

Bài 01:

Giới Thiệu Tổng Quan

Về Lập Trình

Giảng Viên: ThS. Giang Hào Côn

Nội dung

- Khái niệm về chương trình máy tính
- Các ngôn ngữ lập trình
- Các khái niệm cơ bản về lập trình
- Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp
- Thuật ngữ và bài đọc thêm tiếng Anh

Khái niệm về chương trình máy tính

Chương trình mã máy - Khái niệm

- CPU của máy tính được thiết kế để có thể thực hiện được các **chương trình mã máy (machine code program)** đã được hệ điều hành (HĐH) nạp vào RAM của máy tính.
- Chương trình mã máy thường phải tương thích với từng họ máy cụ thể, bao gồm tập hợp các chỉ thị được viết bằng các lệnh CPU của họ máy đó, được lưu trên đĩa dưới dạng một tập tin mã thực thi (executable program file) của HĐH cụ thể.

Khái niệm về chương trình máy tính

Chương trình mã máy – Quy trình thực hiện

- **B1.** Người sử dụng (**người dùng cuối – end user**) ra lệnh thực hiện (**chạy**) chương trình.
- **B2.** HĐH nhận được lệnh sẽ thực hiện:
 - Tìm và nạp tập tin mã thực thi của chương trình (nằm trên đĩa) vào RAM của máy tính.
 - Bộ đếm lệnh của CPU (**CPU program counter**) được trỏ đến lệnh đầu tiên của chương trình (còn gọi là **ngõ vào chương trình – program entry point**).

Khái niệm về chương trình máy tính

Chương trình mã máy – Quy trình thực hiện

- **B3.** CPU thực hiện từng chỉ thị một trong RAM cho đến khi gặp lệnh kết thúc:
 - Chép lệnh mã máy hiện hành vào thanh ghi lệnh.
 - Tăng bộ đếm lệnh (để trỏ đến lệnh kế tiếp).
 - Thi hành lệnh mã máy.
- **B4.** Kết thúc thực hiện chương trình, HĐH chờ nhận lệnh mới.

Khái niệm về chương trình máy tính

Chương trình mã máy – Đặc điểm

- Mỗi **chỉ thị** của chương trình là **một lệnh mã máy** (một dãy các byte chỉ phù hợp với qui ước tập lệnh của một loại CPU nào đó)
- Được cấu trúc hóa theo qui ước của HĐH.
- Được chạy trên một họ CPU và HĐH cụ thể.
- Nội dung rất khó hiểu đối với người dùng máy tính, chỉ có CPU thích hợp với hiểu rõ và thi hành được

Khái niệm về chương trình máy tính

Chương trình mã máy – Nhận xét

- **Khó** có thể sản xuất ra phần mềm bằng cách viết trực tiếp các chương trình mã máy.
- Nếu có làm được theo cách này thì:
 - **Giá cả** sẽ rất đắt do quá khó, tốn quá nhiều thời gian và công sức.
 - **Khả năng dùng lại** rất giới hạn do không thể bán cho người dùng trên họ máy tính khác hay người dùng sử dụng hệ điều hành khác.

Khái niệm về chương trình máy tính

Chương trình nguồn – Khái niệm

- Việc viết các chương trình mã máy **rất cực** và **kém hiệu quả** ngay cả đối với các lập trình viên chuyên nghiệp vì vậy giải pháp khởi đầu là sử dụng các NNLT cấp thấp như hợp ngữ hoặc các NNLT cấp cao.
- Chương trình viết bằng NNLT được gọi là **chương trình nguồn** (**source code program**) hay **mã nguồn** (**source code**).

Khái niệm về chương trình máy tính

Chương trình nguồn – Chương trình dịch

- Do máy tính chỉ có thể hiểu được ngôn ngữ máy, cho nên một chương trình sau khi đã được lập trình (viết bằng ngôn ngữ cấp cao, hợp ngữ) cần phải chuyển đổi thành ngôn ngữ máy thì mới có thể thi hành được.
- Những công cụ làm nhiệm vụ chuyển đổi cho mục đích này thường được gọi là **chương trình dịch**.
- Chương trình dịch thường được phân biệt ở một trong hai dạng: **Trình thông dịch (Interpreter)** và **trình biên dịch (Compiler)**.

