TRƯỜNG ĐẠI HỌC THƯƠNG MẠI

THUONGMAI UNIVERSITY

BÀI GIẢNG ĐIỆN TỬ

HP: LẬP TRÌNH VỚI PYTHON

Số tín chỉ: 3

Bộ môn: Tin học

NỘI DUNG VÀ THỜI GIAN PHÂN BỔ

Chương	Số tiết	LT	TH
Chương 1. Tổng quan về cơ sở lập trình	6	**6**	0
Chương 2. Các thành phần cơ sở của Python	8	**8**	0
Chương 3. Hàm	12	**9**	6
Chương 4. Kiểu dữ liệu tổng hợp	9	**6**	6
Chương 5. Làm việc với tập tin	10	**7**	6
Tổng	45	**36**	18

Chương 1. Kỹ thuật lập trình cơ bản

1.1. Thuật toán

1.2. Sơ đồ khối

1.3. Chương trình và ngôn ngữ lập trình

1.4. Các phương pháp lập trình

1.5. Kỹ thuật thiết kế chương trình

1.1 Thuật toán

1.1.1 Khái niệm

1.1.2 Tính chất của thuật toán

1.1.3 Ví dụ

1.1.1 Khái niệm

Khái niệm 1: Thuật toán là một dãy ***hữu hạn*** các bước được sắp xếp theo ***một trật tự xác định***, mỗi bước mô tả chính xác các phép toán hoặc hành động cần thực hiện, để giải quyết một vấn đề.

Khái niệm 2: Thuật toán là một dãy hữu hạn các thao tác, sắp xếp theo một trật tự xác định, sau khi thực hiện, từ ***Input*** ta nhận được ***Output*** cần tìm.

1.1.1 Khái niệm

Cách biểu diễn thuật toán:

- Dùng “Ngôn ngữ tự nhiên”
- Dùng “Sơ đồ khối”
- Dùng “Giả ngôn ngữ”
- Ngôn ngữ lập trình

1.1.1 Khái niệm

- Cho một danh sách các số nguyên, hãy tìm số lớn nhất?

- **Hãy đề xuất thuật toán thực hiện công việc trên:**
  - Sử dụng giả ngôn ngữ (Giả ngôn ngữ -pseudocode – có cấu trúc tương tự như ngôn ngữ lập trình nhưng không thực sự là ngôn ngữ lập trình)

1.1.1 Khái niệm

- Thuật toán tìm số lớn nhất:

procedure max (a_1, a_2, …, a_n: integers) {

    max := a_1

    for i := 2 **to** n

        if max < a_i **then** *max* := a_i

}

1.1.1 Khái niệm

procedure max (a_1, a_2, …, a_n: integers)

max := a_1

for *i* := 2 **to** n

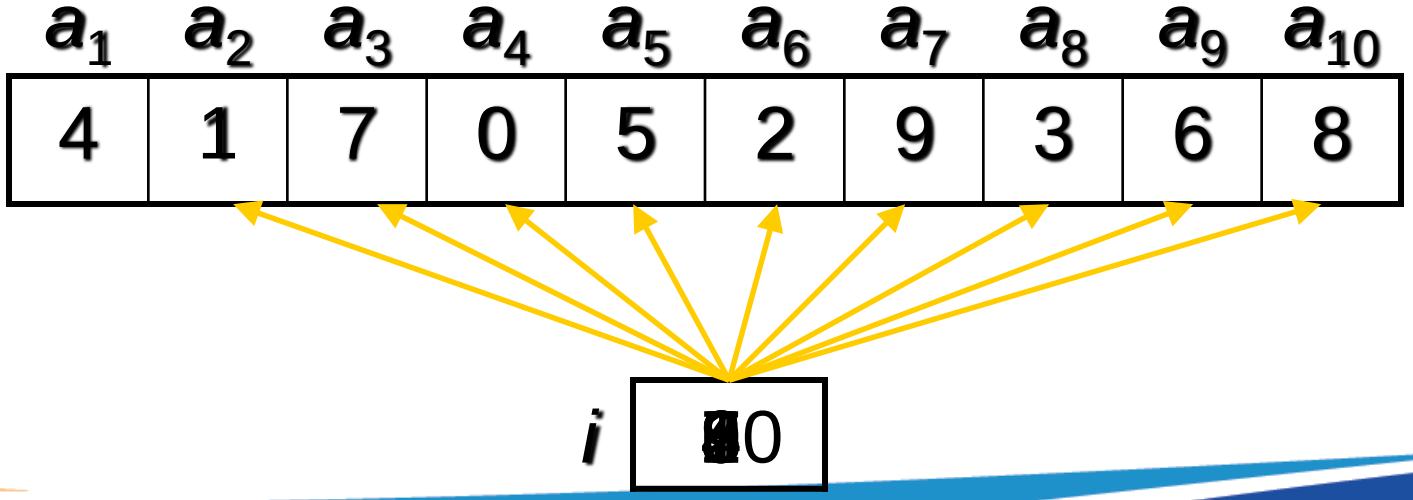
  if max < a_i **then** *max* := a_i

max := a_1

for *i* := 2 to n

  if max < a_i then *max* := a_i

max 4



1.1.2 Tính chất của thuật toán

- Tính vào-ra (input/output)
- Tính đơn định (xác định / đơn nghĩa)
- Tính đúng đắn
- Tính dừng (tính kết thúc / tính đóng)
- Tính phổ dụng
- Tính khả thi/hiệu quả

1.1.2 Tính chất của thuật toán

- Tính vào-ra (input/output):
  - Một thuật toán sẽ tồn tại các đầu vào và đầu ra.
  - **Đầu vào**: Là dữ liệu được đưa vào thuật toán trước khi thực hiện việc tính toán. Một thuật toán phải có đầu vào được định nghĩa cụ thể.
  - **Đầu ra**: Một thuật toán phải có một hoặc nhiều đầu ra được xác định rõ ràng là kết quả thực hiện của thuật toán khi kết thúc.

1.1.2 Tính chất của thuật toán

- **Tính đơn định** (xác định / đơn nghĩa):
  - Thuật toán bao gồm các bước, và tại mỗi bước thực hiện chúng ta cần đảm bảo tính rõ ràng và duy nhất cho luồng thực hiện dựa trên các điều kiện tất định.
  - Các thao tác trong thuật toán phải hết sức rõ ràng, không gây nên sự nhập nhằng, lẫn lộn, tùy tiện.
  - Khi thực hiện thuật toán, với cùng một dữ liệu đầu vào thì cho cùng một kết quả đầu ra.

1.1.2 Tính chất của thuật toán

- Tính đúng đắn:
  - Tính đúng đắn (tính chính xác) đảm bảo kết quả đưa ra của thuật toán đáp ứng được yêu cầu, giải quyết vấn đề của bài toán.
  - Đây cũng là đích đến cần đạt được khi thiết kế thuật toán.
  - Kết quả của thuật toán có thể kiểm tra được.

1.1.2 Tính chất của thuật toán

- Tính dừng (tính kết thúc / tính đóng):
  - Tính chất này có nghĩa là khi áp dụng thuật toán cho một bộ dữ liệu cụ thể thì sau hữu hạn bước thực hiện sẽ cho kết quả và kết thúc.
  - Tính dừng khi bị vi phạm sẽ dẫn đến các ứng dụng bị "treo" khi sử dụng thuật toán đó.

