*Bài 1*

# Logic lập trình
# Tổng quan
# về máy tính và lập trình

# Nội dung

- Hệ thống máy tính
- Logic chương trình
- Chu trình phát triển chương trình
- Mã giả và lưu đồ
- Dừng chương trình bằng giá trị lính canh
- Môi trường lập trình và người dùng
- Sự phát triển của các mô hình lập trình

# Hệ thống máy tính (1)

- Hệ thống máy tính
  - Sự kết hợp tất cả các thành phần cần thiết để xử lý và lưu trữ dữ liệu bằng máy tính.
  - Một hệ thống máy tính bao gồm nhiều phần cứng và phần mềm.
- Phần cứng (hardware)
  - Các thiết bị (devices) liên quan đến máy tính. Ví dụ: bàn phím, chuột, máy in, ổ cứng.
- Phần mềm (software)
  - Các chỉ thị máy tính (computer instruction) ra lệnh cho phần cứng biết phải làm gì.
  - Các chương trình (program), là tập hợp các chỉ thị được viết bởi lập trình viên.

# Hệ thống máy tính (2)

- Phần mềm ứng dụng (application software) là các chương trình sử dụng cho một nhiệm vụ như
  - Xử lý văn bản (Notepad, MS Word)
  - Bảng tính (MS Excel, Google Sheets)
  - Trò chơi (Solitaire, Minesweeper)
- Phần mềm hệ thống (system software) là các chương trình dùng để quản lý máy tính như
  - Hệ điều hành cho máy tính lớn (Linux, Windows, macOS)
  - Hệ điều hành cho thiết bị di động (Android, iOS)

# Hệ thống máy tính (3)

- Phần cứng và phần mềm máy tính thực hiện 3 hoạt động chính:
  - Đầu vào / nhập (input) - các mục dữ liệu như văn bản, số, hình ảnh, âm thanh, chuyển động vuốt chuột hay ngón tay được đưa vào hệ thống máy tính và được đặt trong bộ nhớ, nơi chúng có thể được xử lý.
  - Xử lý (processing) – thực hiện các phép tính và phép so sánh với đầu vào bởi bộ xử lý trung tâm (CPU)
  - Đầu ra / xuất (output) – thông tin thu được sau khi xử lý đầu vào được gửi đến máy in, màn hình, hoặc các thiết bị lưu trữ để mọi người có thể xem, diễn giải và sử dụng.

# Hệ thống máy tính (5)

- Ngôn ngữ lập trình (programming language)
  - Dùng để viết các chỉ thị.
  - Ví dụ: C, C++, Visual Basic, Java, C#, Python
- Cú pháp (syntax)
  - Là các qui tắc về cách sử dụng từ và dấu câu khi viết chỉ thị.
- Mã chương trình (program code)
  - Tập hợp các chỉ thị của một chương trình được viết bằng ngôn ngữ lập trình.
- Mã nguồn (source code)
  - Các câu lệnh (statement) được viết bằng ngôn ngữ lập trình.

# Hệ thống máy tính (6)

- Mã chương trình viết bởi hợp ngữ

```
LOAD 2000    : tải giá trị lưu tại vị trí bộ nhớ 2000H vào bộ
               tích lũy A (giả sử là 10)
MOV B, A     : lưu nội dung vào thanh ghi B từ A
               (B = A = 10)
LOAD 2002    : tải giá trị lưu tại vị trí bộ nhớ 2002 vào bộ
               tích lũy A (giả sử là 5)
ADD B        : cộng A+B và lưu kết quả vào A
               (A = A + B)
STORE 2004   : lưu trữ kết quả tại vị trí bộ nhớ 2004 tức là 15
```

- Mã nguồn viết bằng ngôn ngữ C

```
a = 10;
b = 5;
c = a + b;
```

# Hệ thống máy tính (7)

- Bộ nhớ máy tính (computer memory)
  - Là nơi lưu trữ các chương trình và các mục dữ liệu của máy tính.
- Bộ nhớ trong là vùng lưu trữ tạm thời
  - Bộ nhớ truy cập ngẫu nhiên (RAM) lưu trữ các chương trình đang chạy và các mục dữ liệu đang được sử dụng.
  - Nội dung chứa trên đó bị mất khi máy tính tắt.
- Bộ nhớ ngoài là vùng lưu trữ lâu dài
  - Thiết bị lưu trữ vĩnh viễn – như ổ cứng, thẻ nhớ.
  - Nội dung chứa trên đó không bị mất khi máy tính tắt.

# Hệ thống máy tính (8)

- Trình biên dịch (compiler) hay trình thông dịch (interpreter)
  - Chương trình để dịch mã nguồn sang các lệnh ở dạng ngôn ngữ máy (ngôn ngữ nhị phân) được gọi là mã đối tượng.
  - Kiểm tra các lỗi về cú pháp.
- Thực thi hay chạy chương trình
  - Quá trình các chỉ thị của chương trình được thực hiện.
  - Một số đầu vào sẽ được chấp nhận, một số xử lý sẽ diễn ra và kết quả sẽ là đầu ra.

# Logic chương trình

- Logic của chương trình máy tính
  - Chuỗi các chỉ thị được viết theo một trình tự xác định dẫn đến đầu ra chính xác.

