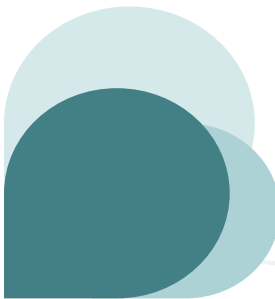




## PHẦN IV: THIẾT KẾ VÀ LẬP TRÌNH DESIGN AND PROGRAMMING

- 
- I. Thiết kế hệ thống
    1. Khái niệm
    2. Thiết kế cấu trúc hóa
    3. Quy trình thiết kế
    4. Các phương pháp thiết kế hệ thống
  - II. Thiết kế chương trình
  - III. Lập trình

1



### 1. Thiết kế hệ thống là gì?

- Là thiết kế cấu hình phần cứng và cấu trúc phần mềm (gồm cả chức năng và dữ liệu) để có được hệ thống thỏa mãn các yêu cầu đề ra.
- Có thể xem như Thiết kế cấu trúc (WHAT), chứ không phải là Thiết kế Logic (HOW).
- Phương pháp thiết kế cấu trúc hóa (Structured Design) của Constantine
- Phương pháp thiết kế tổng hợp (Composite Design) của Myers.

2



## 2. Thiết kế cấu trúc hóa

- Bắt nguồn từ modularity, top-down design, structured programming.
- Còn xem như phương pháp thiết kế hướng luồng dữ liệu (Data flow-oriented design).
- Quy trình 6 bước:
  - Tạo kiểu luồng thông tin;
  - Chỉ ra biên của luồng;
  - Ánh xạ DFD sang cấu trúc chương trình;
  - Xác định phân cấp điều khiển;
  - Tinh lọc cấu trúc;
  - Chọn mô tả kiến trúc.

3



## Đặc trưng của thiết kế cấu trúc hóa

- Dễ thích ứng với mô hình vòng đời thác nước do tính thân thiện cao.
- Thiết kế theo tiến trình, không hợp với thiết kế xử lý theo lô (batch system).
- Dùng phân chia - kết hợp để giải quyết tính phức tạp của hệ thống.
- Topdown trong phân chia module.
- Kỹ thuật lập trình hiệu quả.

4

## 2.1. Module

1. Khái niệm
2. Thiết kế cấu trúc hóa

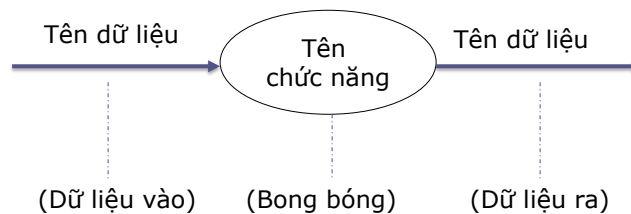
- Dãy các lệnh nhằm thực hiện chức năng (function) nào đó.
- Có thể được biên dịch độc lập
- Module đã được dịch có thể được module khác gọi tới.
- Giao diện giữa các module thông qua các biến tham số (arguments) .
- So sánh với các NNLT!