1.1.2 Tính chất của thuật toán

- Tính phổ dụng
  - Tính chất này có nghĩa là thuật toán không chỉ giải cho một bài toán mà giải cho một lớp các bài toán.
  - Thuật toán có thể làm việc trên các dữ liệu khác nhau với cùng định dạng.
  - Ví dụ: với thuật toán tìm số lớn nhất trong một dãy số nguyên: giá trị các số không cần định trước, số lượng phần tử cũng vậy.

1.1.2 Tính chất của thuật toán

- Tính khả thi/hiệu quả:
  - Kích thước phải đủ nhỏ.
  - Thuật toán phải được máy tính thực hiện trong thời gian cho phép.
  - Thuật toán phải dễ hiểu và dễ cài đặt.

1.1.3 Ví dụ

- ***Yêu cầu***: Xây dựng thuật toán để giải phương trình:

  $$ax + b=0$$

  Trong đó a,b là các số thực được nhập vào từ bàn phím
- ***Phân tích:***

  Input: a, b

  Output: kết luận về x

  Có 3 khả năng xảy ra:

  - Có một nghiệm duy nhất: x = -b/a khi a≠0.
  - Phương trình vô nghiệm khi a=0 và b≠0.
  - Phương trình có vô số nghiệm khi a=0 và b=0.

1.1.3 Ví dụ

- Thuật toán:

+ Bước 1: Nhập a, b.

+ Bước 2: Kiểm tra a≠0?

Nếu đúng chuyển sang bước 3.

Nếu sai chuyển sang bước 5.

+ Bước 3: Tính nghiệm x = -b/a.

+ Bước 4: In x rồi chuyển bước 8.

+ Bước 5: Kiểm tra b≠0?

Nếu đúng chuyển sang bước 6.

Nếu sai chuyển sang bước 7.

+ Bước 6: In “PTVN”, rồi chuyển bước sang 8.

+ Bước 7: In “PTCVSN”.

+ Bước 8: Kết thúc.

1.1.3 Ví dụ

Kiểm tra tính chất của thuật toán:

- Tính vào-ra
- Tính đơn định
- Tính đúng đắn
- Tính dừng
- Tính phổ dụng
- Tính khả thi/hiệu quả

- Thuật toán:

+ Bước 1: Nhập a, b.

+ Bước 2: Kiểm tra a≠0?

Nếu đúng chuyển sang bước 3.

Nếu sai chuyển sang bước 5.

+ Bước 3: Tính nghiệm x = -b/a.

+ Bước 4: In x rồi chuyển bước 8.

+ Bước 5: Kiểm tra b≠0?

Nếu đúng chuyển sang bước 6.

Nếu sai chuyển sang bước 7.

+ Bước 6: In “PTVN”, rồi chuyển bước sang 8.

+ Bước 7: In “PTCVSN”.

+ Bước 8: Kết thúc.

1.1.3 Ví dụ

Kiểm tra tính chất của thuật toán:

- Tính vào-ra
- Tính đơn định
- Tính đúng đắn
- Tính dừng
- Tính phổ dụng
- Tính khả thi/hiệu quả

- Thuật toán:
  - + Bước 1: Nhập a, b.
  - + Bước 2: Kiểm tra a≠0?
    - Nếu đúng chuyển sang bước 3.
    - Nếu sai chuyển sang bước 5.
  - + Bước 3: Tính nghiệm x = -b/a.
  - + Bước 4: In x rồi chuyển bước 8.
  - + Bước 5: Kiểm tra b≠0?
    - Nếu đúng chuyển sang bước 6.
    - Nếu sai chuyển sang bước 7.
  - + Bước 6: In “PTVN”, rồi chuyển bước sang 8.
  - + Bước 7: In “PTCVSN”.
  - + Bước 8: Kết thúc.

1.1.3 Ví dụ

Kiểm tra tính chất của thuật toán:

- Tính vào-ra
- Tính đơn định
- Tính đúng đắn
- Tính dừng
- Tính phổ dụng
- Tính khả thi/hiệu quả

- Thuật toán:
  - + Bước 1: Nhập a, b.
  - + Bước 2: Kiểm tra a≠0?
    - Nếu đúng chuyển sang bước 3.
    - Nếu sai chuyển sang bước 5.
  - + Bước 3: Tính nghiệm x = -b/a.
  - + Bước 4: In x rồi chuyển bước 8.
  - + Bước 5: Kiểm tra b≠0?
    - Nếu đúng chuyển sang bước 6.
    - Nếu sai chuyển sang bước 7.
  - + Bước 6: In “PTVN”, rồi chuyển bước sang 8.
  - + Bước 7: In “PTCVSN”.
  - + Bước 8: Kết thúc.

1.1.3 Ví dụ

Kiểm tra tính chất của thuật toán:

- Tính vào-ra
- Tính đơn định
- Tính đúng đắn
- Tính dừng
- Tính phổ dụng
- Tính khả thi/hiệu quả

- Thuật toán:
  - + Bước 1: Nhập a, b.
  - + Bước 2: Kiểm tra a≠0?
    - Nếu đúng chuyển sang bước 3.
    - Nếu sai chuyển sang bước 5.
  - + Bước 3: Tính nghiệm x = -b/a.
  - + Bước 4: In x rồi chuyển bước 8.
  - + Bước 5: Kiểm tra b≠0?
    - Nếu đúng chuyển sang bước 6.
    - Nếu sai chuyển sang bước 7.
  - + Bước 6: In “PTVN”, rồi chuyển bước sang 8.
  - + Bước 7: In “PTCVSN”.
  - + Bước 8: Kết thúc.

1.1.3 Ví dụ

Kiểm tra tính chất của thuật toán:

- Tính vào-ra
- Tính đơn định
- Tính đúng đắn
- Tính dừng
- Tính phổ dụng
- Tính khả thi/hiệu quả

- Thuật toán:
  - + Bước 1: Nhập a, b.
  - + Bước 2: Kiểm tra a≠0?
    - Nếu đúng chuyển sang bước 3.
    - Nếu sai chuyển sang bước 5.
  - + Bước 3: Tính nghiệm x = -b/a.
  - + Bước 4: In x rồi chuyển bước 8.
  - + Bước 5: Kiểm tra b≠0?
    - Nếu đúng chuyển sang bước 6.
    - Nếu sai chuyển sang bước 7.
  - + Bước 6: In “PTVN”, rồi chuyển bước sang 8.
  - + Bước 7: In “PTCVSN”.
  - + Bước 8: Kết thúc.

1.1.3 Ví dụ

Kiểm tra tính chất của thuật toán:

- Tính vào-ra
- Tính đơn định
- Tính đúng đắn
- Tính dừng
- Tính phổ dụng
- Tính khả thi/hiệu quả

- Thuật toán:
  - + Bước 1: Nhập a, b.
  - + Bước 2: Kiểm tra a≠0?
    - Nếu đúng chuyển sang bước 3.
    - Nếu sai chuyển sang bước 5.
  - + Bước 3: Tính nghiệm x = -b/a.
  - + Bước 4: In x rồi chuyển bước 8.
  - + Bước 5: Kiểm tra b≠0?
    - Nếu đúng chuyển sang bước 6.
    - Nếu sai chuyển sang bước 7.
  - + Bước 6: In “PTVN”, rồi chuyển bước sang 8.
  - + Bước 7: In “PTCVSN”.
  - + Bước 8: Kết thúc.

