Phần IV
Thiết kế và Lập trình
Design and Programming

Chương 6:
Phương pháp thiết kế hệ thống

6.1. Thiết kế hệ thống là gì?

6.2. Phương pháp thiết kế hệ thống

6.1. 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)

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 bố 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

Quy trình thiết kế hệ thống (tiếp)

- Tìm ra nguồn tài nguyên chung và cơ chế điều khiển truy nhập chúng
- 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)

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

Các điểm lưu ý (tiếp)

(4) Từ kiến trúc các chức năng theo (3), 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 môđun

Các điểm lưu ý (tiếp)

(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**
  - **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)]**

6.2 Phương pháp thiết kế hệ thống

- Phương pháp thiết kế cấu trúc hóa (Structured Design) của Constantine
- Ngoài ra còn các phương pháp khác, như Phương pháp thiết kế tổng hợp (Composite Design) của Myers

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: (1) tạo kiểu luồng thông tin; (2) chỉ ra biên của luồng; (3) ánh xạ DFD sang cấu trúc chương trình; (4) xác định phân cấp điều khiển; (5) tinh lọc cấu trúc; (6) chọn mô tả kiến trúc

Thiết kế cấu trúc hóa

(1) Môđun và tham số

(2) Lưu đồ bong bóng và cấu trúc phân cấp

Lưu đồ bong bóng (Bubble chart)

Cấu trúc phân cấp (Hierarchical structured chart)

(3) Phương pháp phân chia STS (Source/Transform/Sink) và TR (Transaction)

(4) Phân tích cấu trúc hóa

(5) Chuẩn phân chia môđun

(1) Môđun

- Dãy các lệnh nhằm thực hiện chức năng (function) nào đó
- Có thể được biên dịch độc lập
- Môđun đã được dịch có thể được môđun khác gọi tới
- Giao diện giữa các môđun thông qua các biến tham số (arguments)

(2a) Lưu đồ bong bóng (Bubble chart)

- **Biểu thị luồng xử lý dữ liệu**
- **Ký pháp**

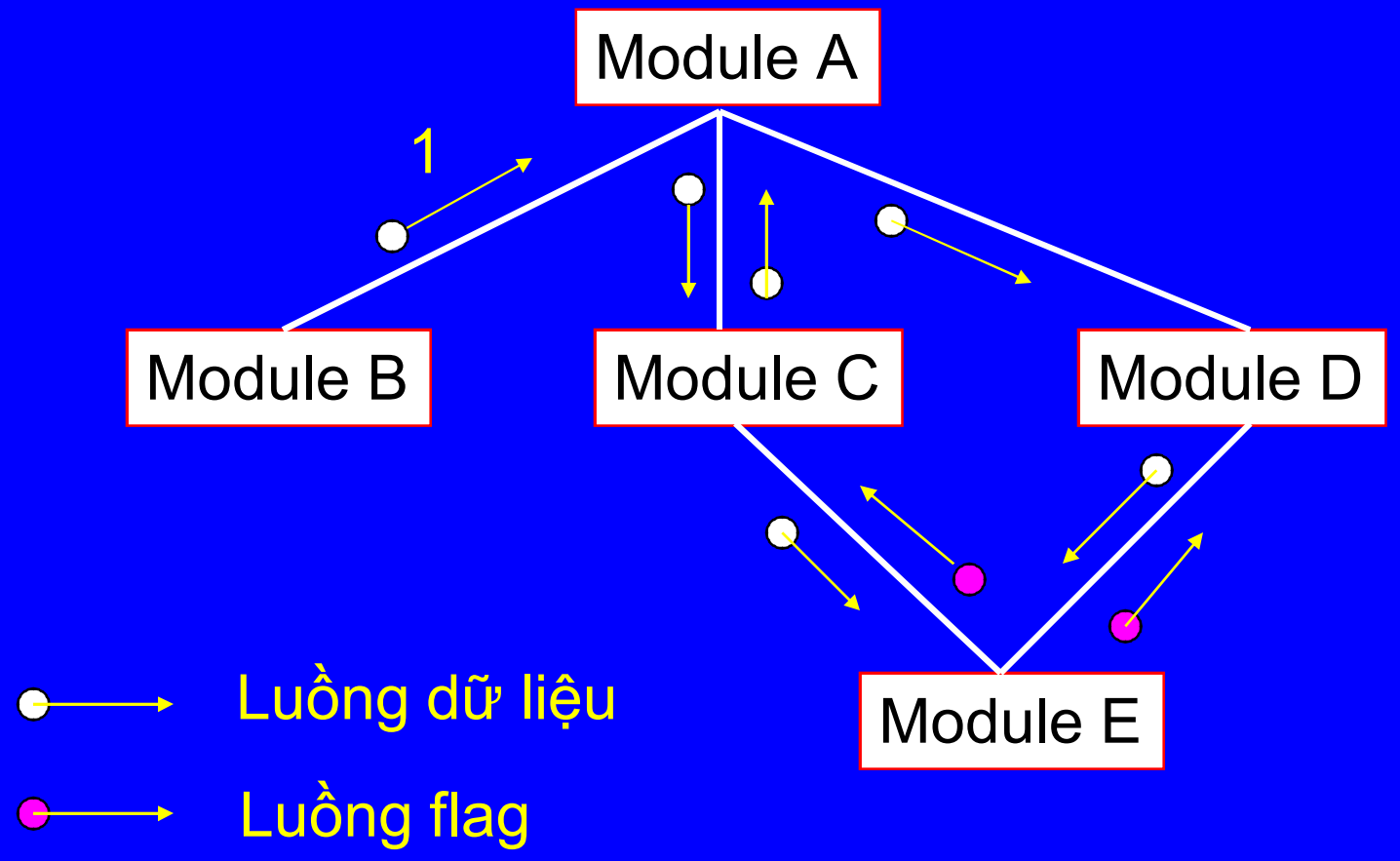
(2b) Cấu trúc phân cấp (Hierarchical structured chart)