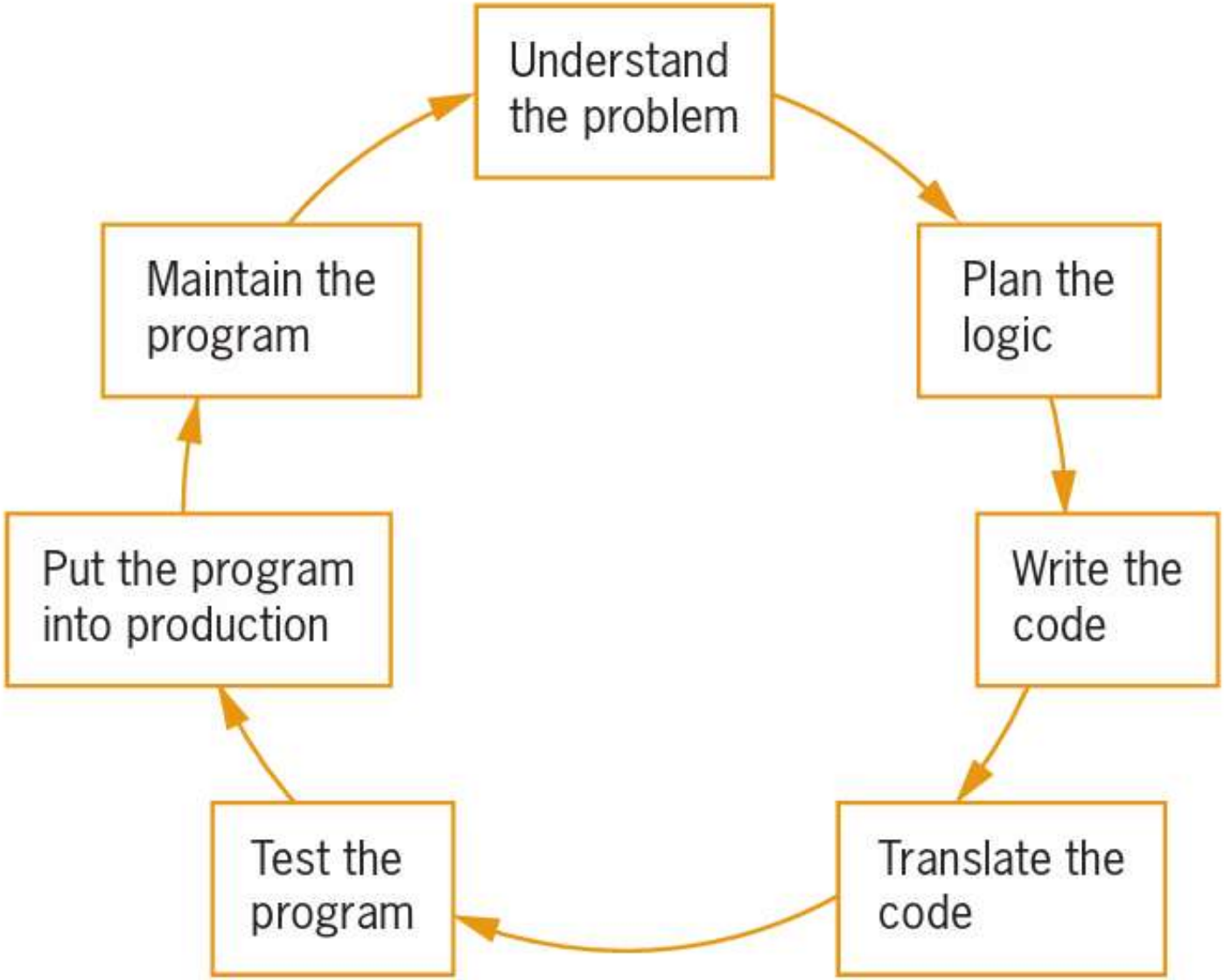
```
Nhập vào một số
Nhân số đó với 2
Xuất ra kết quả
```

- Các chương trình có lỗi cú pháp (syntax error) không thể dịch hoàn chỉnh và không thể thực thi được.
- Lỗi logic (logical error)
  - Lỗi trong logic của chương trình dẫn đến đầu ra không chính xác.
  - Nhiều lỗi logic không được nhận ra ngay lập tức.

# Chu trình phát triển chương trình (1)

- Có thể chia thành ít nhất 7 bước:
  - Tìm hiểu vấn đề.
  - Lập kế hoạch logic chương trình.
  - Viết mã chương trình.
  - Dịch mã chương trình.
  - Kiểm thử chương trình.
  - Đưa chương trình vào sản xuất.
  - Bảo trì chương trình.

# Chu trình phát triển chương trình (2)



# Tìm hiểu vấn đề

- Hiểu và xác định các yêu cầu của người sử dụng chương trình.
- Hiểu đầy đủ về vấn đề là một trong những khía cạnh khó khăn nhất của lập trình.
- Tài liệu (documentation) hỗ trợ cho chương trình thường được cung cấp giúp lập trình viên hiểu được vấn đề.