1.2 Sơ đồ khối

1.2.1 Khái niệm

1.2.2 Các cấu trúc điều khiển

1.2.3 Ví dụ

1.2.1 Khái niệm

Khái niệm 1: Sơ đồ khối là tập hợp các ký hiệu và quy tắc dùng để biểu diễn thuật toán.

Khái niệm 2: Sơ đồ khối là cách thể hiện thuật toán bằng các hình khối hình học được kết nối với nhau bằng đường đi có hướng.

=> ***Biểu diễn thuật toán một cách trực quan***.

1.2.1 Khái niệm

Các thành phần:

–Hình thoi: thể hiện thao tác so sánh.

–Hình chữ nhật: thể hiện các phép tính toán.

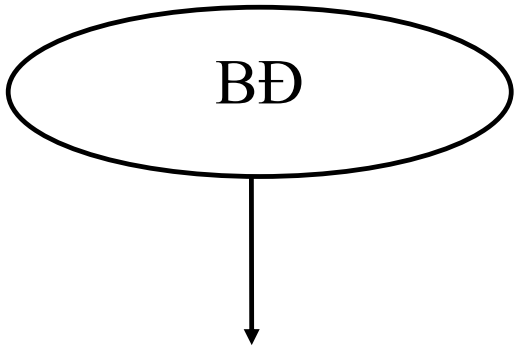
–Hình bình hành: thể hiện thao tác nhập xuất dữ liệu.

–Hình oval: thể hiện sự bắt đầu/kết thúc của thuật toán.

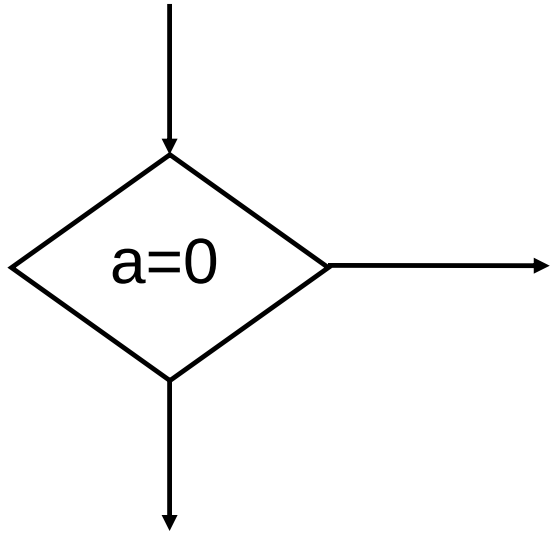
–Các mũi tên: thể hiện trình tự thực hiện các thao tác.

1.2.1 Khái niệm

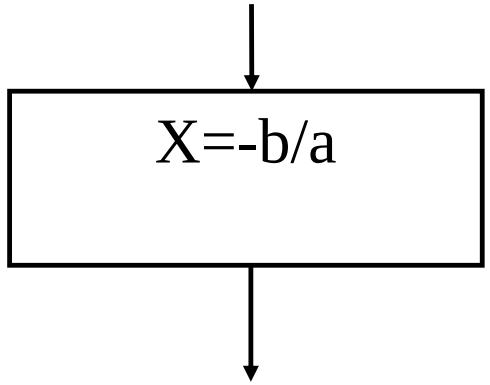
- ***Các khối***



Khối bắt đầu



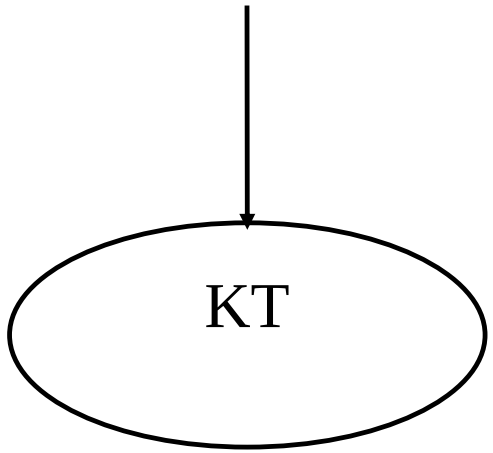
Khối kiểm tra điều kiện



Khối tính toán

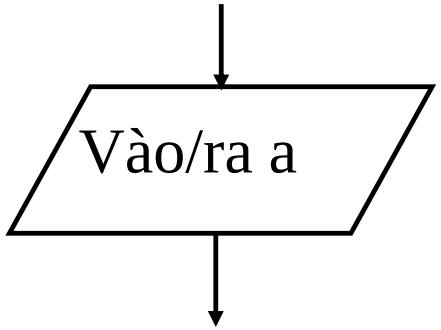
1.2.1 Khái niệm

- ***Các khối***



Khối kết thúc

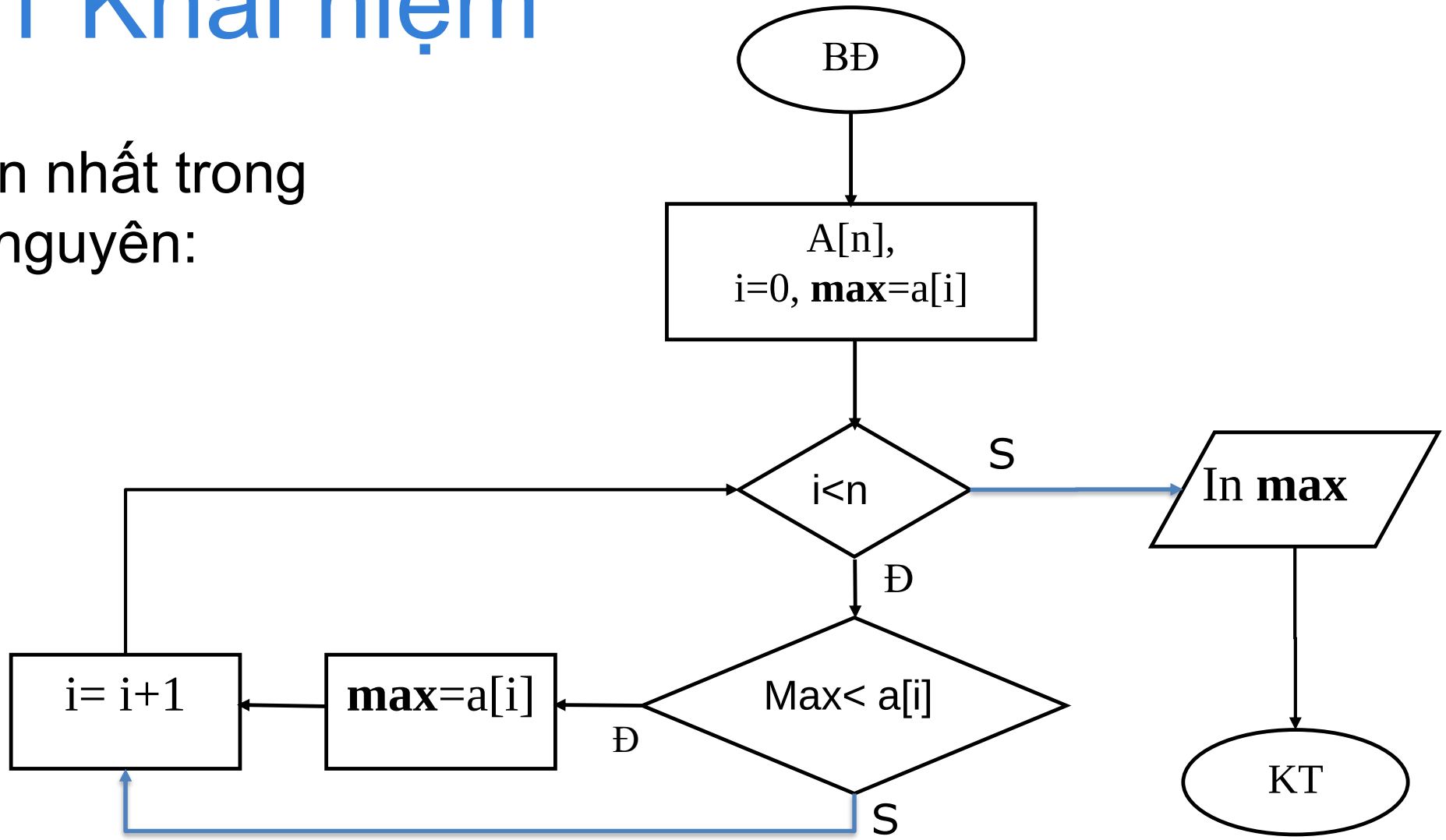
Hướng thực hiện.