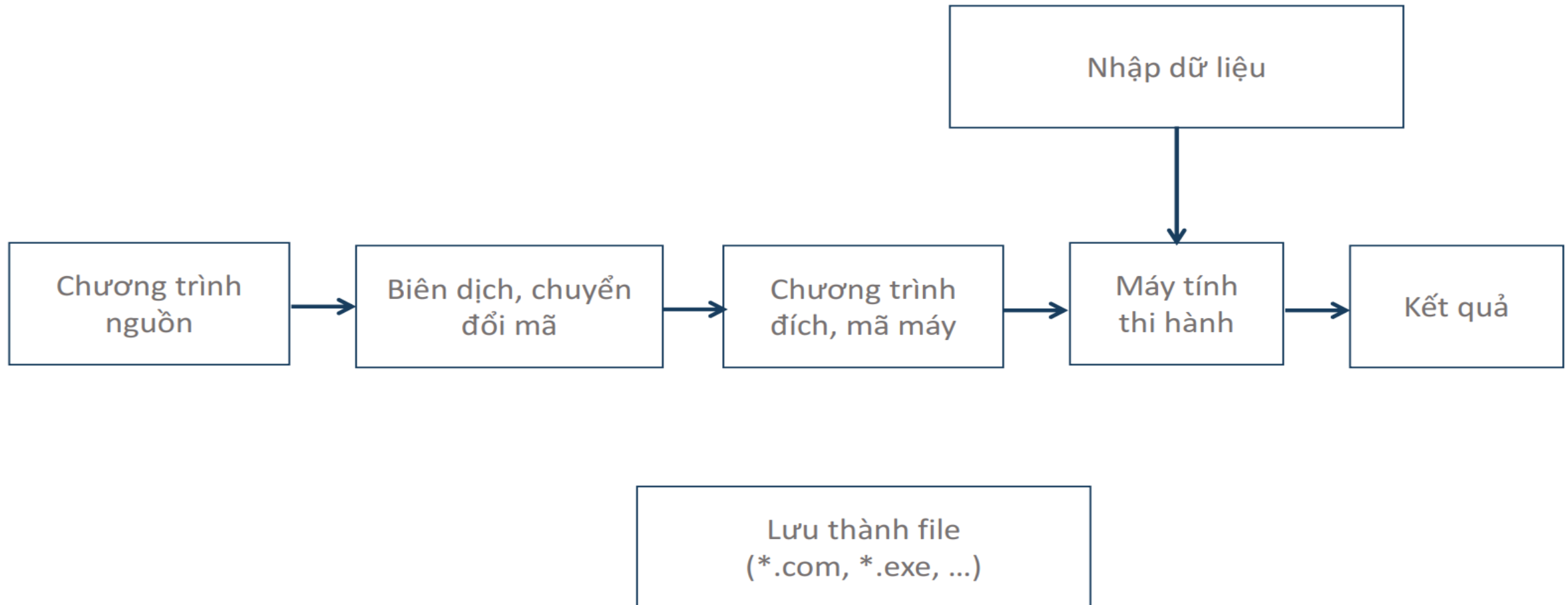
Khái niệm về chương trình máy tính

Chương trình nguồn - Trình biên dịch

- Làm nhiệm vụ chuyển đổi một chương trình đã được viết bằng một ngôn ngữ lập trình nào đó (chương trình nguồn) thành ngôn ngữ máy (chương trình đích).
- Quá trình chuyển đổi từ chương trình nguồn thành chương trình đích thường được gọi là **thời gian dịch** (**Compile-time**) và thời gian thực thi chương trình sau khi đã biên dịch thành công được gọi là **thời gian thực thi** (**Run-time**)