# Lập kế hoạch logic chương trình

- Trọng tâm của tiến trình lập trình
  - Xác định các bước và trình tự của chúng để dẫn đến giải pháp của vấn đề.
  - Có thể lập kế hoạch giải pháp cho một vấn đề bằng nhiều cách.
  - Nên suy nghĩ cẩn thận về tất cả các đầu vào mà chương trình có thể gặp và cách chương trình xử lý từng tình huống.
- Công cụ lập kế hoạch phổ biến nhất là lưu đồ (flowchart) và mã giả (pseudocode).
- Lập kế hoạch logic chương trình còn được gọi là phát triển thuật toán.
- Kiểm thử thủ công (desk-checking) là quá trình xem xét từng bước logic chương trình trên giấy.

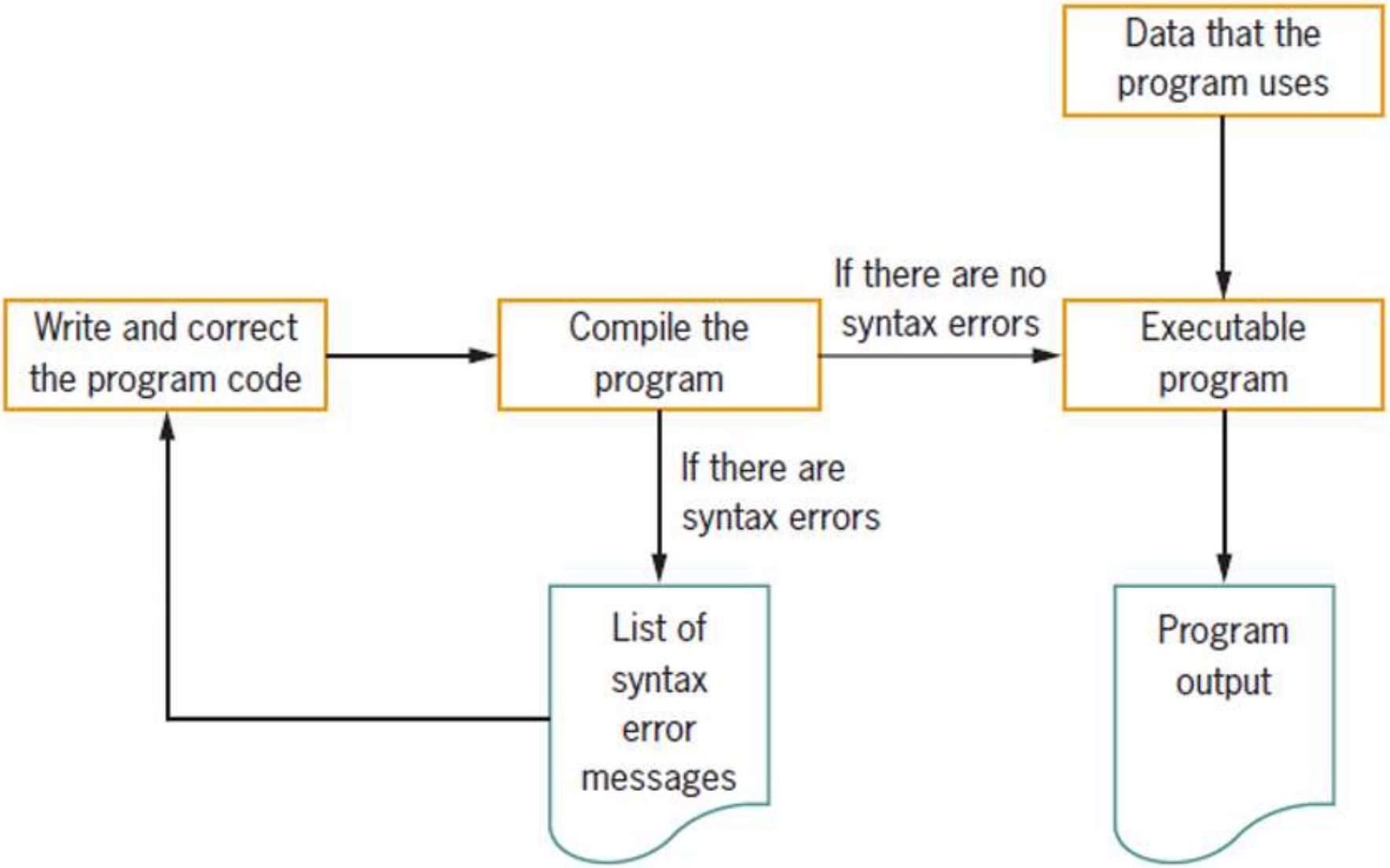
# Viết mã chương trình

- Lập trình viên lựa chọn một ngôn ngữ lập trình để viết mã nguồn cho chương trình.
  - Các ngôn ngữ lập trình dù khác biệt nhưng tương tự nhau về các tính năng cơ bản như xử lý các hoạt động đầu vào, số học, các hoạt động đầu ra và các hàm tiêu chuẩn.
  - Ngôn ngữ được chọn dựa trên các đặc tính mang lại sự hiệu quả.
- Bước nào khó hơn: lập kế hoạch logic hay viết mã chương trình?
  - Viết mã chương trình dễ hơn.