Khối Vào/Ra

1.2.1 Khái niệm

- Tìm số lớn nhất trong mảng số nguyên:

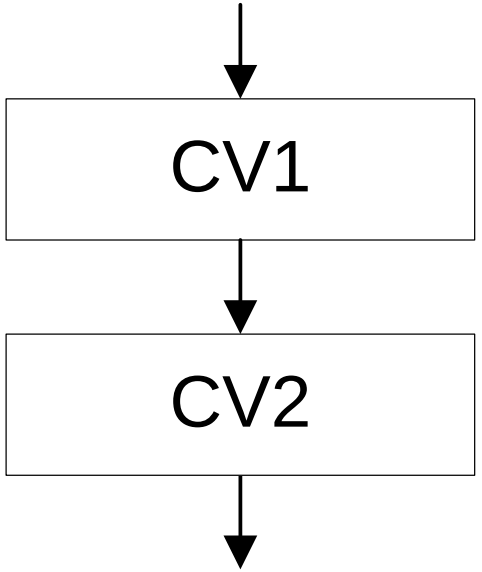


1.2.2 Các cấu trúc điều khiển

- ***Sơ đồ khối dạng tuần tự***: Là sơ đồ khối thể hiện thuật toán gồm n khối mà khi thực hiện thuật toán với một bộ dữ liệu cụ thể sẽ lần lượt thực hiện từ khối đầu tiên đến khối cuối cùng theo thứ tự viết của nó.
- ***Sơ đồ khối dạng rẽ nhánh***: Là sơ đồ khối thể hiện thuật toán gồm n nhánh mà khi thực hiện thuật toán với một bộ dữ liệu cụ thể sẽ thực hiện một trong n nhánh mà thôi.
- ***Sơ đồ khối dạng chu trình***: Là sơ đồ khối thể hiện thuật toán gồm n khối mà khi thực hiện thuật toán với một bộ dữ liệu cụ thể sẽ có một số khối lặp đi lặp lại nhiều lần phụ thuộc vào một điều kiện nào đó vẫn thỏa mãn.

1.2 Sơ đồ khối

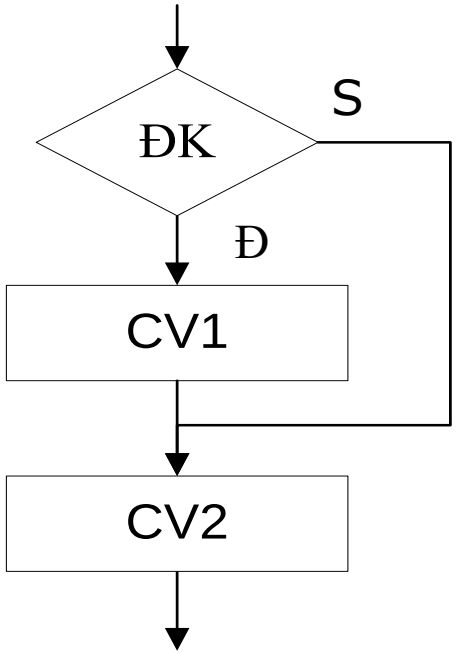
a) Cấu trúc tuần tự:



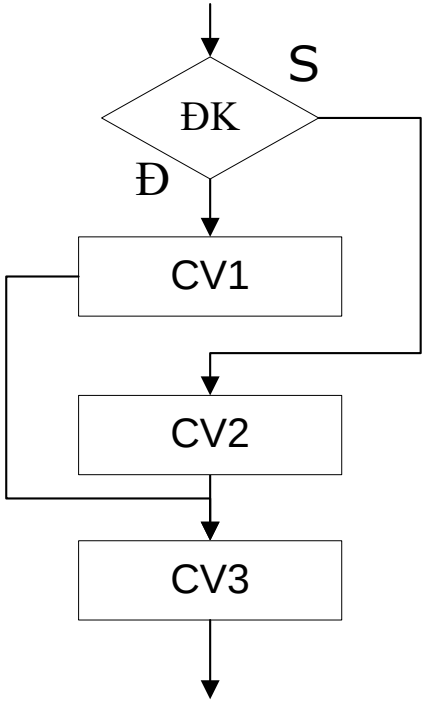
b) Cấu trúc rẽ nhánh

- Có 4 dạng cơ bản:

Dạng thứ nhất:

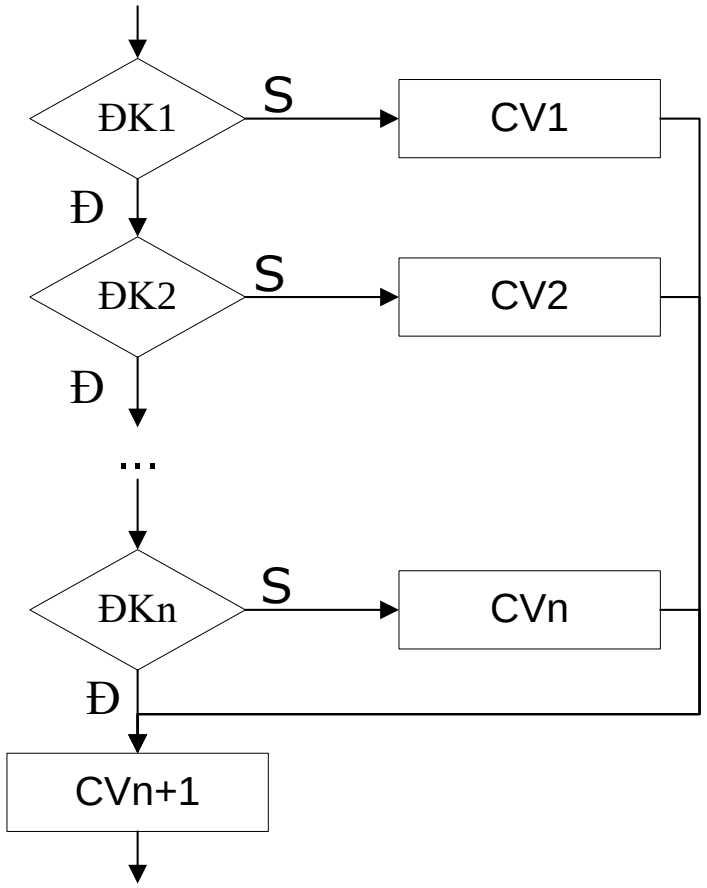


Dạng thứ hai:

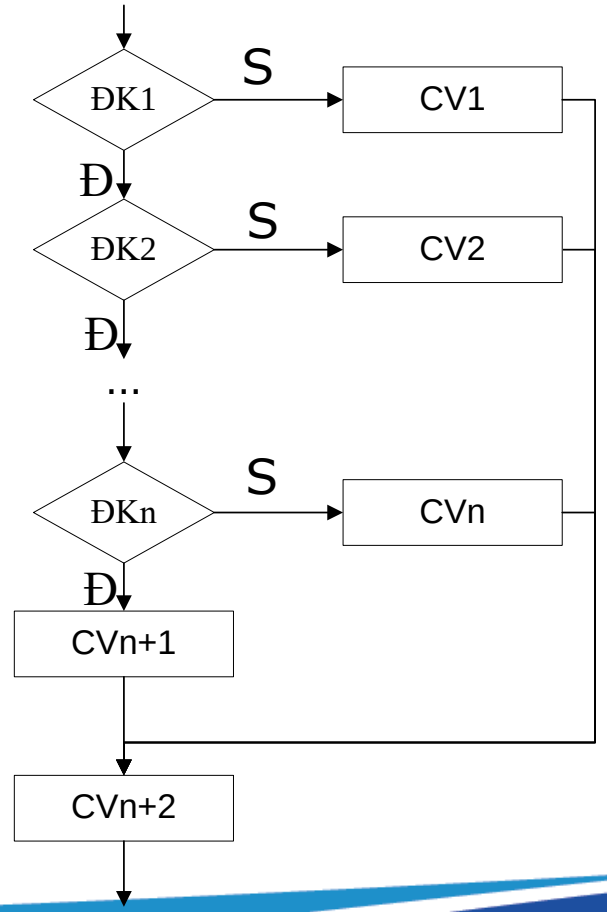


b) Cấu trúc rẽ nhánh

Dạng thứ ba:



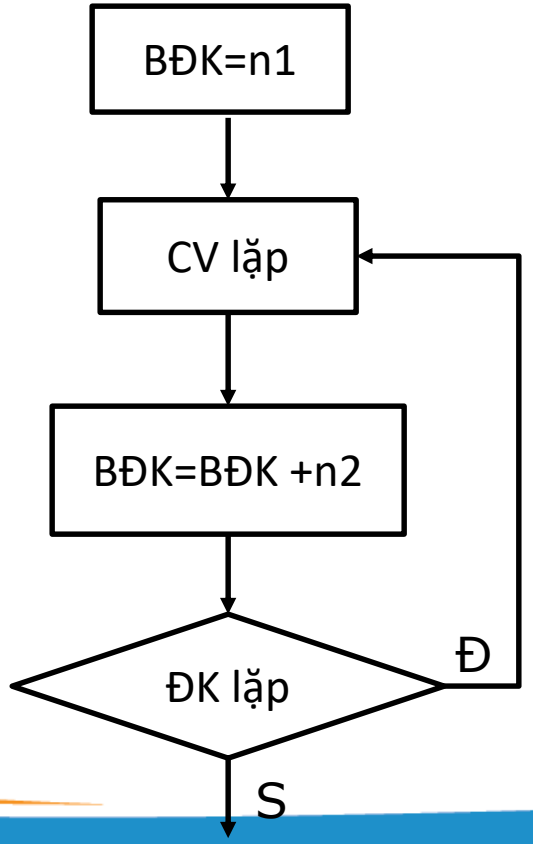
Dạng thứ tư:



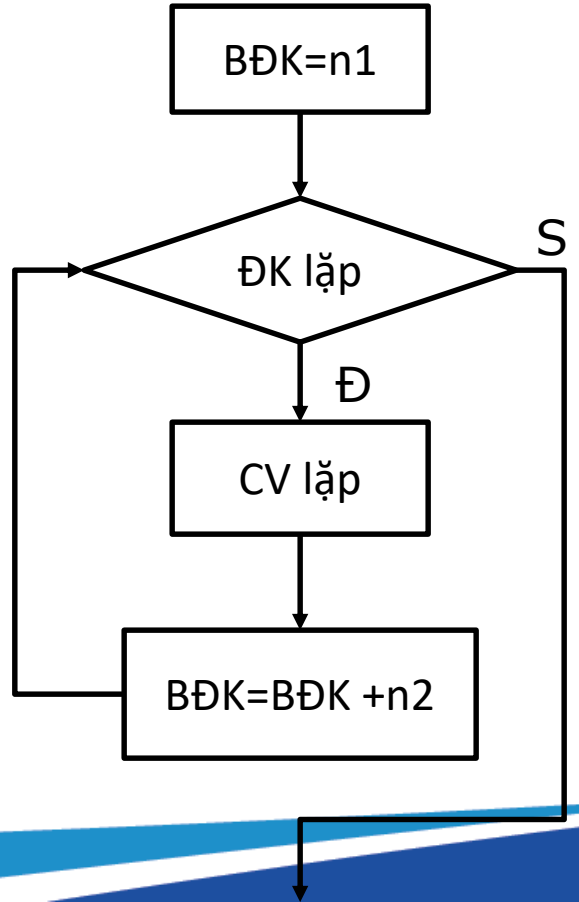
c) Cấu trúc lặp

- Có 2 dạng cơ bản.

Dạng thứ nhất:



Dạng thứ hai:



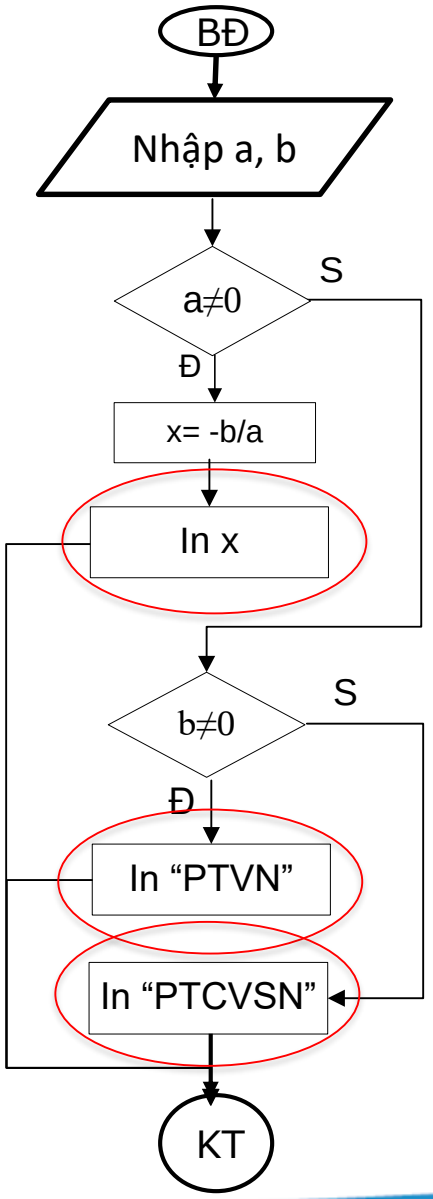
1.2.3 Ví dụ

Xây dựng sơ đồ khối mô tả thuật toán giải phương trình:

ax + b=0

Trong đó a,b là các số thực được nhập vào từ bàn phím

1.2.3 Ví dụ



1.2.3 Ví dụ

Sinh viên vẽ sơ đồ khối thể hiện thuật toán làm việc với dãy số nguyên:

1. Tìm giá trị lớn nhất của dãy số nguyên
2. Tính tổng các phần tử chia hết cho 3 của dãy số nguyên

1.3 Chương trình và ngôn ngữ lập trình

1.3.1 Chương trình

1.3.2 Ngôn ngữ lập trình

1.3.3 Trình tự giải bài toán trên MTĐT

1.3.4 Đánh giá chương trình MTĐT

1.3.1 Chương trình

Khái niệm: Chương trình là một tập hợp các ***lệnh*** để thể hiện một thuật toán giải quyết một bài toán hay một nhiệm vụ nào đó.