Khái niệm về chương trình máy tính

Chương trình nguồn – Cơ chế biên dịch



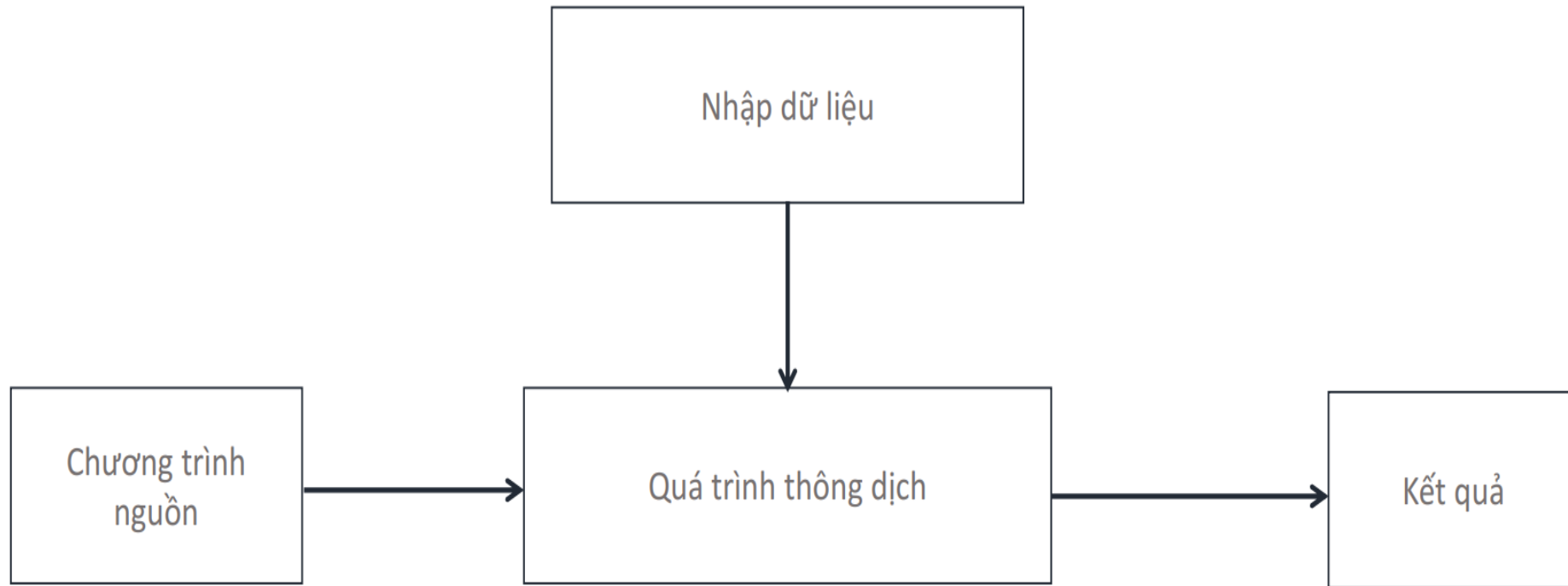
Khái niệm về chương trình máy tính

Chương trình nguồn – Trình thông dịch

- Những phần mềm có khả năng đọc và chuyển đổi mã nguồn của một chương trình đã viết bằng ngôn ngữ lập trình ra mã máy để ra lệnh cho máy tính thi hành được gọi là trình thông dịch.
- Khác với trình biên dịch, trình thông dịch sẽ dịch từng câu lệnh từ chương trình nguồn theo yêu cầu thực thi.
- Như vậy, thời gian dịch diễn ra đồng thời với thời gian thực thi chương trình, quá trình này gọi là **Thông dịch**.

Khái niệm về chương trình máy tính

Chương trình nguồn – Cơ chế thông dịch



Biên dịch và thông dịch khác nhau ở đây?

Các ngôn ngữ lập trình

Đặc điểm

- **Ngôn ngữ lập trình (programming language)** là ngôn ngữ được lập trình viên sử dụng để viết chương trình cho máy tính.
- Khi một chương trình được viết bằng một NNLT nào đó thì các chỉ thị, câu lệnh trong chương trình phải tuân theo các qui tắc, các luật do NNLT đó qui định.

Các ngôn ngữ lập trình

NNLT cấp thấp - Đặc điểm

- Là ngôn ngữ lập trình phụ thuộc vào từng họ máy cụ thể, vì vậy không có tính tương thích.
- Dễ viết, đọc, sửa hơn chương trình mã máy.
- Ưu điểm là tận dụng và khai thác được tính năng của mỗi họ máy cụ thể, nhờ vậy chương trình có thể chạy nhanh hơn.

Các ngôn ngữ lập trình

NNLT cấp cao – Đặc điểm

- Được đề xuất để khắc phục các hạn chế của NNLT cấp thấp
- Dễ dùng và dễ diễn đạt được các ý tưởng trừu tượng.
- Có tính tương thích cao (khi thay đổi dạng máy tính thì chỉ cần sửa chương trình rất ít hoặc thậm chí không cần sửa mà vẫn đảm bảo chạy đúng).

Một vài NNLT thông dụng

Ngôn ngữ lập trình cấp thấp

- Hợp ngữ (assembly language)

Ngôn ngữ lập trình cấp cao

- C/C++
- COBOL
- FORTRAN
- Java, C#
- PHP, Ruby, Perl
- Ada, BASIC, Visual Basic (VB), Lisp, Pascal, ...

Các khái niệm cơ bản về lập trình

Khái niệm cơ bản

- **Một chương trình (program)** là một dãy các chỉ thị (instruction) điều khiển sự hoạt động của máy tính nhằm giải quyết một công việc nào đó.
- **Người viết chương trình** (còn gọi là lập trình viên hay thảo chương viên – programmer) là những người tạo lập ra những chương trình máy tính.

Các khái niệm cơ bản về lập trình

Chương trình minh họa

- Hai chương trình đơn giản sau chỉ in ra một dòng chữ có nội dung là “Hello everybody!” bằng NNLT Java và C.

	Chương trình Java	Chương trình C
1	<code>// Hello.java</code>	<code>/* Hello.c */</code>
2	<code>import java.util.*;</code>	<code>#include <stdio.h></code>
3	<code>public class Hello {</code>	
4	<code> public static void main(String argv[])</code>	<code>void main(void)</code>
5	<code>{</code>	<code>{</code>
6	<code> System.out.print("Hello everybody!");</code>	<code> printf("Hello everybody!");</code>
7	<code>}</code>	<code>}</code>
8	<code>}</code>	

Các khái niệm cơ bản về lập trình

Viết >> Dịch >> chạy Chương trình

- Đối với các ngôn ngữ lập trình cấp cao truyền thống (trước thế hệ của Java và C#), quá trình viết, dịch và chạy chương trình gồm các công đoạn như sau:
 - **B1.** Soạn chương trình nguồn và lưu lên đĩa.
 - **B2.** Dịch chương trình nguồn nhờ trình biên dịch.
 - **B3.** Nối kết các tập tin mã trung gian tạo ra ở B2.
 - **B4.** Chạy chương trình ngôn ngữ máy tạo ra ở B3

Công nghệ lập trình hiện đại

- **Hạn chế** của các chương trình cấp cao truyền thống là trình biên dịch của chúng phát sinh trực tiếp mã thực thi **phụ thuộc vào mã máy tính của một họ máy tính và hệ điều hành cụ thể** nên không thể mang đi sử dụng ở các hệ điều hành khác.
- NNLT hiện đại như Java hay C# trình biên dịch không dịch trực tiếp mã nguồn thành mã thực thi mà được thiết kế để có thể dịch thành mã thực thi trừu tượng (abstract executable code) độc lập máy và hệ điều hành.