5

### a. Lưu đồ bong bóng (Bubble chart)

1. Khái niệm
2. Thiết kế cấu trúc hóa
  - 2.1. Mô đun
  - 2.2. Lưu đồ bong bóng và cấu trúc phân cấp

- Biểu thị luồng xử lý dữ liệu
- Ký pháp



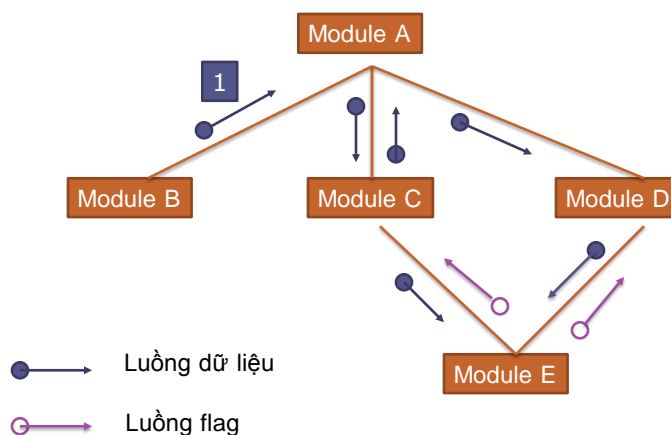
6

## b. Cấu trúc phân cấp (Hierarchical structured chart)

- Là phân cấp biểu thị quan hệ phụ thuộc giữa các module và giao diện (interface) giữa chúng
- Các quy ước:
  - Không liên quan đến trình tự gọi các module, nhưng ngầm định là từ trái qua phải
  - Mỗi module xuất hiện trong cấu trúc 1 lần, có thể được gọi nhiều lần
  - Quan hệ trên dưới: không cần nêu số lần gọi
  - Tên module biểu thị chức năng ("làm gì"), đặt tên sao cho các module ở phía dưới tổng hợp lại sẽ biểu thị đủ chức năng của module tương ứng phía trên.
  - Biến số (arguments) biểu thị giao diện giữa các module, biến số ở các module gọi/bị gọi có thể khác nhau.
  - Mũi tên xanh biểu thị dữ liệu, tím biểu thị flag.
  - Chiều của mũi tên là hướng truyền tham số.

7

## Cấu trúc phân cấp



8



### 3. Quy trình thiết kế hệ thống

- Phân chia mô hình phân tích ra các hệ con
- Tìm ra sự tương tranh (concurrency) trong hệ thống
- Phân bổ các hệ con cho các bộ xử lý hoặc các nhiệm vụ (tasks)
- Phát triển thiết kế giao diện
- Chọn chiến lược cài đặt quản trị dữ liệu
- Tìm ra nguồn tài nguyên chung và cơ chế điều khiển truy nhập chung.
- Thiết kế cơ chế điều khiển thích hợp cho hệ thống, kể cả quản lý nhiệm vụ.
- Xem xét các điều kiện biên được xử lý như thế nào.
- Xét duyệt và xem xét các thỏa hiệp (trade-offs).

9



### Các điểm lưu ý khi thiết kế hệ thống

1. Có thể trích được luồng dữ liệu từ hệ thống: đó là phần nội dung đặc tả yêu cầu và giao diện.
  2. Xem xét tối ưu tài nguyên kiến trúc lên hệ thống rồi quyết định kiến trúc.
  3. Theo quá trình biến đổi dữ liệu, hãy xem những chức năng được kiến trúc như thế nào?
  4. Từ kiến trúc các chức năng, hãy xem xét và chỉnh lại, từ đó chuyển sang kiến trúc chương trình và thiết kế chi tiết.
  5. Quyết định các đơn vị chương trình theo các chức năng của hệ phần mềm có dựa theo luồng dữ liệu và phân chia ra các thành phần.
  6. Khi cấu trúc chương trình lớn quá, phải phân chia nhỏ hơn thành các module.
  7. Xem xét dữ liệu vào-ra và các tệp dùng chung của chương trình. Truy cập tệp tối ưu.
  8. Hãy nghĩ xem để có được những thiết kế trên thì nên dùng phương pháp luận và những kỹ thuật gì ?
- Thiết kế hệ thống
    - Thiết kế hệ thống phần cứng [(1), (2)]
    - Thiết kế hệ thống phần mềm [(3)-(7)]
  - Thiết kế hệ thống phần mềm
    - Thiết kế tệp (file design) [(7)]
    - Thiết kế chức năng hệ thống [(3)-(6)]

10



## 4. Các phương pháp thiết kế hệ thống

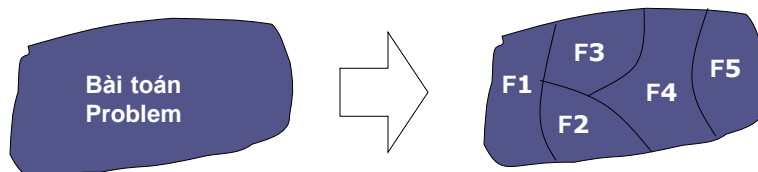
- Thiết kế cấu trúc:
  - Phương pháp phân chia STS (Source/Transform/Sink: Nguồn/Biến đổi/Hấp thụ)
  - Phương pháp phân chia TR (Transaction)
- Minh họa phân chia chức năng theo bong bóng của DFD (biểu đồ luồng dữ liệu)