# Dịch mã chương trình (1)

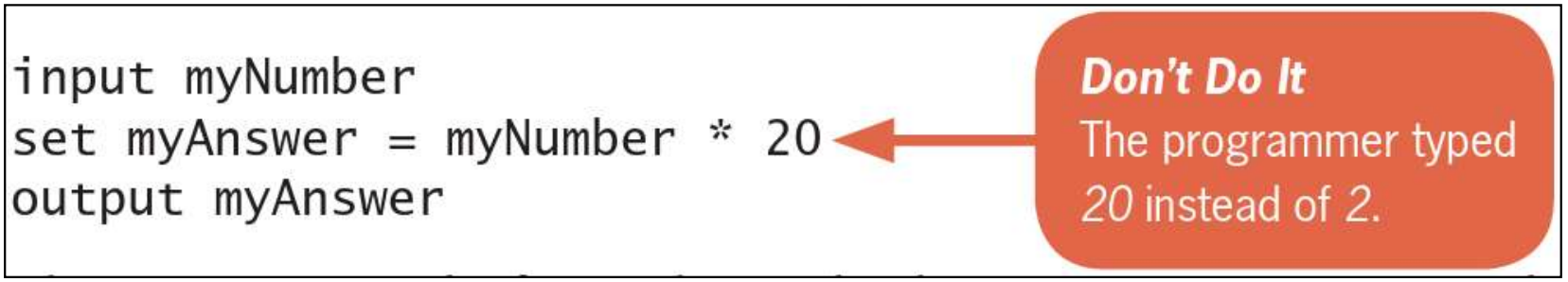
- Sử dụng trình biên dịch hoặc trình thông dịch để chuyển mã nguồn sang ngôn ngữ máy.
- Lỗi cú pháp (syntax error)
  - Sử dụng sai cú pháp của ngôn ngữ như viết từ sai chính tả, từ không có trong danh mục, dùng sai dấu câu.
- Lập trình viên sửa các lỗi cú pháp được liệt kê và có thể dịch lại mã nhiều lần.
- Chương trình thực thi được tạo ra khi mã nguồn không có lỗi cú pháp, được lưu lại và có thể chạy mà không cần lặp lại bước dịch.

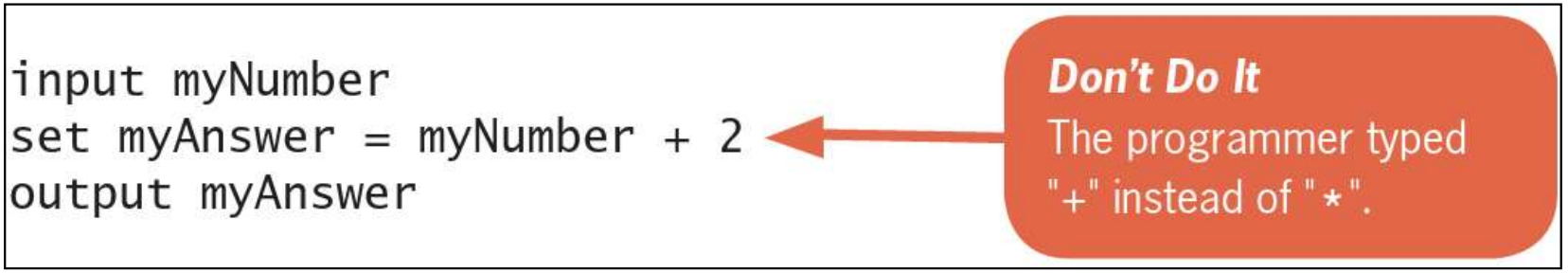
# Dịch mã chương trình (2)



# Kiểm thử chương trình

- Chạy chương trình với nhiều mẫu đầu vào để kiểm tra tính chính xác của đầu ra.
- Gỡ lỗi (debugging) là quá trình tìm và sửa lỗi logic của chương trình.



# Đóng gói chương trình

- Tùy vào mục đích của chương trình
  - Đơn giản là chạy chương trình một lần nếu nó được viết để đáp ứng yêu cầu của người dùng về một công việc đặc biệt.
  - Mất nhiều thời gian nếu chương trình được chạy thường xuyên hoặc nếu đó là một hệ thống chương trình lớn đang được phát triển.
- Bước chuyển đổi (conversion)
  - Toàn bộ các hành động mà một tổ chức phải thực hiện để chuyển sang sử dụng một chương trình hay một nhóm chương trình mới.

# Bảo trì chương trình

- Bảo trì
  - Thực hiện các thay đổi sau khi chương trình được đưa vào sản xuất.
- Duy trì
  - Giữ cho các chương trình đã viết trước chạy ổn định.
- Cải tiến
  - Thực hiện thay đổi đối với các chương trình hiện có.
  - Lặp lại chu trình phát triển.

# Mã giả và lưu đồ

- Mã giả (pseudocode)
  - Là cách trình bày tựa tựa tiếng Anh các bước logic cần thực hiện để giải quyết vấn đề.
  - Các câu lệnh có vẻ như được viết bằng ngôn ngữ lập trình nhưng không nhất thiết phải đúng cú pháp của bất kỳ ngôn ngữ cụ thể nào.
- Lưu đồ / sơ đồ khối (flowchart)
  - Là sự thể hiện bằng hình ảnh các bước logic cần thực hiện để giải quyết vấn đề.