Trong đó:

Lệnh (lệnh máy): là một chỉ thị để máy tính có thể thực hiện một cách tự động.

=> Các chương trình giúp người dùng sử dụng, khai thác MTĐT!

1.3.2 Ngôn ngữ lập trình

Khái niệm: Ngôn ngữ lập trình là ngôn ngữ để viết chương trình.

Phân loại: (phụ thuộc vào kiến trúc và hoạt động của máy tính)

- Ngôn ngữ máy (ngôn ngữ bậc thấp)
- Ngôn ngữ hợp ngữ
- Ngôn ngữ bậc cao

1.3.2 Ngôn ngữ lập trình

- Ngôn ngữ lập trình:
  - C/ C++/ C#
  - Java
  - PHP, NodeJS
  - JavaScript
  - Python

1.3.3 Trình tự giải bài toán trên MTĐT

Bước 1: Xác định bài toán

Bước 2: Tìm cấu trúc biểu diễn bài toán

Bước 3: Chọn phương pháp giải

Bước 4: Lập trình giải bài toán

Bước 5: Thử nghiệm chương trình

Bước 6: Tối ưu chương trình (bảo trì)

1.3.4 Đánh giá chương trình MTĐT

- ***Mục tiêu***: xây dựng chương trình tốt, có chất lượng
- ***Tiêu chuẩn đánh giá***:
  - Tính đúng đắn:
  - Tính bền vững:
  - Tính sử dụng lại
  - Tính thích nghi (mở rộng)
  - Tính tương thích

1.3.4 Đánh giá chương trình MTĐT

- ***Mục tiêu***: xây dựng chương trình tốt, có chất lượng
- ***Tiêu chuẩn đánh giá***:
  - Tính hiệu quả
  - Tính dễ chuyển đổi
  - Tính an toàn
  - Thân thiện với người sử dụng

1.4 Các phương pháp lập trình

- 1.4.1 Lập trình hướng thủ tục
- 1.4.2 Lập trình hướng đối tượng

1.4.1 Lập trình hướng thủ tục

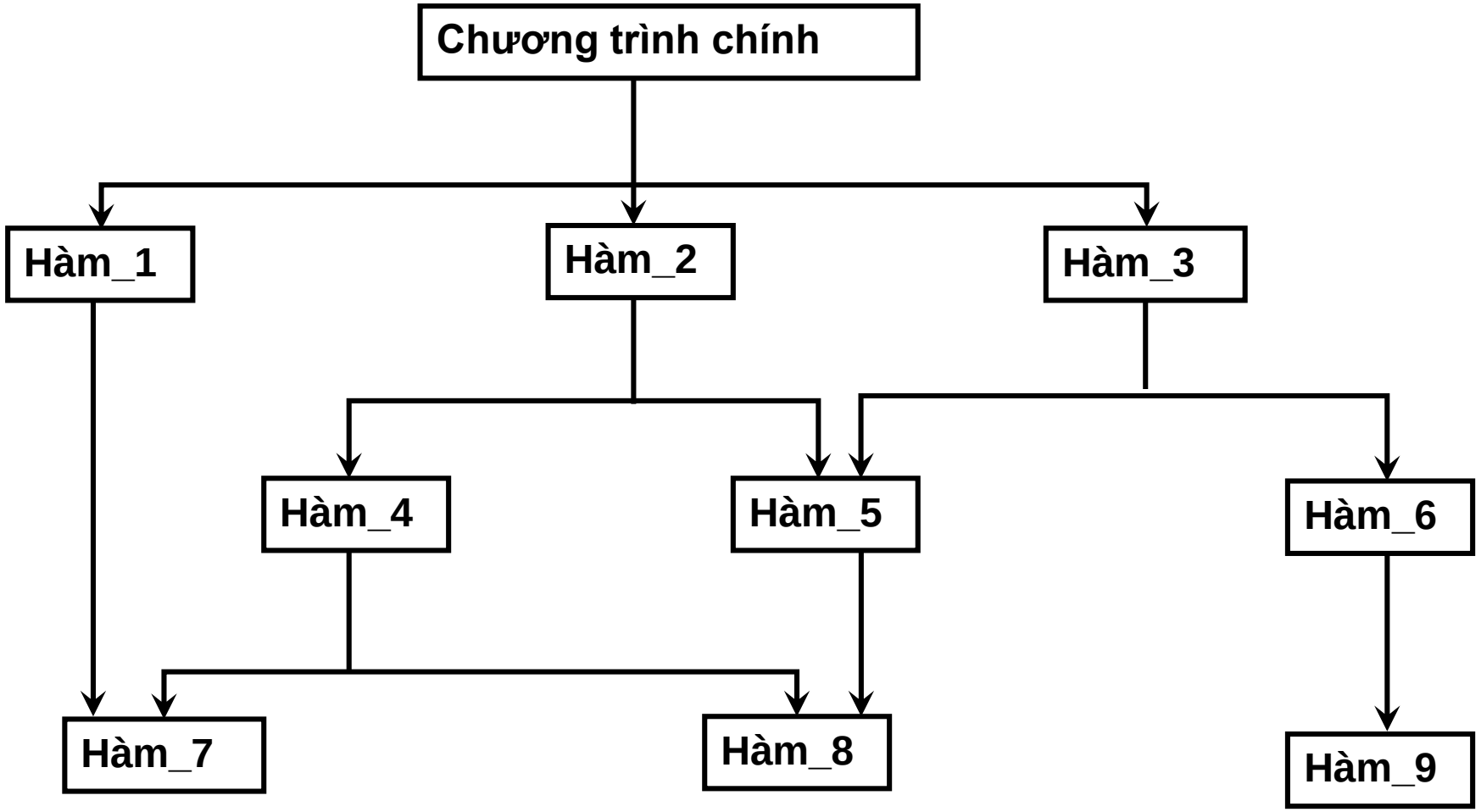
(procedural/functional programming)

- ***Tư tưởng***: Chia nhỏ bài toán cho đến khi không còn chia nhỏ được nữa. Từ đó xây dựng cấu trúc dữ liệu và các hàm, thủ tục để giải các bài toán con.

  Trong đó: hàm/thủ tục là một đơn vị chương trình độc lập dùng để thực hiện một phần việc nào đó như: nhập số liệu, in kết quả hoặc thực hiện một số tính toán. Hàm có thể có biến và tham số của nó.