11



### 4.1. Phương pháp phân chia STS (Source/Transform/Sink)

- 1) Chia đối tượng “bài toán” thành các chức năng thành phần

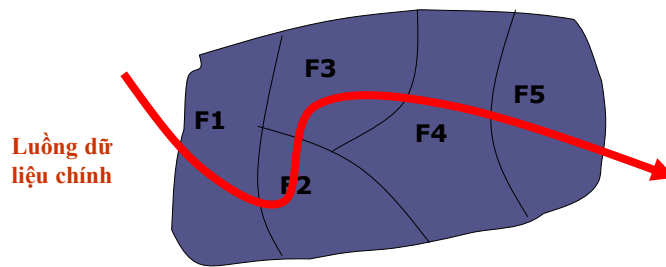


13



## Quyết định luồng dữ liệu chính

- 2) Tìm ra luồng dữ liệu chính đi qua các chức năng: từ đầu vào (Input) tới đầu ra (Output)



14



## Quyết định bong bóng và dữ liệu

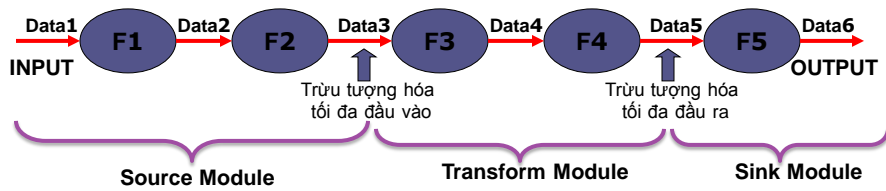
- 3) Theo luồng dữ liệu chính: thay từng chức năng bởi bong bóng và làm rõ dữ liệu giữa các bong bóng



15

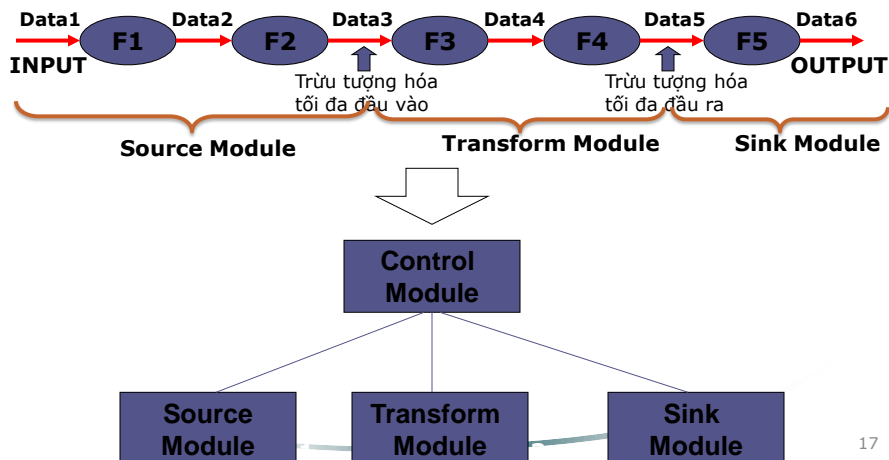
## Từ sơ đồ bong bóng sang sơ đồ phân cấp