# Viết mã giả

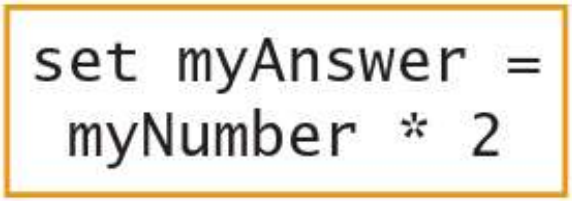
```
start
  input myNumber
  set myAnswer = myNumber * 2
  output myAnswer
stop
```

- Cách viết mã giả thông thường:
  - Câu lệnh khởi động chương trình – *start*
  - Câu lệnh dừng chương trình – *stop*
  - Các câu lệnh khác (tựa tựa tiếng Anh) được viết thụt lề.
- Mã giả là công cụ lập kế hoạch khá linh hoạt
  - Có thể dùng từ *begin, end* thay cho *start, stop*; *get, read* thay cho *input*; *calculate, compute* thay cho set; *print, write* thay cho *output*.

# Vẽ lưu đồ (1)

- Cách tạo một lưu đồ
  - Vẽ các dạng hình học bên trong có chứa các câu lệnh riêng lẻ.
  - Các hình được kết nối với nhau bằng các mũi tên.
- Ký hiệu đầu vào
  - Hình bình hành bên trong chứa câu lệnh *input.*
  - Biểu thị hoạt động nhập


- Ký hiệu xử lý
  - Hình chữ nhật bên trong chứa các câu lệnh như *set, calculate*.
  - Biểu thị hoạt động xử lý.

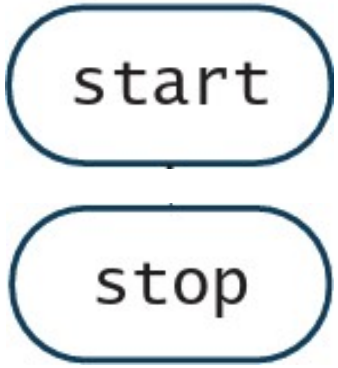


# Vẽ lưu đồ (2)

- Ký hiệu đầu ra
  - Hình bình hành bên trong chứa câu lệnh output.
  - Biểu thị hoạt động xuất.

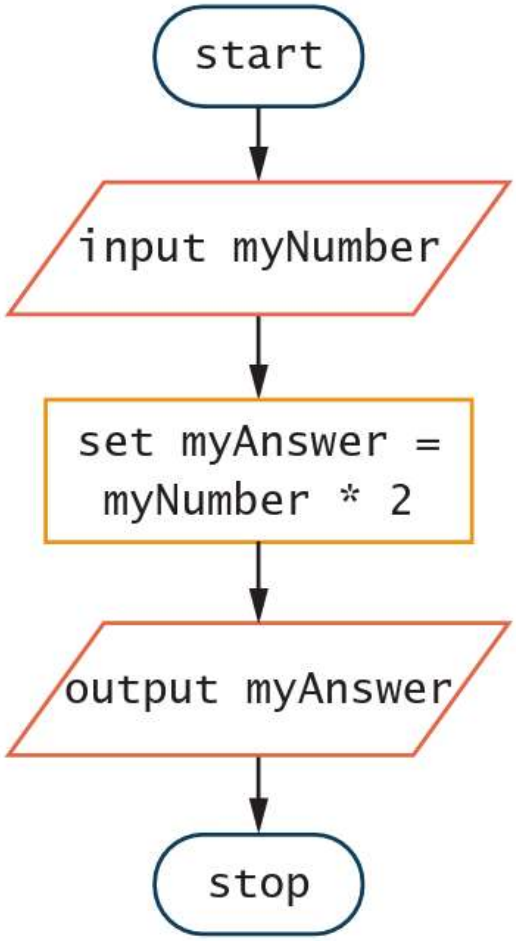

- Ký hiệu dòng chảy
  - Mũi tên kết nối các hình.
  - Biểu thị trình tự chính xác của các câu lệnh.

- Ký hiệu đầu cuối
  - Hình tròn, hình viên nhộng bên trong chứa câu lệnh *start*, *stop*.
  - Biểu thị điểm bắt đầu và kết thúc của lưu đồ. Mỗi lưu đồ chỉ có 2 ký hiệu.