- Là phân cấp biểu thị quan hệ phụ thuộc giữa các môđun và giao diện (interface) giữa chúng
- Các quy ước:
  - Không liên quan đến trình tự gọi các môđun, nhưng ngầm định là từ trái qua phải
  - Mỗi môđun xuất hiện trong cấu trúc 1 lần, có thể được gọi nhiều lần
  - Quan hệ trên dưới: không cần nêu số lần gọi

Hierarchical structured chart

- **Các quy ước (tiếp):**
  - **Tên môđun biểu thị chức năng ("làm gì"), đặt tên sao cho các môđun ở phía dưới tổng hợp lại sẽ biểu thị đủ chức năng của môđun tương ứng phía trên**
  - **Biến số (arguments) biểu thị giao diện giữa các môđun, biến số ở các môđun gọi/bịgọi có thể khác nhau**
  - **Mũi tên với đuôi tròn trắng biểu thị dữ liệu, đuôi tròn đen (hồng) biểu thị flag**
  - **Chiều của mũi tên là hướng truyền tham số**

Hierarchical structured chart

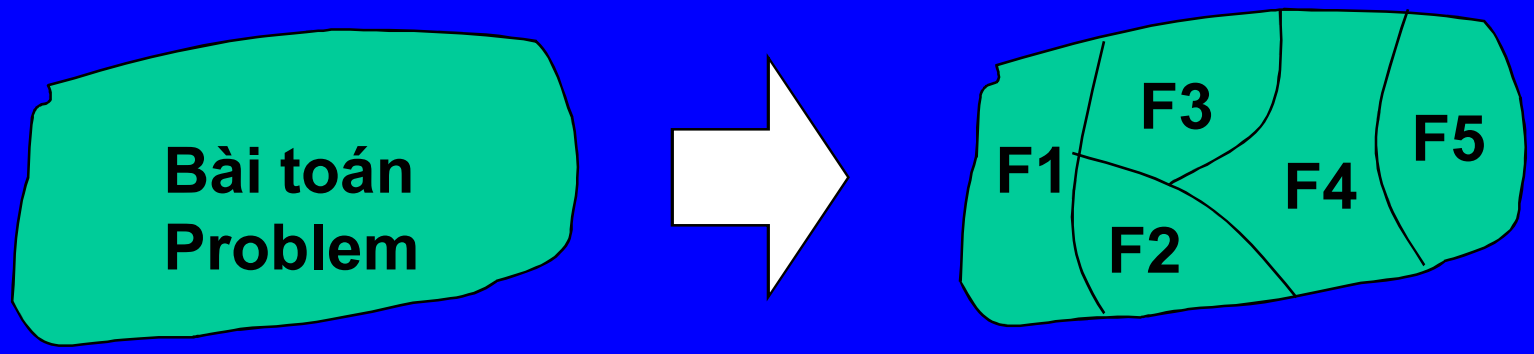


(3) Phương pháp phân chia STS, TR

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

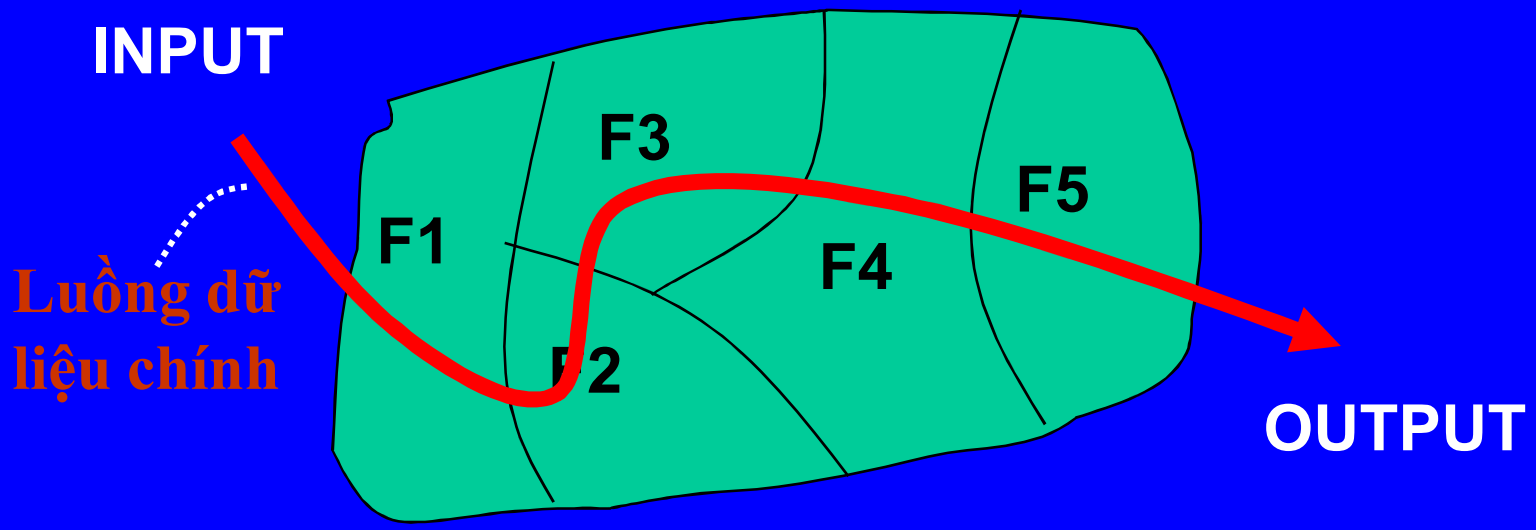
(3a) Phương pháp phân chia STS

1) Chia đối tượng "bài toán" thành các chức năng thành phần



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)