Công nghệ lập trình hiện đại

- Do máy tính thật không thể hiểu được mã trừu tượng nên những chương trình dạng mã thực thi trừu tượng chỉ chạy được khi có sẵn máy ảo hỗ trợ cho việc thi hành loại mã thực thi đó.
- Chương trình nguồn Java (tập tin *.java) được dịch thành mã thực thi không phụ thuộc máy tính (**tập tin *.class**) có thể chạy được trên bất kỳ máy tính nào đã cài đặt máy ảo Java (**Java Virtual Machine – JVM**).

Công nghệ lập trình hiện đại

- Trong các năm gần đây, các ứng dụng chạy trên web phát triển rất mạnh.
 - Chạy trên internet thông qua một trình duyệt web.
 - Được viết bằng các ngôn ngữ như **PHP, ASP.NET, JSP, Java Script, VB Script...** có tính tương thích cao, hoạt động trên bất kỳ máy tính nào có internet.

Môi trường lập trình

Toàn bộ qui trình biên dịch được thực một cách dễ dàng và thuận tiện nhờ vào công cụ gọi là môi trường phát triển phần mềm (**Integrated Development Environment – IDE**)

- Soạn thảo chương trình.
- Quản lý hệ thống tập tin mã nguồn.
- Quản lý hệ thống các phiên bản của mã nguồn.
- Kiểm tra lỗi cú pháp, biên dịch, liên kết chương trình.
- Chạy từng dòng lệnh (debug) để tìm lỗi.

Môi trường lập trình

Một số IDE thông dụng:

- Eclipse: hỗ trợ nhiều ngôn ngữ.
- **IntelliJ IDEA Communication**
- C++ Visual Studio: ngôn ngữ C++.
- C# Visual Studio: ngôn ngữ C#.
- Visual Café: ngôn ngữ Java.
- J Builder: ngôn ngữ Java.



▪ Thuật ngữ tiếng Anh

- **abstract executable code**: mã trừu tượng
- **assembler**: trình hợp dịch
- **assembly language**: hợp ngữ
- **compiler**: trình biên dịch
- **data type**: kiểu dữ liệu
- **debug**: chạy chương trình theo từng dòng lệnh để tìm lỗi
- **executable program file**: một tập tin mã thực thi

▪ Thuật ngữ tiếng Anh

- **end user(s)**: người sử dụng, người dùng cuối
- **IDE**: viết tắt của “Integrated Development Environment”, môi trường phát triển chương trình tích hợp
- **instruction**: chỉ thị
- **interpreter**: trình thông dịch
- **link**: nối kết các mã trung gian
- **linker** (hay link program): chương trình liên kết mã trung gian

Thuật ngữ và bài đọc thêm tiếng Anh

▪ Thuật ngữ tiếng Anh

- **machine code program**: chương trình mã máy
- **object code**: mã đối tượng, một loại mã trung gian chưa phải là mã máy thật sự
- **program entry point**: ngõ vào chương trình
- **program**: chương trình
- **programmer**: người viết chương trình, lập trình viên, từ cũ: “thảo chương viên”
- **programming language**: ngôn ngữ lập trình

Thuật ngữ và bài đọc thêm tiếng Anh

▪ Thuật ngữ tiếng Anh

- **low-level programming language**: ngôn ngữ lập trình cấp thấp
- **high-level programming language**: ngôn ngữ lập trình cấp cao
- **programming**: lập trình
- **source code program**: chương trình nguồn
- **source code**: mã nguồn
- **syntax error**: lỗi cú pháp
- **text editor**: trình soạn thảo văn bản (có thể dùng để soạn mã nguồn)

Bài đọc thêm tiếng Anh

- Jose M. Garrido, “**Object-Oriented Programming: From Problem Solving to Java**”, Charles River Media
- Paul Deitel, Harvey Deitel, “**Java : How to program**”, 9th edition, 2012
- Oracle, “**The Java™ Tutorials**”,
<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html>,
6:20PM, 18/01/2018
- Java tutorial, <https://www.javatpoint.com/java-tutorial> , 6:20PM,
18/01/2018

Thuật toán – giải thuật

- Máy tính là một công cụ đắc lực hỗ trợ con người trong việc tính toán và xử lý.
- Phát biểu **bài toán** bằng ngôn ngữ tự nhiên không thể là đầu vào cho máy tính.
- Con người phải **mô hình hóa bài toán** thông qua những cấu trúc dữ liệu, vốn được **hỗ trợ bởi các ngôn ngữ lập trình**, từ cơ sở đến nâng cao như mảng, cấu trúc, tập hợp, đồ thị, cây, ...
- Trên cơ sở mô hình dữ liệu đã được xây dựng, con người phải chỉ ra cho máy tính **một cách thức để giải quyết bài toán (gọi là thuật toán hay giải thuật)**.

Thuật toán – giải thuật

- Thuật toán có thể hiểu là một qui trình xử lý bao gồm các bước cụ thể có thể thực hiện để giải quyết một bài toán.

Đặc tính của thuật toán

- **Tính hữu hạn:** Thuật toán phải kết thúc thực thi sau một số lượng hữu hạn các bước xử lý.
- **Tính xác định:** Mỗi bước xử lý phải được mô tả rõ ràng, chính xác, không nhập nhằng.
- **Tồn tại dữ liệu đầu vào:** Thuật toán phải có dữ liệu đầu vào hợp lệ, được mô tả rõ ràng.