- Ví dụ về ngôn ngữ: C, pascal, Foxpro, …

1.4.1 Lập trình hướng thủ tục



1.4.1 Lập trình hướng thủ tục

- ***Ví dụ***: Bài toán quản lý dãy số
- Các bài toán con:
  - Nhập, xuất dãy
  - Tìm kiếm phần tử thỏa mãn điều kiện
  - Tính tổng các phần tử
  - Sắp xếp dãy số

1.4.1 Lập trình hướng thủ tục

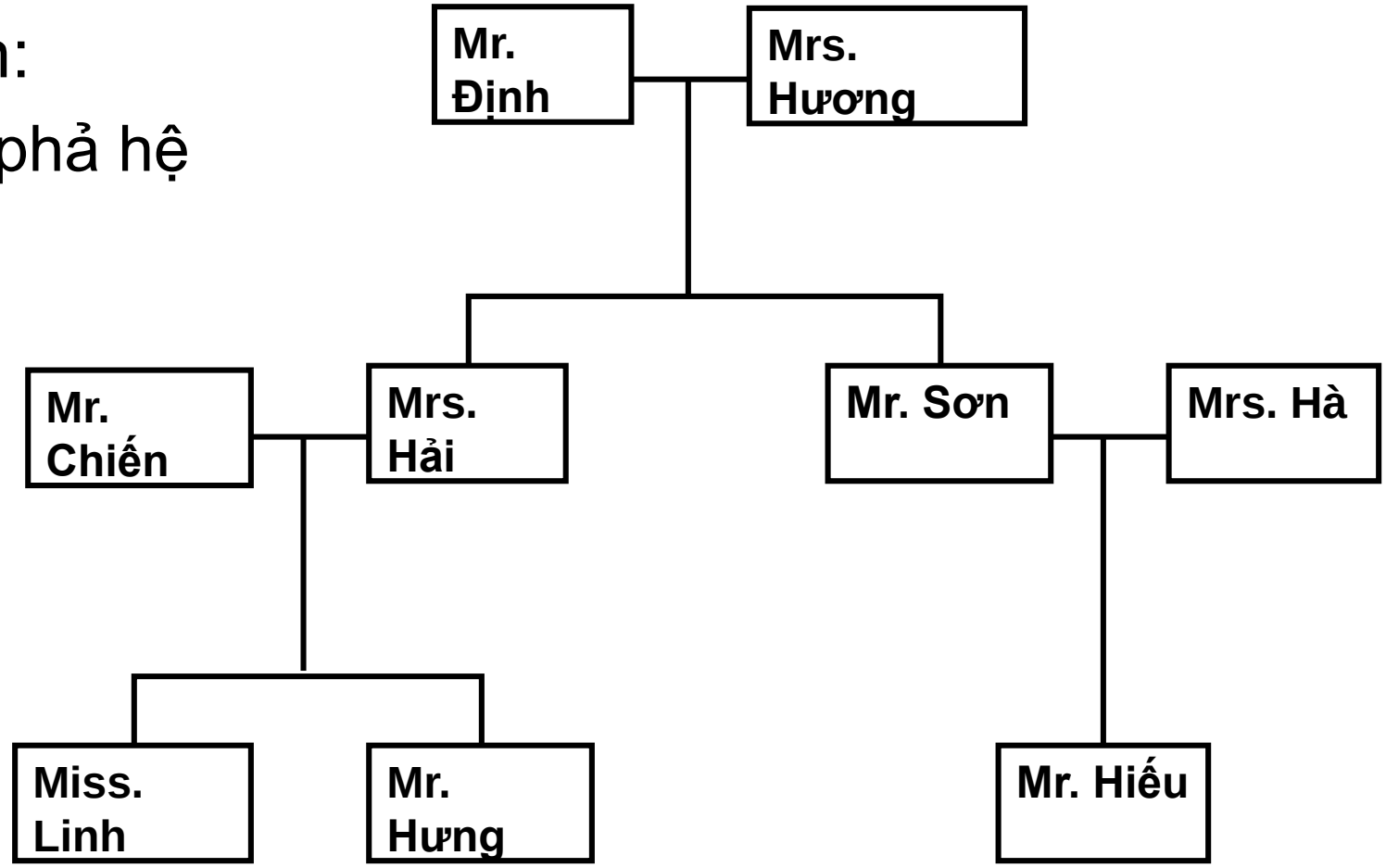
- ***Đặc điểm:***
  - Tập trung vào công việc cần thực hiện (thuật toán)
  - Chương trình lớn được chia thành các hàm nhỏ hơn.
  - Phần lớn các hàm sử dụng dữ liệu chung
  - Các hàm truyền thông tin cho nhau thông qua cơ chế truyền tham số.
  - Dữ liệu trong hệ thống được chuyển động từ hàm này sang hàm khác.
  - ***Đóng gói chức năng***
  - ***Chương trình được thiết kế theo cách tiếp cận từ trên xuống (top-down)***

1.4.1 Lập trình hướng thủ tục

- *Hạn chế:*
  - Có hàm có thể truy cập và thay đổi dữ liệu chung → khó kiểm soát (nhất là đối với chương trình lớn, phức tạp)
  - Nếu thay đổi cấu trúc dữ liệu dùng chung cho một số hàm thì phải thay đổi các hàm liên quan dữ liệu đó.
  - Mô hình được xây dựng không mô tả được đầy đủ và trung thực các hệ thống trong thực tế.

Ví dụ: Bài toán quan hệ gia đình

- Xây dựng chương trình:
  - Quản lý dữ liệu theo phả hệ



Ví dụ: Bài toán quan hệ gia đình

Vấn đề: xây dựng cấu trúc dữ liệu để thể hiện cây quan hệ: thành thạo con trỏ, cập nhập thông tin trên cây quan hệ

Yêu cầu bài toán: Chia nhỏ thành các yêu cầu như

- Thêm mới thông tin 1 người trong gia đình.
- Cập nhật thông tin.
- Xác định quan hệ giữa các thành viên.
- .v.v.

Ví dụ: Bài toán quan hệ gia đình

Cách thức giải quyết một công việc nhỏ:

- Ví dụ: Xác định quan hệ giữa các thành viên: "Hưng và Hiếu có quan hệ như thế nào?"

→xây dựng giải thuật xác định mối quan hệ:

+ Tham số hóa bài toán,

+ xây dựng phép duyệt cây.

+ xác định tên gọi các mối quan hệ ở Việt nam rất phong phú →vét cạn các mối quan hệ

Ví dụ: Bài toán quan hệ gia đình

- ***Câu hỏi***: Muốn quản lý nhiều gia đình và các gia đình có quan hệ thông gia với nhau?

  → cấu trúc biểu diễn dữ liệu thay đổi

  → thuật toán thay đổi

1.4.2 Lập trình hướng đối tượng

- ***Tư tưởng***: Phân tích bài toán thành các thực thể được gọi là các đối tượng → từ đó xây dựng chương trình dựa trên sự tương tác giữa các đối tượng đó.

  Trong đó: đối tượng là sự kết hợp giữa dữ liệu và hàm (phương thức) thao tác trên dữ liệu đó.

- ***Ví dụ về ngôn ngữ:***
  - C++, Java, Smalltalk, Python, .v.v.