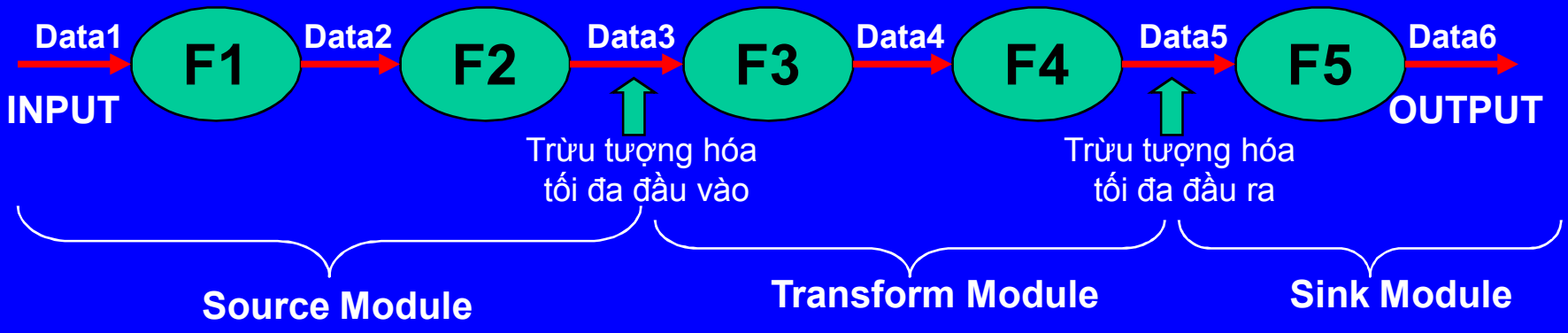


Quyết định bong bóng và dữ liệu

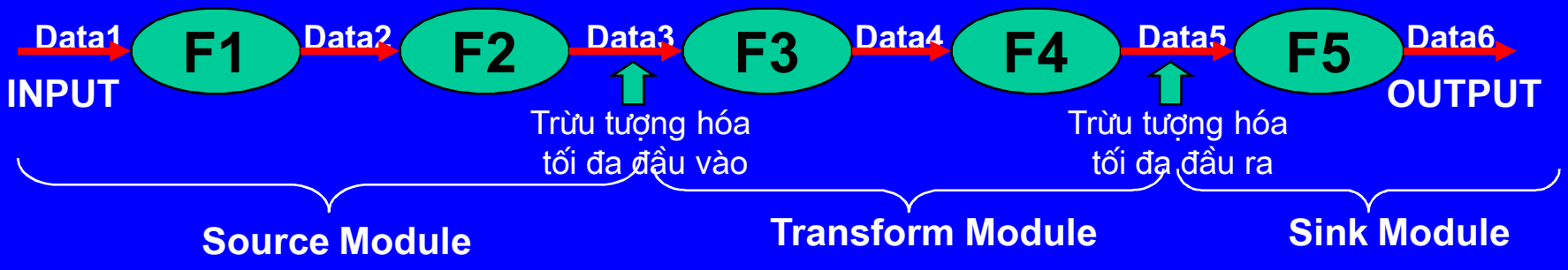
3) Theo luồng dữ liệu chính: thay từng chức năng bởi bong bóng và làm rõ dữ liệu giữa các bong bóng