Thuật toán – giải thuật

- Thuật toán có thể hiểu là một qui trình xử lý bao gồm các bước cụ thể có thể thực hiện để giải quyết một bài toán.

Đặc tính của thuật toán

- **Tính có kết quả:** Thuật toán phải cho ra kết quả đúng trên cơ sở dữ liệu đầu vào hợp lệ.
- **Tính hiệu quả:** Mỗi bước xử lý phải đơn giản với thời gian thực thi hữu hạn. Trong thực tế điều này có nghĩa là phải thực thi trong khoảng thời gian có thể chấp nhận được.
- **Tính phổ dụng:** Thuật toán có thể áp dụng để xử lý một họ các bài toán (luôn đúng đắn trong nhiều tình huống khác nhau).

Các phương pháp biểu diễn (mô tả)

- **Biểu diễn bằng ngôn ngữ tự nhiên (Native Language)** : tiếng Việt, tiếng Anh,...
- **Biểu diễn bằng lưu đồ (Flow Chart)**
- **Biểu diễn bằng mã giả (Pseudo code)**: thường dựa vào cú pháp của một số ngôn ngữ lập trình thông dụng như Pascal, C/C++, ...

Mô tả thuật toán bằng ngôn ngữ tự nhiên

- Khi **biểu diễn thuật toán** theo ngôn ngữ tự nhiên, người ta sử dụng ngôn ngữ thường ngày để liệt kê các bước của thuật toán.
- Ví dụ: bài toán tìm nghiệm x của phương trình $ax + b = c$

1 – Nhập các giá trị a, b, c

2 – Nếu $a = 0$

Nếu $b = c$ thì thông báo phương trình có vô số nghiệm

nếu $b \neq c$ thì thông báo phương trình vô nghiệm

3 – Nếu $a \neq 0$

Thông báo: Phương trình có 1 nghiệm duy nhất

$$x = (c - b) / a$$

Mô tả thuật toán bằng mã giả

- Khi **biểu diễn thuật toán** bằng mã giả, người ta thường dựa vào cú pháp một của một số ngôn ngữ lập trình như Pascal, C/C++.
- Ví dụ: bài toán tìm nghiệm x của phương trình $ax + b = c$;

Nhập a, b, c

If $a = 0$ then

if $b = c$ then

write('có vô số nghiệm');

if $b \neq c$ then

write('phương trình vô nghiệm');

Else

begin

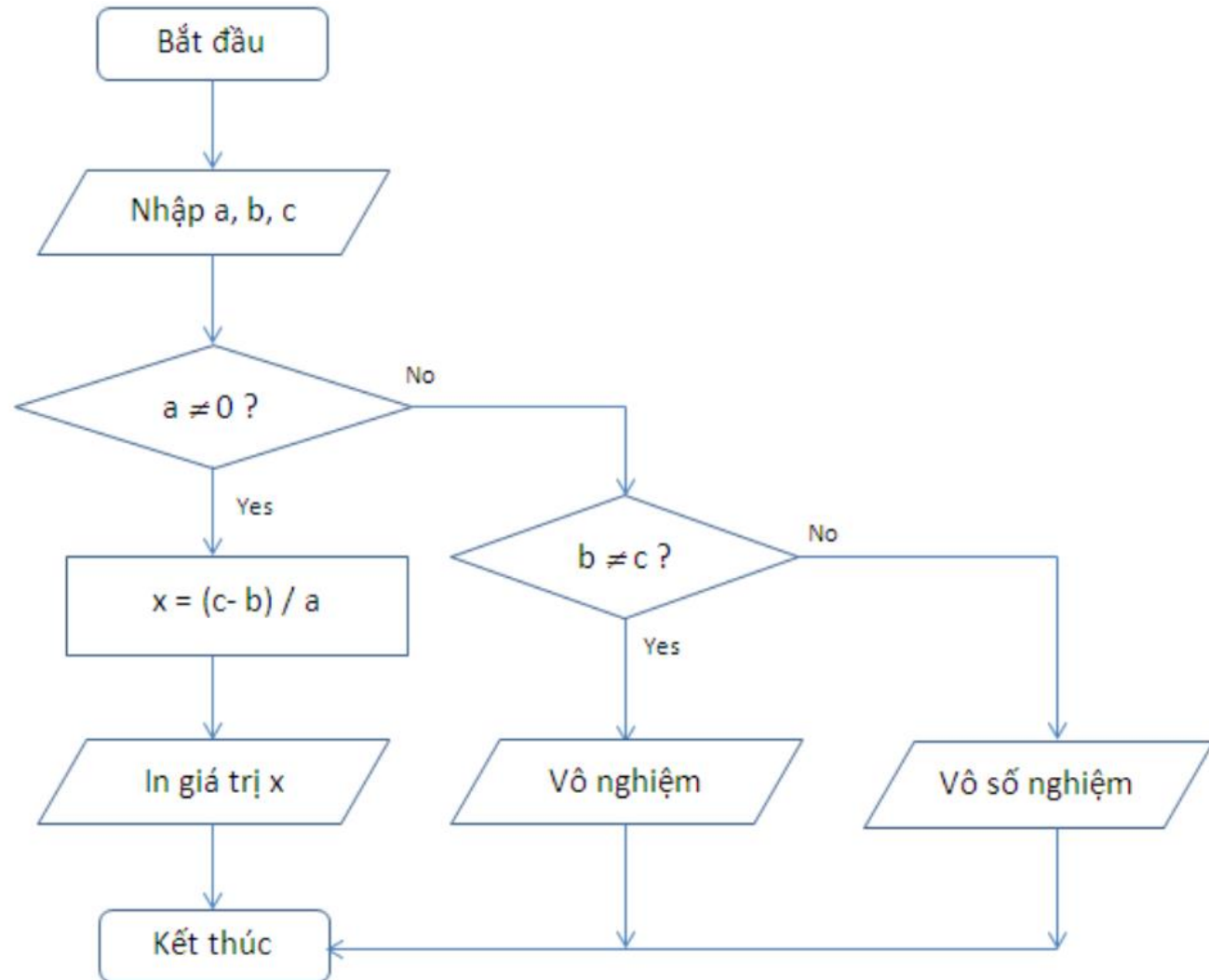
$x := (c - b) / a$;