- 4) Xác định vị trí trừu tượng hóa tối đa đầu vào và đầu ra



16

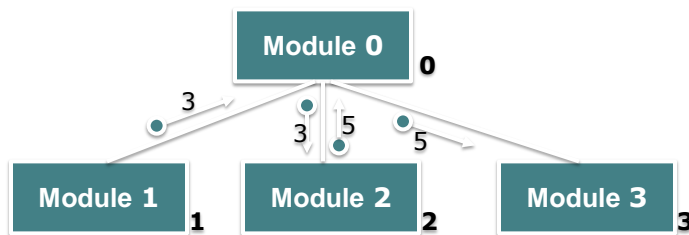
- 5) Chuyển sang sơ đồ phân cấp



17



- 6) Xác định các tham số giữa các module dựa theo quan hệ phụ thuộc



18

- 7) Với từng module (Source, Transform, Sink) lại áp dụng cách phân chia STS lặp lại các bước từ 1) đến 6). Đôi khi có trường hợp không chia thành 3 mô đun nhỏ mà thành 2 hoặc 1.
- 8) Tiếp tục chia đến mức cấu trúc logic khi module tương ứng với thuật toán đã biết thì dừng. Tổng hợp lại ta được cấu trúc phân cấp: mỗi nút là 1 module với số nhánh phía dưới không nhiều hơn 3.

19

## 4.2. Phương pháp phân chia TR (Transaction)

- Khi không tồn tại luồng dữ liệu chính, mà dữ liệu vào có đặc thù khác nhau như những nguồn khác nhau xem như các Giao dịch khác nhau.
- Mỗi giao dịch ứng với 1 module xử lý nó
- Phân chia module có thể: theo kinh nghiệm; theo tính độc lập module; theo số bước tối đa trong 1 module (ví dụ  $< 50$ ) và theo chuẩn.

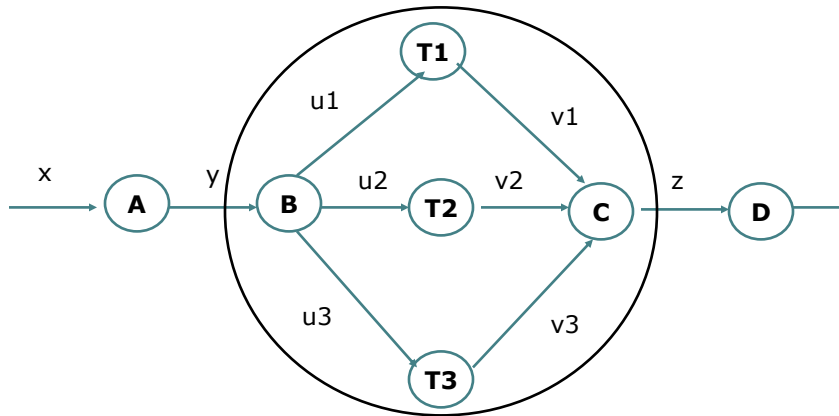
20

## Các bước thực hiện

- Xác định trung tâm giao dịch gồm 1 số chức năng phân loại và các chức năng tham gia vào quá trình xử lý:
  - Vào: 1 số chức năng truyền dẫn thông tin từ nguồn và 1 số chức năng sơ chế.
  - Ra: Một số chức năng dẫn thông tin ra từ các tình huống xử lý.
- Vẽ 2 mức cao nhất của lưu đồ cấu trúc:
  - Một mô đun mức cao nhất (đỉnh): mức 1
  - 1 cho vào, 1 mô đun xử lý cho mỗi trường hợp và 1 mô đun ra: mức 2.

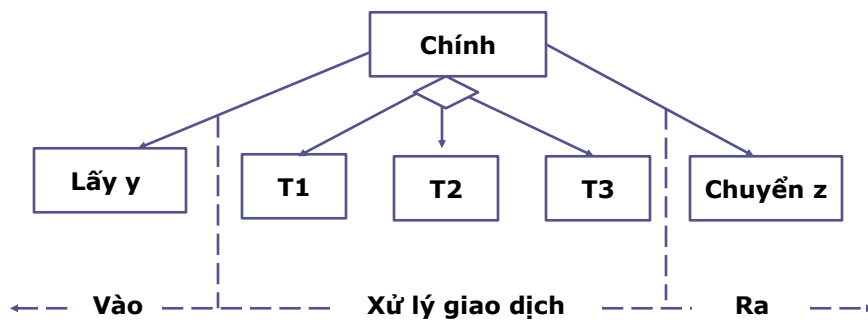
21

## Thí dụ



Biểu đồ chức năng có 1 trung tâm giao dịch

## Thí dụ



Lưu đồ cấu trúc mô đun mức đỉnh



### 4.3. Phân tích cấu trúc hóa

- Xác định luồng dữ liệu
- Luồng tuyến tính thì theo phân chia STS
- Luồng phân nhánh thì theo phân chia TR