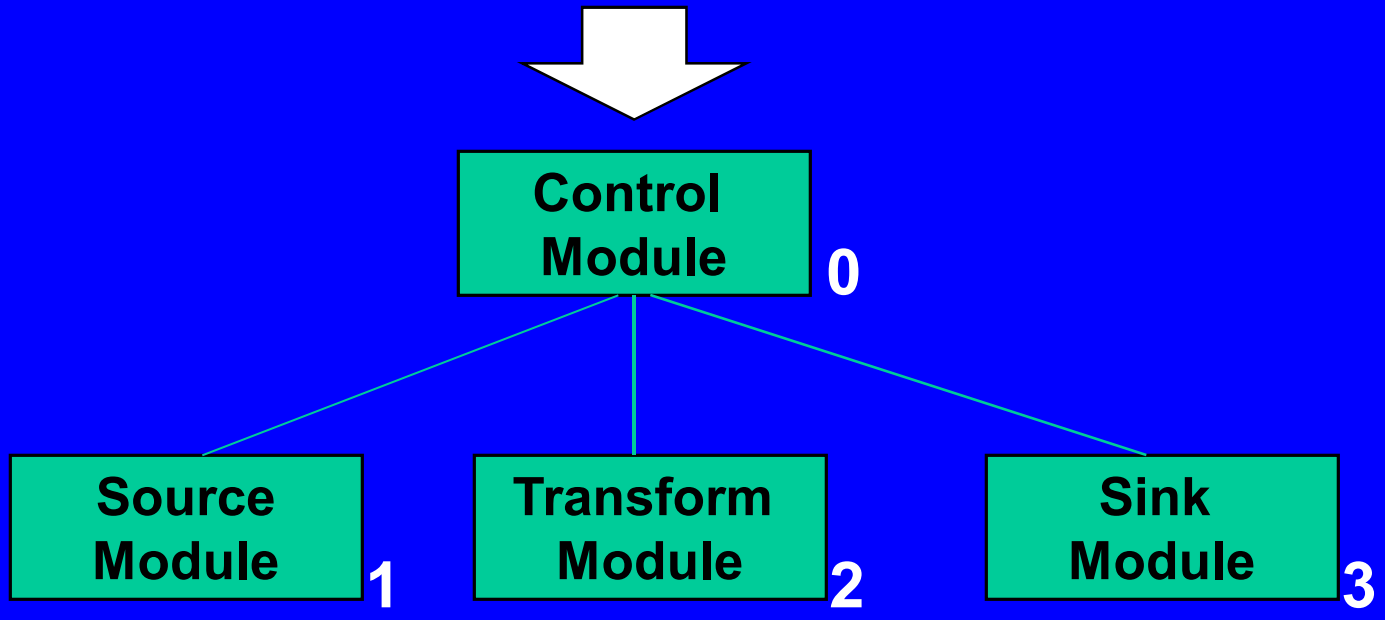
Từ sơ đồ bong bóng sang sơ đồ phân cấp

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

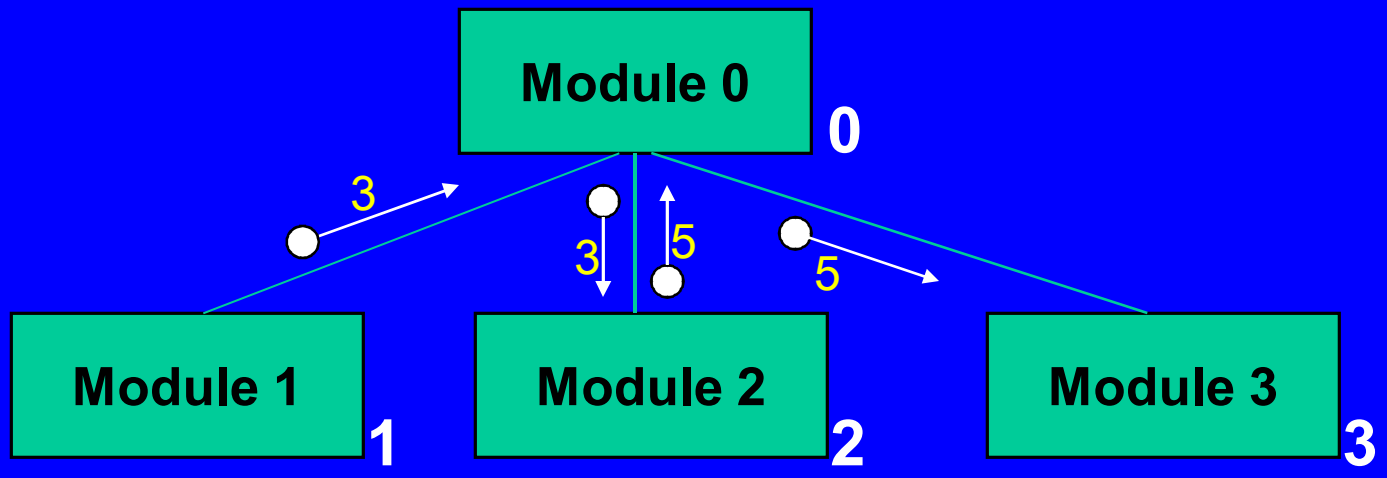


5) Chuyển sang sơ đồ phân cấp





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



7) Với từng môđun (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 lôgic khi môđun tương ứng với thuật toán đã biết thì dừng. Tổng hợp lại ta được cấu trúc phân cấp: mỗi nút là 1 môđun với số nhánh phía dưới không nhiều hơn 3