write('Phương trình có 1 nghiệm duy nhất $x =$ ', x);

end



Mô tả thuật toán bằng lưu đồ

- Lưu đồ hay sơ đồ khối là một công cụ trực quan để diễn đạt các thuật toán. Biểu diễn thuật toán bằng lưu đồ sẽ giúp người đọc theo dõi được sự phân cấp các trường hợp và quá trình xử lý của thuật toán.
- Ví dụ: bài toán tìm nghiệm x của phương trình $ax + b = c$



Mô tả thuật toán bằng lưu đồ

- Lưu đồ thuật toán là công cụ dùng để biểu diễn thuật toán của chương trình việc mô tả quá trình nhập (Input), xuất dữ liệu (output) và các luồng xử lý (processing) đều được thể hiện dựa trên ký hiệu hình học.
- Ký hiệu sử dụng trên lưu đồ

Ký hiệu - Biểu tượng	Ý nghĩa sử dụng trong lưu đồ
	Biểu thị bắt đầu hoặc kết thúc chương trình
	Biểu thị hướng xử lý trong một quá trình / chương trình

Mô tả thuật toán bằng lưu đồ

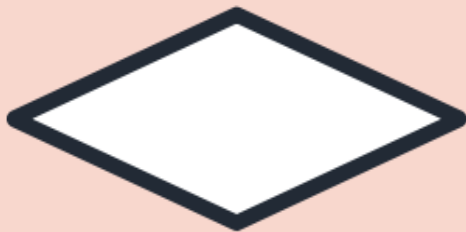
■ Ký hiệu sử dụng trên lưu đồ



Biểu thị thông tin vào hoặc thông tin ra (*Nhập hoặc xuất dữ liệu*) trong chương trình



Biểu thị một hoạt động (*Hay một quá trình*) trong thuật toán



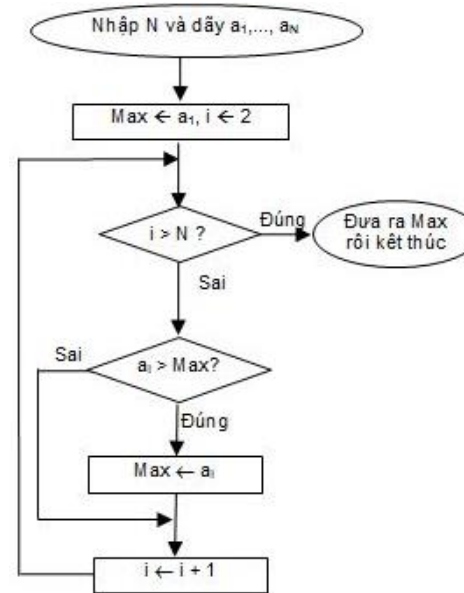
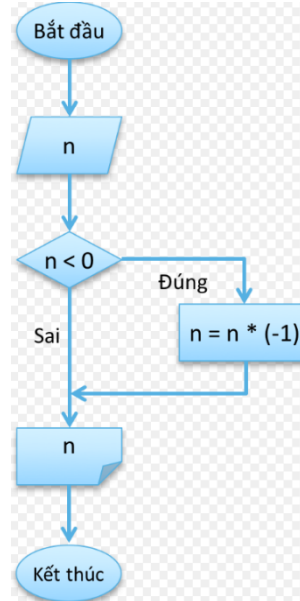
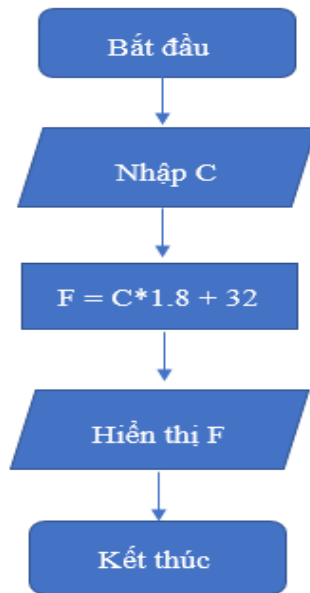
Biểu thị một quyết định khi đứng trước một vấn đề logic cần phải lựa chọn (*hoặc phân nhánh xử lý trong chương trình*).