24



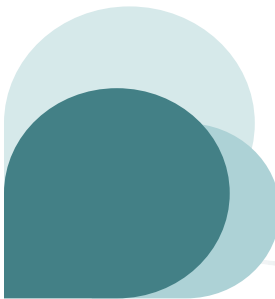
### 4.4. Chuẩn phân chia module

- Tính độc lập: Độ kết hợp (coupling) và Độ bền vững (strength)
- 5 tiêu chuẩn của Myers
  - Decomposability
  - Composability
  - Understandability
  - Continuity
  - Protection

25



## PHẦN IV: THIẾT KẾ VÀ LẬP TRÌNH DESIGN AND PROGRAMMING

- 
- I. Thiết kế hệ thống
  - II. Thiết kế chương trình
    1. Khái niệm
    2. Phương pháp thiết kế chương trình
    3. Công cụ thiết kế
  - III. Lập trình

26



### 1. Khái niệm

- Thiết kế chương trình là thiết kế chi tiết cấu trúc bên trong của phần mềm: thiết kế tính năng từng module và giao diện tương ứng.
- Cấu trúc ngoài của phần mềm: thiết kế hệ thống.
- Trình tự xử lý bên trong: Thuật toán (giải thuật, Algorithm); Logic.

27



## 2. Phương pháp thiết kế chương trình

- Nguyên tắc
  - Không có trạng thái mờ (fuzzy), để đảm bảo thiết kế cấu trúc trong đúng đắn.
  - Ngôn ngữ lập trình phù hợp.
  - Triển khai đúng đắn đặc tả chức năng các module và chương trình nhờ phương pháp luận thiết kế chi tiết.
  - Dùng quy trình thiết kế để chuẩn hóa từng bước.
- Kỹ thuật thiết kế mô hình hệ phần mềm
  - Hướng tiến trình (process) : Kỹ thuật thiết kế cấu trúc điều khiển.
  - Hướng cấu trúc dữ liệu (data): Kỹ thuật thiết kế cấu trúc dữ liệu.
  - Hướng sự vật / đối tượng (object): Kỹ thuật thiết kế hướng đối tượng.

28



### 2.1. Lập trình cấu trúc hóa

- Khái niệm cơ bản: tuần tự, nhánh (chọn), lặp; cấu trúc mở rộng, tiền xử lý, hậu xử lý.
- Những điểm lợi khi thiết kế thuật toán :
  - Tính độc lập của module: chỉ quan tâm vào-ra
  - Làm cho chương trình dễ hiểu
  - Dễ theo dõi chương trình thực hiện
  - Hệ phức tạp sẽ dễ hiểu nhờ tiếp cận phân cấp

29



## Loại bỏ GOTO

- GOTO dùng để làm gì?
  - Cho phép thực hiện các bước nhảy đến một nhãn nhất định
- Tại sao cần loại bỏ GOTO ?
  - Phá vỡ tính cấu trúc của lập trình cấu trúc hóa
- Phương pháp loại bỏ GOTO
- Có thể loại bỏ GOTO trong mọi trường hợp?
- Thế nào là “kỹ năng lập trình cấu trúc”

30



## VD chương trình Spaghetti

```

Start:  Get (Time-on, Time-off, Time, Setting, Temp, Switch)
        if Switch = off goto off
        if Switch = on goto on
        goto Cntrl
off: if Heating-status = on goto Sw-off
    goto loop
on:  if Heating-status = off goto Sw-on
    goto loop
Cntrl: if Time = Time-on goto on
      if Time = Time-off goto off
      if Time < Time-on goto Start
      if Time > Time-off goto Start
      if Temp > Setting then goto off
      if Temp < Setting then goto on
Sw-off: Heating-status := off
        goto Switch
Sw-on:  Heating-status := on
Switch: Switch-heating
loop:   goto Start
  