# Vẽ lưu đồ (3)

**Flowchart**



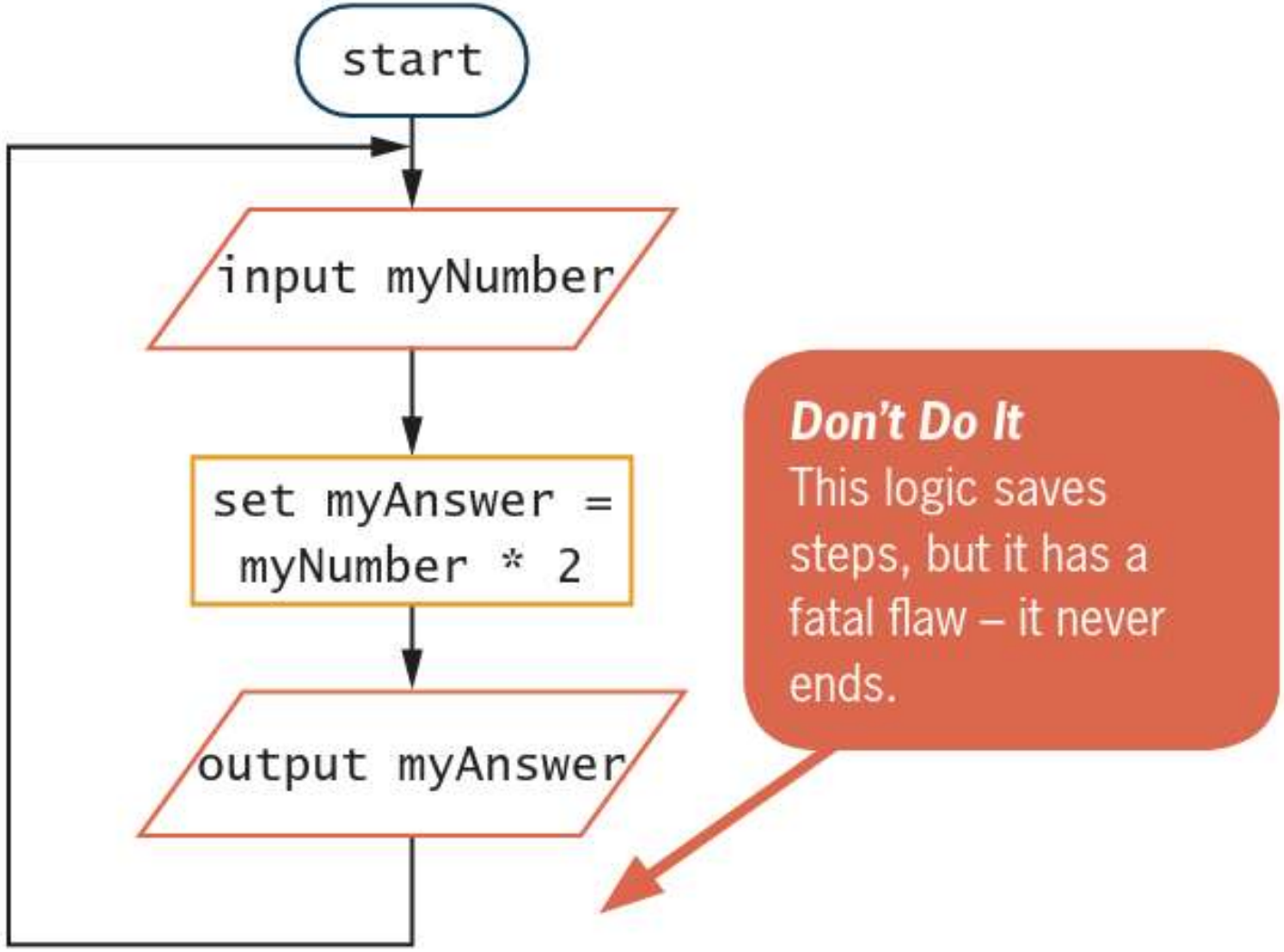

**Pseudocode**

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

# Dừng chương trình bằng giá trị lính canh (1)

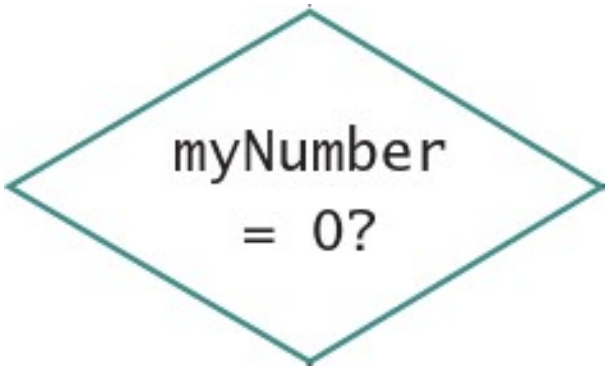
- Thực thi một chương trình nhiều lần
  - Không khả thi khi chạy đi chạy lại chương trình 10,000 lần.
  - Không khả thi khi thêm 10,000 lần các chỉ thị vào một chương trình.
  - Giải pháp là tạo một vòng lặp (lặp lại một chuỗi các chỉ thị của chương trình).
  - Tránh vòng lặp vô hạn (lặp lại luồng logic mà không bao giờ kết thúc).