(3b) Phương pháp phân chia TR

- **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 môđun xử lý nó**
- **Phân chia môđun có thể: theo kinh nghiệm; theo tính độc lập môđun; theo số bước tối đa trong 1 môđun (ví dụ < 50) và theo chuẩn**

(4) 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

(5) Chuẩn phân chia môđun

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

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 môđun
- Kỹ thuật lập trình hiệu quả

Chương 7:
Kỹ thuật thiết kế chương trình

7.1 Thiết kế chương trình là gì ?

7.2 Phương pháp thiết kế chương trình

7.3 Công cụ thiết kế

7.1 Thiết kế chương trình là gì ?

- Là thiết kế chi tiết cấu trúc bên trong của phần mềm: thiết kế tính năng từng môđun 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

7.2 Phương pháp thiết kế chương trình

- 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 môđun và chương trình nhờ phương pháp luận thiết kế chi tiết
- Dùng quy trình thiết kế dễ chuẩn hóa từng bước

Kỹ thuật thiết kế chương trình

- 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

7.2.1 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 môđun: 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

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"

Lưu ý khi thiết kế chương trình

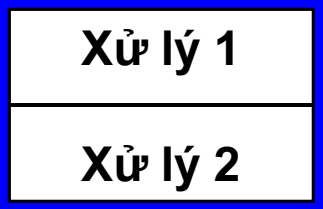
- Phụ thuộc vào kỹ năng và kinh nghiệm của người thiết kế
- Cần chuẩn hóa tài liệu đặc tả thiết kế chi tiết
- Khi thiết kế cấu trúc điều khiển của giải thuật, vì theo các quy ước cấu trúc hóa nên đôi khi tính sáng tạo của người thiết kế bị hạn chế, bó buộc theo khuôn mẫu đã có

7.2.2 Lưu đồ cấu trúc hóa

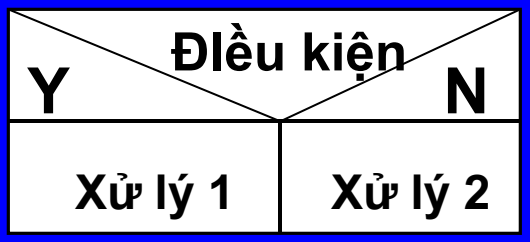
- Tác dụng của lưu đồ (flow chart)
- Quy phạm (discipline)
- Trừu tượng hóa thủ tục
- Lưu đồ cấu trúc hóa
  - Cấu trúc điều khiển cơ bản
  - Chi tiết hóa từng bước giải thuật
  - Thể hiện được trình tự điều khiển thực hiện

Lưu đồ Nassi-Shneiderman (NS chart by IBM)

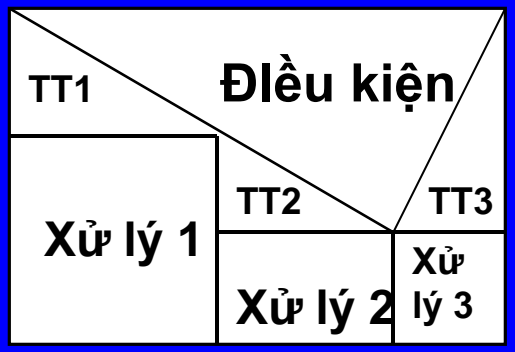
a- Nối (concatination)



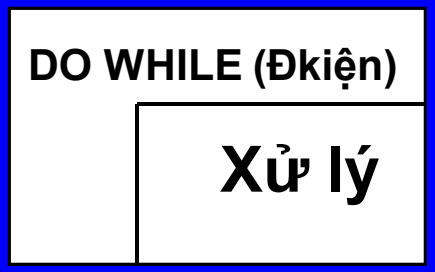
b- Chọn (selection)