```

31



## CT được cấu trúc hoá

```

loop
  -- The Get statement finds values for the given variables from the system's
  -- environment.
  Get (Time-on, Time-off, Time, Setting, Temp, Switch) ;
  case Switch of
    when On => if Heating-status = off then
      Switch-heating ; Heating-status := on ;
    end if ;
    when Off => if Heating-status = on then
      Switch-heating ; Heating-status := off ;
    end if ;
    when Controlled =>
      if Time >= Time-on and Time <= Time-off then
        if Temp > Setting and Heating-status = on then
          Switch-heating; Heating-status = off ;
        elsif Temp < Setting and Heating-status = off then
          Switch-heating; Heating-status := on ;
        end if ;
      end if ;
    end case ;
  end loop ;

```

32



## Lưu ý khi thiết kế chương trình

- Phụ thuộc vào kỹ năng và kinh nghiệm của người thiết kế.
- Cần chuẩn hóa tài liệu đặc tả thiết kế chi tiết.
- Khi thiết kế cấu trúc điều khiển của giải thuật, vì theo các quy ước cấu trúc hóa nên đôi khi tính sáng tạo của người thiết kế bị hạn chế, bó buộc theo khuôn mẫu đã có.

33





## 2.2. Lưu đồ cấu trúc hóa

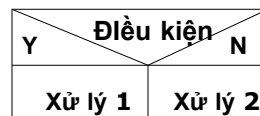
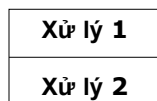
- Tác dụng của lưu đồ (flow chart)
- Quy phạm (discipline)
- Trừu tượng hóa thủ tục
- Lưu đồ cấu trúc hóa
  - Cấu trúc điều khiển cơ bản
  - Chi tiết hóa từng bước giải thuật
  - Thể hiện được trình tự điều khiển thực hiện

34

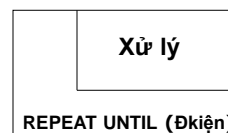
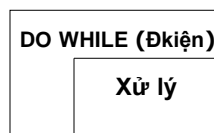
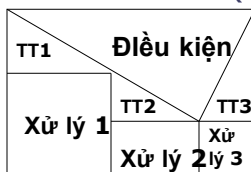


## Lưu đồ Nassi-Shneiderman (NS chart by IBM)

- a- Nối (concatination)
- b- Chọn (selection)



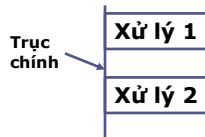
- c- Đa nhánh (CASE)
- d- Lặp (repetition)



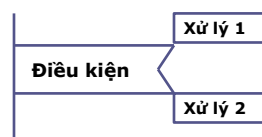
35

## Lưu đồ Phân tích bài toán (PAD chart by Hitachi)

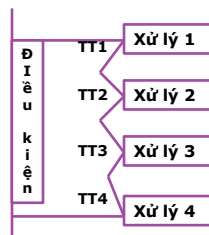
### a- Nối (concatination)



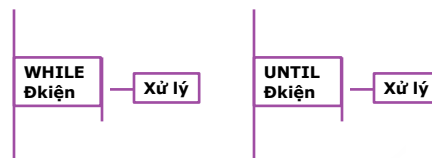
### b- Chọn (selection)



### c- Đa nhánh (CASE)



### d- Lặp (repetition)



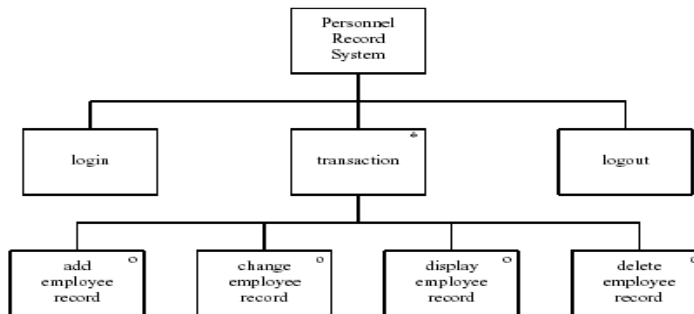
36

## 2.3 Phương pháp Jackson

- JSP: Jackson Structured Programming
- Các ký pháp:
  - Cơ sở (elementary)
  - Tuần tự (sequence)
  - Lặp
  - Rẽ nhánh