# Dừng chương trình bằng giá trị lính canh (2)

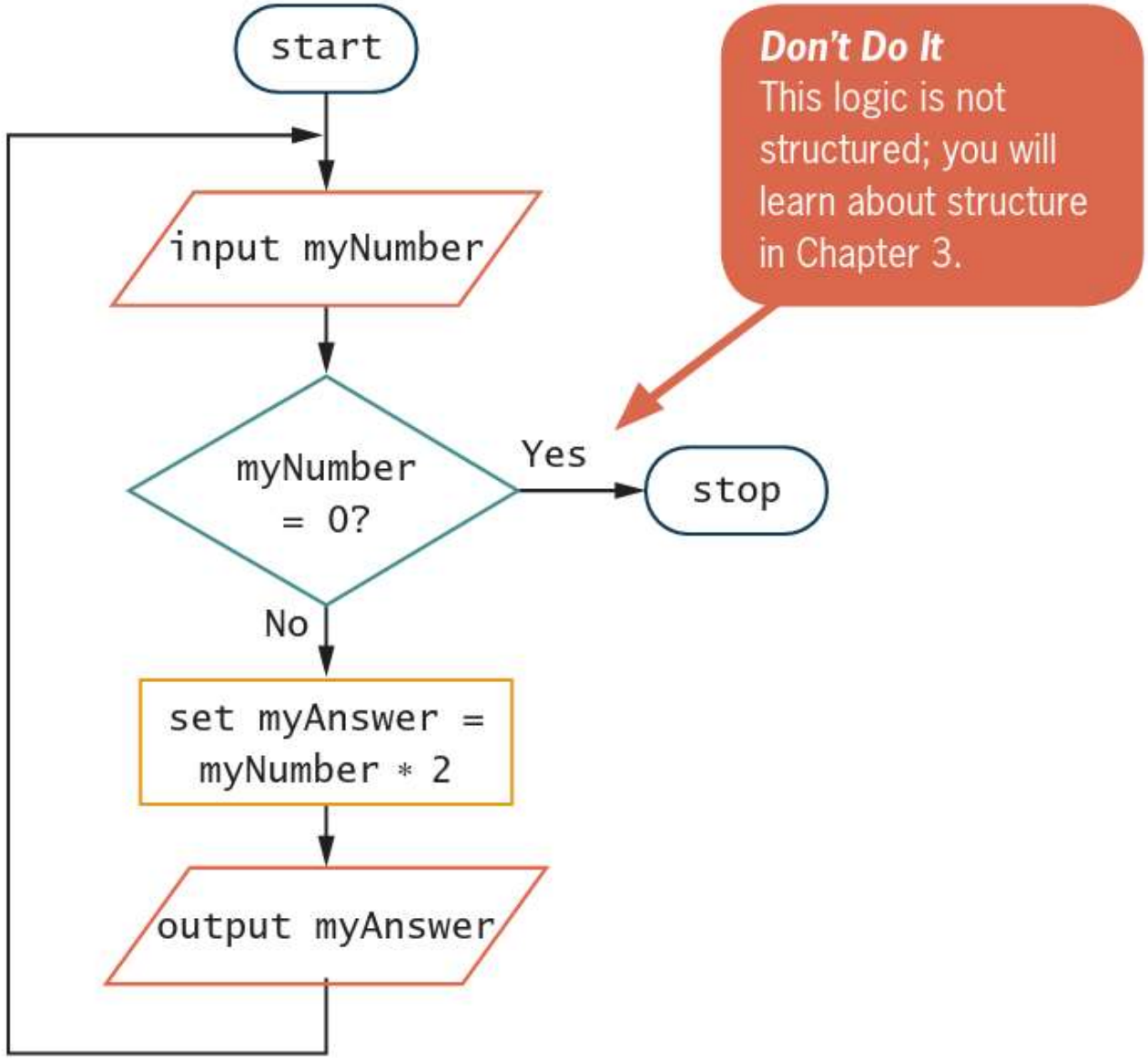


# Dừng chương trình bằng giá trị lính canh (3)

- Tạo điểm kết thúc luồng logic bằng cách đưa ra một quyết định
  - Đặt một giá trị định trước cho đầu vào có nghĩa “Dừng chương trình”, còn gọi là **giá trị lính canh** (sentinel).
    - Giá trị đầu vào mà người dùng không bao giờ cần.
  - Kiểm tra đầu vào, nếu nó bằng giá trị lính canh thì dừng chương trình.
- Ký hiệu quyết định
  - Hình thoi bên trong chứa một câu hỏi có câu trả lời thường là Có / Không hoặc Đúng / Sai.
  - Biểu thị một quyết định cho bước tiếp theo.