Điểm nối trên lưu đồ (*Sử dụng khi lưu đồ có kích thước lớn, phức tạp*)

Các hình thức biểu diễn

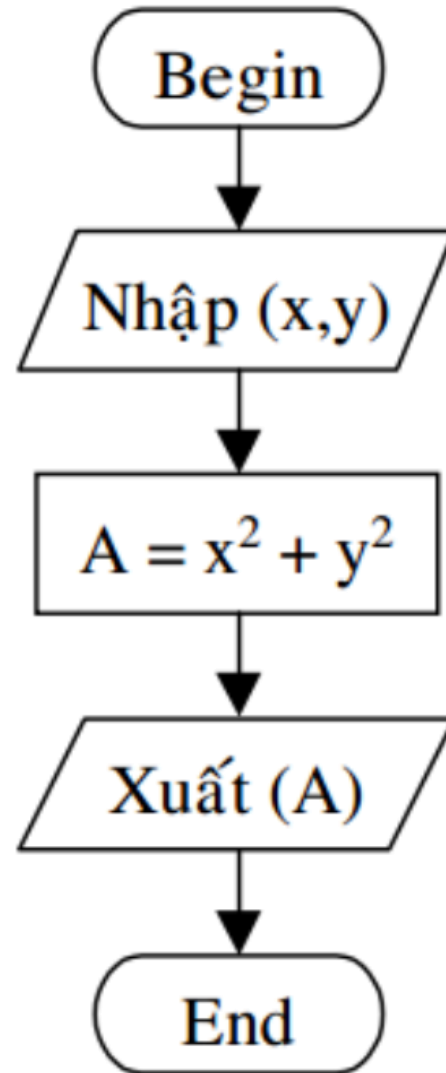
- Biểu diễn tuần tự
- Biểu phân nhánh (có chọn lựa hướng xử lý)
- Biểu diễn chu trình (Cấu trúc lặp)



Cấu trúc tuần tự

Yêu cầu:

- Nhập 2 số x, y. Sau đó tính tổng bình phương của 2 số và in kết quả ra màn hình

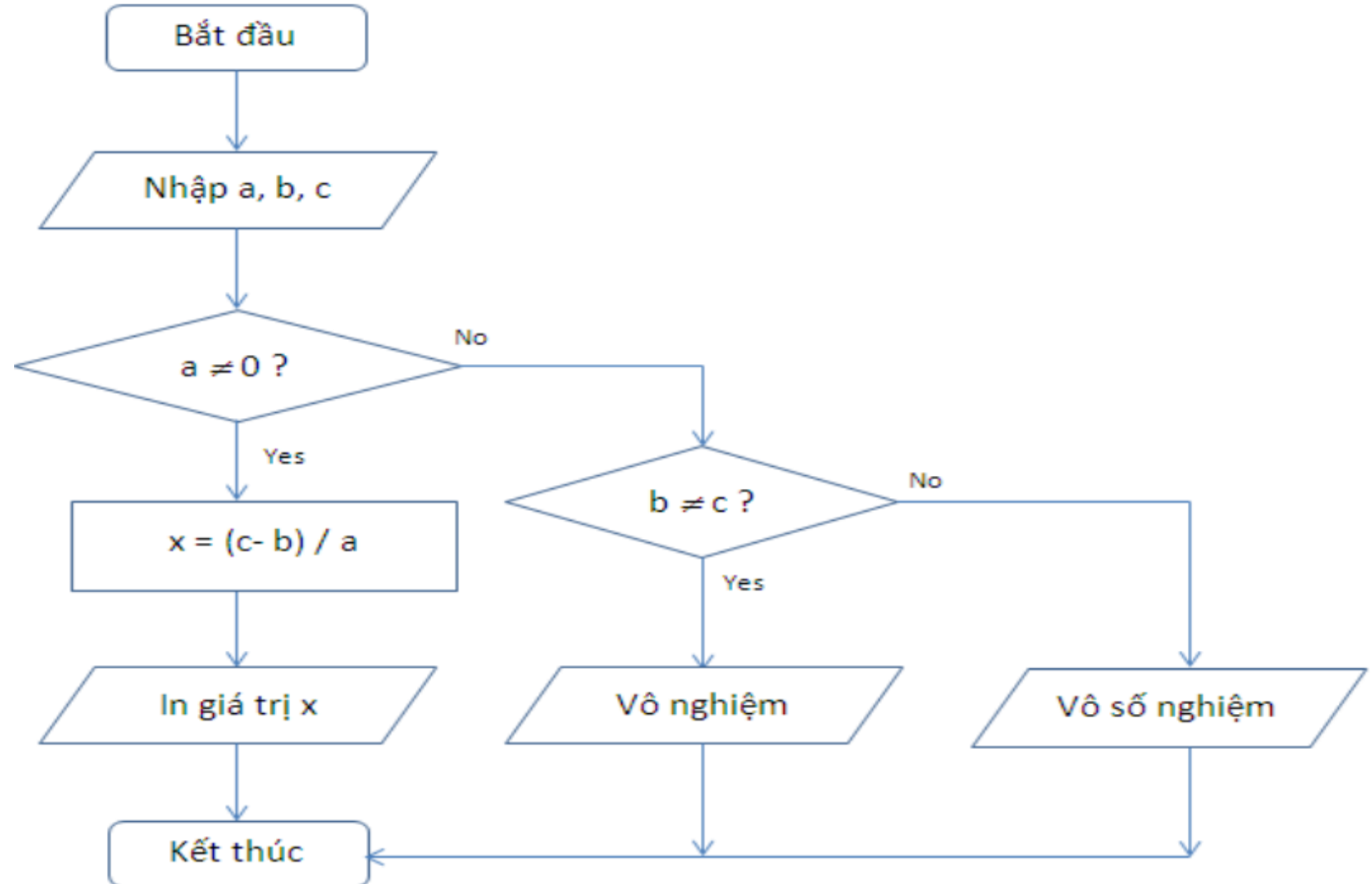


Cấu trúc phân nhánh

Yêu cầu:

- Giải phương trình

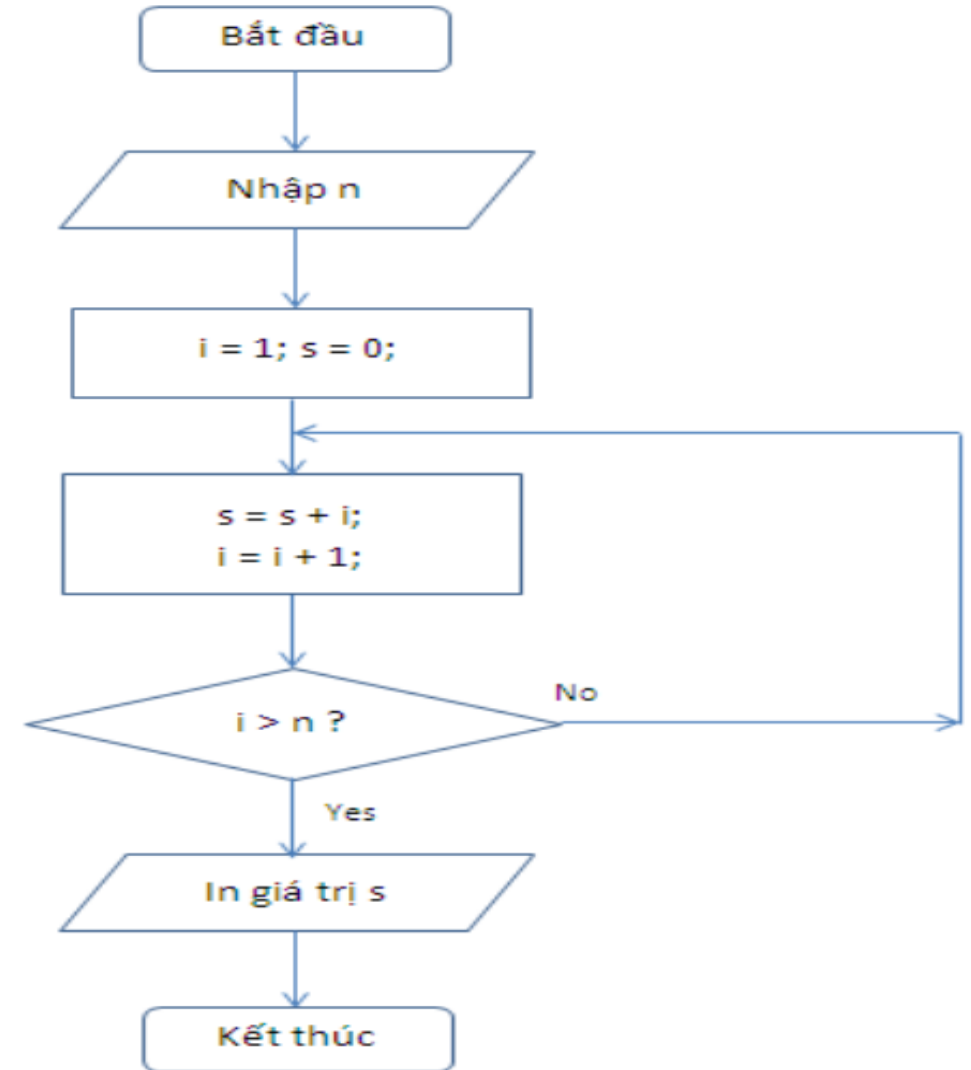
$$P(x): aX + b = c$$



Cấu trúc lặp (xử lý theo chu trình)

Yêu cầu:

- Tính tổng các số tự nhiên từ 1 đến n
(n được nhập từ bàn phím)



1. Cái khái niệm thuật toán, giải thuật. Các đặc tính của thuật toán. Các phương pháp biểu diễn thuật toán.
2. Lưu đồ, ký hiệu sử dụng trên lưu đồ. Các hình thức biểu diễn lưu đồ.
3. Ngôn ngữ lập trình thường được chia làm bao nhiêu loại.
4. Khái niệm về chương trình nguồn, chương trình mã máy, chương trình dịch và các cơ chế biên dịch, thông dịch.

Vẽ lưu đồ cho các bài toán sau:

1. Chia hai số nguyên dương (input: 2 số, output: thương số)
2. Tính tiền 1 loại sản phẩm trong siêu thị (input: số lượng, đơn giá – output: thành tiền)
3. Xếp loại học lực dựa trên điểm trung bình (input: điểm trung bình – output: xếp loại).
4. Tìm số lớn nhất trong dãy số (input: a_0, a_1, \dots, a_n , output: số lớn nhất)