37

## Lưu đồ JSD (tiếp)



**JSD biểu diễn các chức năng của 1 HT nhân sự**

38

## Trình tự thiết kế chung

- Thiết kế cấu trúc dữ liệu (Data step)
- Thiết kế cấu trúc chương trình (Program step)
- Thiết kế thủ tục (Operation step)
- Thiết kế đặc tả chương trình (Text step)

39



## 2.4. Phương pháp Warnier

- Khái niệm chung
- Trình tự thiết kế
  - Thiết kế dữ liệu ra
  - Thiết kế dữ liệu vào
  - Thiết kế cấu trúc chương trình
  - Thiết kế lưu đồ
  - Thiết kế lệnh thủ tục
  - Thiết kế đặc tả chi tiết

40



## PHẦN IV: THIẾT KẾ VÀ LẬP TRÌNH DESIGN AND PROGRAMMING

- I. Thiết kế hệ thống
- II. Thiết kế chương trình
- III. Lập trình
  1. Lịch sử phát triển của ngôn ngữ lập trình
  2. Cấu trúc chương trình
  3. Các công cụ lập trình

41



# 1. Lịch sử ngôn ngữ lập trình

- Các ngôn ngữ thế hệ thứ nhất:
  - Ngôn ngữ lập trình mã máy (machine code)
  - Ngôn ngữ lập trình assembly
- Các ngôn ngữ thế hệ thứ hai:
  - FORTRAN, COBOL, ALGOL, BASIC
  - Phát triển 1950-1970
- Các ngôn ngữ thế hệ thứ ba
  - Ngôn ngữ lập trình cấp cao vạn năng (cấu trúc)
  - Lập trình hướng đối tượng
  - Lập trình hướng suy diễn – logic
- Các ngôn ngữ thế hệ thứ tư

42



## 2.1. Cấu trúc dữ liệu dễ hiểu

1. Lịch sử phát triển của ngôn ngữ lập trình
2. Cấu trúc chương trình

- Nên xác định tất cả các cấu trúc dữ liệu và các thao tác cần thực hiện trên từng cấu trúc dữ liệu.
- Việc biểu diễn/khai báo các cấu trúc dữ liệu chỉ nên thực hiện ở những mô đun sử dụng trực tiếp dữ liệu.
- Nên thiết lập và sử dụng từ điển dữ liệu khi thiết dữ liệu.

43

## 2.2. Cấu trúc thuật toán dễ hiểu

1. Lịch sử phát triển của ngôn ngữ lập trình
2. Cấu trúc chương trình

- Giải thuật
- 9 điểm lưu ý khi viết chương trình có cấu trúc:
  - Tuân theo quy cách lập trình
  - Một đầu vào, một đầu ra
  - Tránh GOTO, trừ khi phải ra khỏi lặp và dừng
  - Dùng comments hợp lý
  - Dùng tên biến có nghĩa, gợi nhớ
  - Cấu trúc lồng rõ ràng
  - Tránh dùng CASE / switch nhiều hoặc lồng nhau
  - Mã nguồn 1 chương trình / module nên viết trên 1 trang
  - Tránh viết nhiều lệnh trên 1 dòng

44

## 2.3. Chú thích trong chương trình

- Tại sao cần đặt các chú thích trong chương trình ?
- Vị trí đặt các chú thích trong chương trình
  - Thành phần/ Module
  - Lớp
  - Hàm/thủ tục
  - Các vị trí đặc biệt khác
- Một số quy định khi đặt chú thích:
  - Ngắn gọn
  - Gợi nhớ

45



### 3. Các công cụ lập trình

- Environments: DOS, WINDOWS, UNIX/LINUX
- Editors, Compilers, Linkers, Debuggers
- TURBO C, PASCAL
- MS C, Visual Basic, Visual C++, ASP
- UNIX/LINUX: C/C++, gcc (Gnu C Compiler)
- JAVA, CGI, perl
- C#, .NET