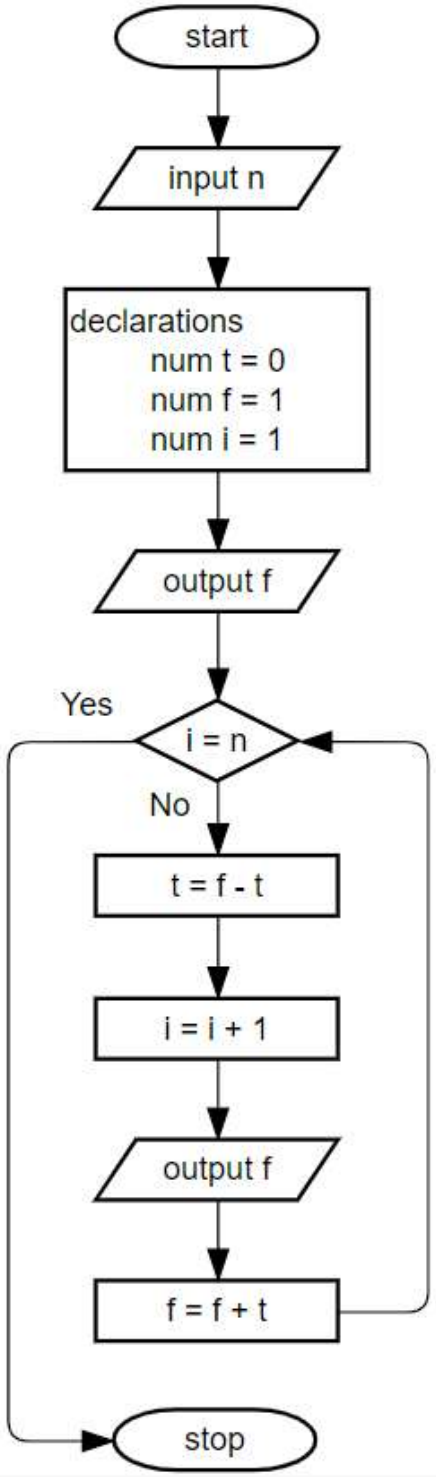
- Logic bằng lưu đồ

Hình B.1
Lưu đồ cho logic A.1
có cấu trúc, **thiếu khai báo biến n**



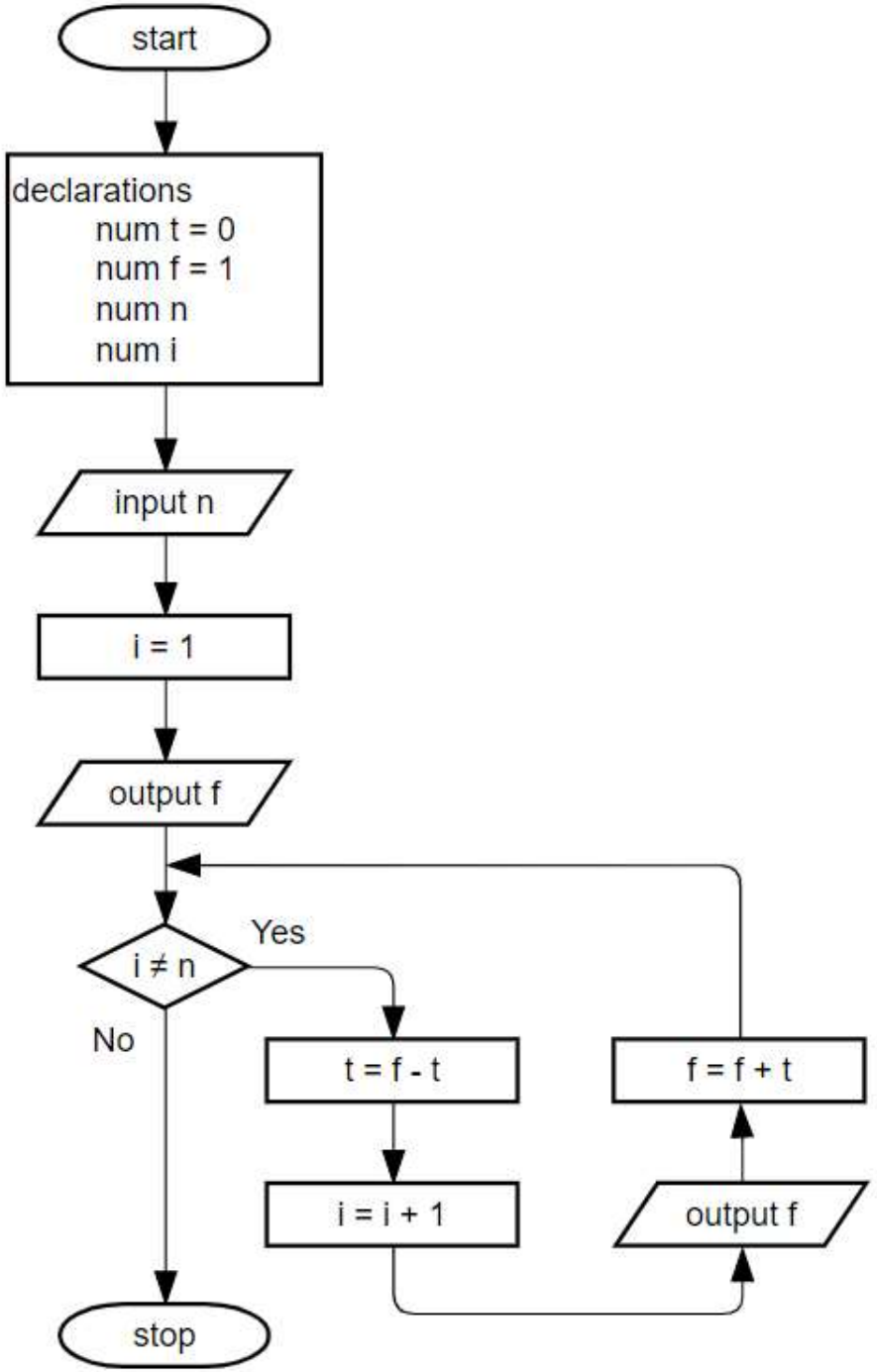
Bài tập lập trình (4)

Hình B.2
Lưu đồ cho logic A.2
không có cấu trúc



Bài tập lập trình (5)

Hình C
Lưu đồ cho logic có cấu trúc



Bài tập lập trình (6)

- Logic bằng mã giả

```
start
     declarations
          num t = 0
          num f = 1
          num n, i
     input n
     i = 1
     output f
     while i ≠ n
          t = f – t
          i = i + 1
          output f
          f = f + t
     endwhile
stop
```