# Dừng chương trình bằng giá trị lính canh (4)



# Môi trường lập trình và người dùng

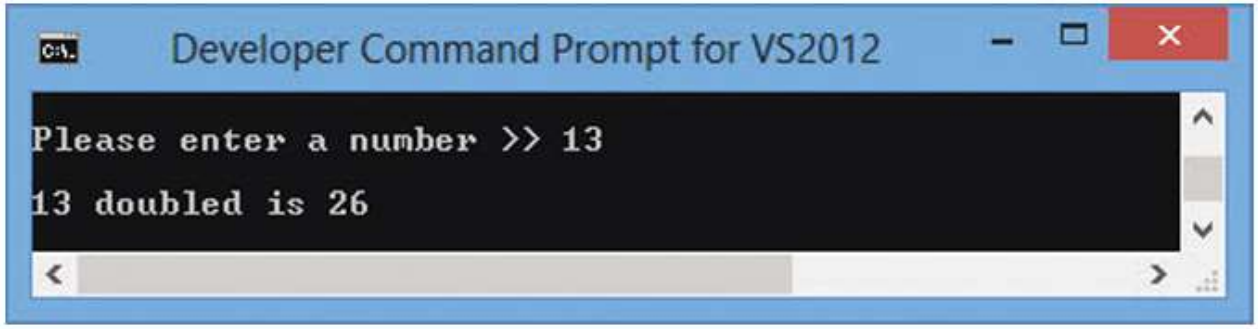
- Môi trường lập trình và người dùng là các công cụ và cách thức cho phép lập trình viên viết chương trình và người dùng thực thi chương trình. Có nhiều sự lựa chọn:
  - Lập kế hoạch logic
    - Viết mã giả, vẽ lưu đồ bằng tay hoặc phần mềm.
  - Viết mã chương trình
    - Bằng các trình soạn thảo.
  - Thực thi chương trình
    - Với đầu vào từ bàn phím, chuột, micro.
  - Kết xuất của chương trình
    - Với đầu ra là văn bản, hình ảnh, âm thanh.

# Môi trường lập trình

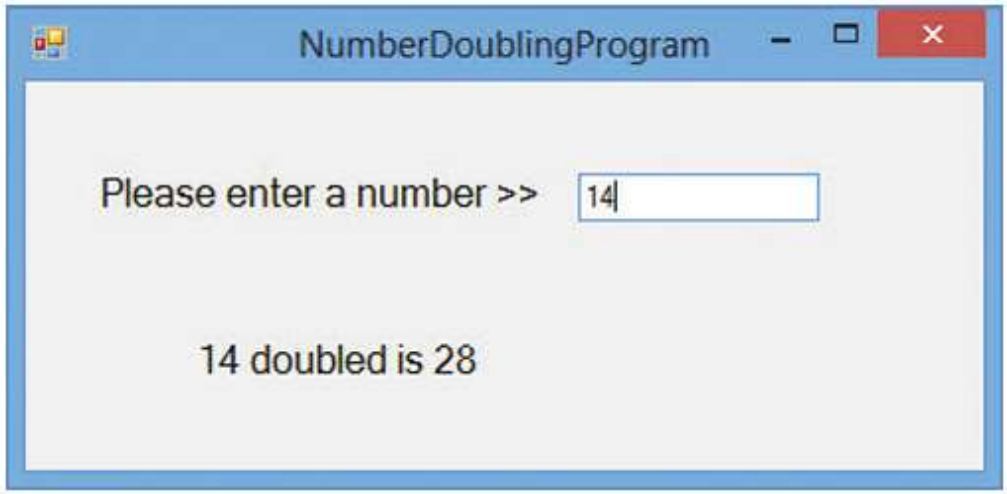
- Môi trường phát triển tích hợp (IDE)
  - Một gói phần mềm cung cấp trình soạn thảo mã nguồn, trình biên dịch và các công cụ lập trình khác như quản lý phiên bản, xây dựng giao diện đồ họa người dùng.
  - Ví dụ: Visual Studio, Eclipse
- Trình soạn thảo mã nguồn của IDE có nhiều tính năng:
  - Làm nổi bật các thành phần của ngôn ngữ lập trình với các màu khác nhau.
  - Làm nổi bật các lỗi cú pháp.
  - Hoàn thành câu lệnh tự động.
  - Cung cấp công cụ thi hành từng câu lệnh.

# Môi trường người dùng

- Môi trường dòng lệnh
  - Dòng lệnh là một vị trí trên màn hình máy tính nơi bạn nhập các mục văn bản để giao tiếp với hệ điều hành của máy tính.




- Môi trường giao diện đồ họa người dùng (GUI)
  - Cho phép người dùng tương tác với chương trình trong môi trường đồ họa.



# Sự phát triển của các mô hình lập trình (1)

- Các chương trình máy tính hiện đại được viết từ những năm 1940.
- Các ngôn ngữ lập trình cấp thấp
  - Yêu cầu phải làm việc với các địa chỉ bộ nhớ và ghi nhớ các mã khó liên quan đến ngôn ngữ máy.
- Các ngôn ngữ lập trình cấp cao
  - Tựa tựa ngôn ngữ tự nhiên hơn.
  - Dễ sử dụng hơn, một phần vì chúng cho phép đặt tên biến thay vì sử dụng các địa chỉ bộ nhớ.
  - Cho phép tạo các mô-đun hoặc phân đoạn chương trình độc lập có thể được ghép lại với nhau theo nhiều cách khác nhau.

# Sự phát triển của các mô hình lập trình (2)

- Hai mô hình lập trình đáng chú ý
  - Lập trình thủ tục (procedural programming)
    - Tập trung vào các thủ tục do người lập trình tạo ra tương ứng với các hành động cần thực hiện.
  - Lập trình hướng đối tượng (object-oriented programming)
    - Tập trung vào các đối tượng, mô tả các đặc điểm và các hành vi của chúng.
- Lập trình hướng dịch vụ (service-oriented programming)
  - Được xây dựng trên OOP, trong đó các vấn đề có thể được mô hình hóa theo các dịch vụ mà một đối tượng cung cấp hoặc sử dụng.