1.4.2 Lập trình hướng đối tượng

- ***Đặc điểm***
  - Đặt trọng tâm vào đối tượng, tập trung vào dữ liệu thay vì các phương thức.
  - Chương trình được chia thành các đối tượng.
  - Các đối tượng tác động và trao đổi thông tin cho nhau thông qua các hàm với cơ chế thông báo.
  - Đóng gói chức năng và dữ liệu (không thể truy cập trực tiếp vào các thành phần dữ liệu của đối tượng mà phải thông qua các phương thức của đối tượng đó)
  - Chương trình được thiết kế theo cách tiếp cận từ dưới lên (bottom - up)

1.4.2 Lập trình hướng đối tượng

- ***Ưu điểm:***
  - Dữ liệu và các hàm mới có thể dễ dàng bổ sung vào đối tượng khi cần thiết → dễ nâng cấp thành hệ thống lớn hơn.
  - Cơ chế đóng gói dữ liệu →Dữ liệu được bao bọc và không cho phép các hàm ngoại lai truy cập tự do → chương trình an toàn
  - Mô hình được xây dựng gần với hệ thống thực tế .
  - Thông qua nguyên lý kế thừa → loại bỏ đoạn chương trình lặp lại khi khai báo lớp và mở rộng khả năng sử dụng các lớp → ngắn gọn, tiết kiệm thời gian
  - Cách thiết kế đặt trọng tâm vào dữ liệu → xây dựng mô hình chi tiết và cài đặt dễ hơn

Ví dụ: bài toán quan hệ gia đình

- ***Giải quyết***: Xem xét dưới góc độ quản lý tập các đối tượng con người
- Đối tượng "con người" được xác định bao gồm:
  - **Thuộc tính**: tên, cha, mẹ, anh em, con cái, vợ chồng,…
  - **Phương thức**: cưới, sinh con, là anh, là ông nội,…
- Các đối tượng khi biểu diễn trong phần mềm được gọi là "đối tượng phần mềm". Các đối tượng này là kết quả của quá trình "trừu tượng hóa" các đối tượng thực tế.

Ví dụ: bài toán quan hệ gia đình

- Các chức năng của hệ thống được xây dựng từ tương tác giữa các đối tượng.
- Ví dụ: Tạo cây quan hệ bằng các sự kiện:
  - Chiến.Cưới(Hải)
  - Hải.Sinh con(gái,Linh)
  - Hải.Sinh con(trai, Hưng)

→ Không cần quan tâm tạo cấu trúc cây quan hệ mà vẫn trả lời được các câu hỏi liên quan đến cây phả hệ.

Con người
Tên ?
Cha ?
Mẹ ?
Anh em ?
Con cái ?
Vợ chồng?
Sinh con
Cưới
Là anh
Là ông nội
….

Ví dụ: bài toán quan hệ gia đình

- Yêu cầu:
  - Xác định các thuộc tính phù hợp trong phạm vi bài toán.
  - Xây dựng các phương thức
  - Xây dựng cơ chế truyền thông điệp.
- Cách tiếp cận hướng đối tượng cho phép xây dựng phần mềm như là một mô hình hóa của hệ thống thực tế.

Con người
Tên ? Cha ? Mẹ ? Anh em ? Con cái ? Vợ chồng?
Sinh con Cưới Là anh Là ông nội ….

1.5 Các kỹ thuật thiết kế chương trình

1.5.1 Kỹ thuật thiết kế trên xuống

1.5.2 Kỹ thuật chương trình con

1.5.3 Kỹ thuật đệ quy

1.5 Các kỹ thuật thiết kế chương trình

- Thiết kế chương trình?

  - Thiết kế chi tiết cấu trúc bên trong phần mềm.
  - Thiết kế tính năng cho các mô đun chương trình.
  - Trình tự xử lý bên trong cho các mô đun.
  - .v.v.

- ***Thiết kế chương trình là một công việc phức tạp và cần cách tiếp cận phù hợp.***

1.5 Các kỹ thuật thiết kế chương trình

- Thiết kế chương trình?

  - Thiết kế chi tiết cấu trúc bên trong phần mềm.
  - Thiết kế tính năng cho các mô đun chương trình.
  - Trình tự xử lý bên trong cho các mô đun.
  - .v.v.

- ***Thiết kế chương trình là một công việc phức tạp và cần cách tiếp cận phù hợp.***

1.5.1 Kỹ thuật thiết kế trên xuống

- **Tư tưởng**: giải quyết bài toán theo chiến lược chia để trị và tinh chỉnh từng bước
  - Bắt đầu từ việc phân tích tổng quát toàn bộ vấn đề => Đề ra những công việc chủ yếu => Sau đó mới đi dần vào giải quyết các phần cụ thể.
  - Từ các bước sau, các công việc cần giải quyết sẽ được chi tiết hóa dần dần tương ứng với các công việc nhỏ hơn mà ta gọi đó là các bước tinh chỉnh.

1.5.2 Kỹ thuật chương trình con

- **Tư tưởng**: Một chương trình con thường được viết mã sao cho nó có thể được chạy (hay được gọi) nhiều lần từ nhiều nơi trong chương trình lớn hơn, thậm chí có thể được gọi bởi chính nó
  - Kỹ thuật chương trình con => giảm đáng kể kích thước đồng thời nâng cao tính dễ đọc và độ tin cậy của chương trình.
  - Các chương trình con hay sử dụng còn được tập trung thành các thư viện chương trình.

1.5.2 Kỹ thuật Đệ quy

- **Tư tưởng**: Một bài toán P được gọi là có tính chất đệ quy khi lời giải của nó có thể đưa về lời giải của bài toán P′ nhỏ hơn nó và có dạng giống nó, đồng thời lời giải của P′ không cần dùng tới P.
- Lời giải cho những bài toán như vậy được gọi là giải thuật đệ quy.
- Tương tự như tư tưởng của phương pháp quy nạp toán học.

1.5.2 Kỹ thuật Đệ quy

- Bản chất của giải thuật đệ quy là phân tách một bài toán lớn thành những bài toán nhỏ hơn và dễ giải hơn, sau đó tìm cách kết hợp lời giải của các bài toán nhỏ lại thành lời giải cho bài toán lớn ban đầu.
- Trong đó bài toán nhỏ hơn có tính chất như bài toán ban đầu, nhưng có giới hạn của tham số nhỏ hơn.
- Quá trình thu nhỏ bài toán là một quá trình lặp đi lặp lại cho đến khi một điều kiện dừng được thỏa mãn.
- Bài toán tìm giai thừa của $n! = n * (n-1)!$ chính là một ví dụ cơ bản nhất cho các bài toán có tính chất đệ quy.

1.5.2 Kỹ thuật Đệ quy

- Thông thường một chương trình con đệ qui cơ bản gồm hai phần:
  - **Phần cơ sở:** Phần này chứa các tác vụ của hàm với một số giá trị cụ thể ban đầu của tham số;
  - **Phần đệ qui:** Phần này định nghĩa các tác vụ cần được thực hiện cho giá trị hiện thời của các tham số bằng các tác vụ đã được định nghĩa trước đây với kích thước tham số nhỏ hơn.

Tổng kết Chương 1

- Nội dung đã học:
  - 1.1. Thuật toán
  - 1.2. Sơ đồ khối
  - 1.3. Chương trình và ngôn ngữ lập trình
  - 1.4. Các phương pháp lập trình
  - 1.5. Kỹ thuật thiết kế chương trình
- Chúc các em có thời gian học tập vui vẻ!