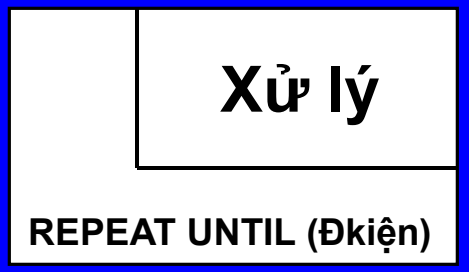


c- Đa nhánh (CASE)



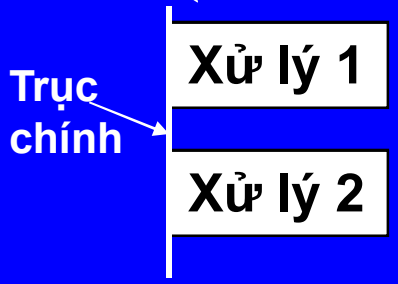
d- Lặp (repetition)



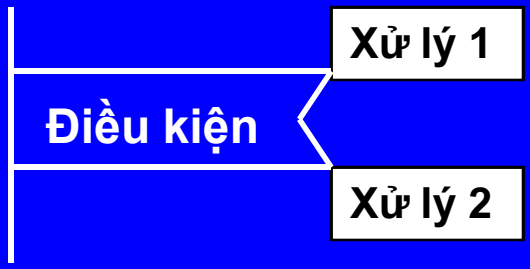


Lưu đồ Phân tích bài toán (PAD chart by Hitachi)

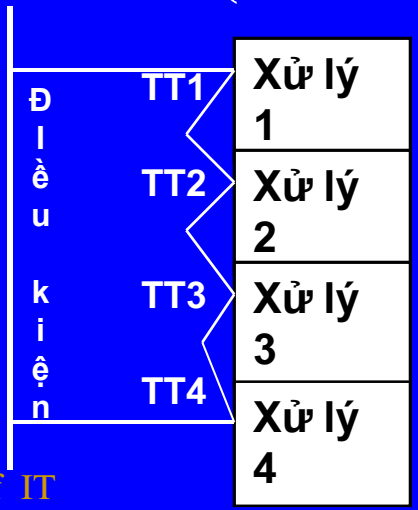
a- Nối (concatination)



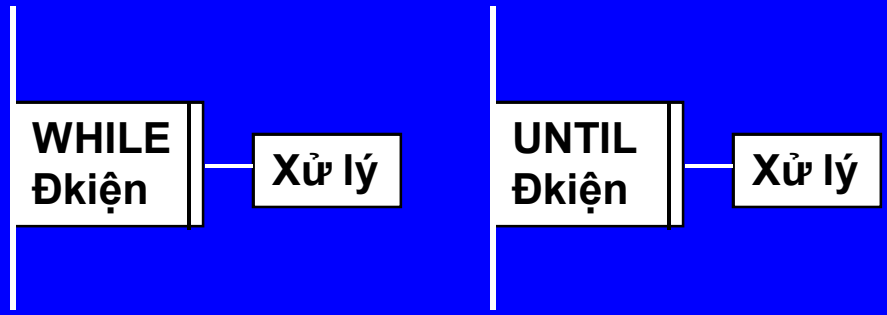
b- Chọn (selection)



c- Đa nhánh (CASE)



d- Lặp (repetition)



7.2.3 Về Phương pháp Giắc-sơn (Jackson's method)

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

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)

7.2.4 Về Phương pháp Wa-ny (Warnier's method)

- 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

Chương 8:
Kỹ thuật lập trình

8.1 Lịch sử phát triển của ngôn ngữ lập trình

8.2 Cấu trúc chương trình

- Cấu trúc dữ liệu dễ hiểu
- Cấu trúc thuật toán dễ hiểu

8.3 Các công cụ lập trình

8.1 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 thế thứ hai:
  - FOTRAN, 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
  - Truy vấn
  - Các ngôn ngữ hỗ trợ quyết định

8.2 Cấu trúc dữ liệu dễ hiểu

- 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

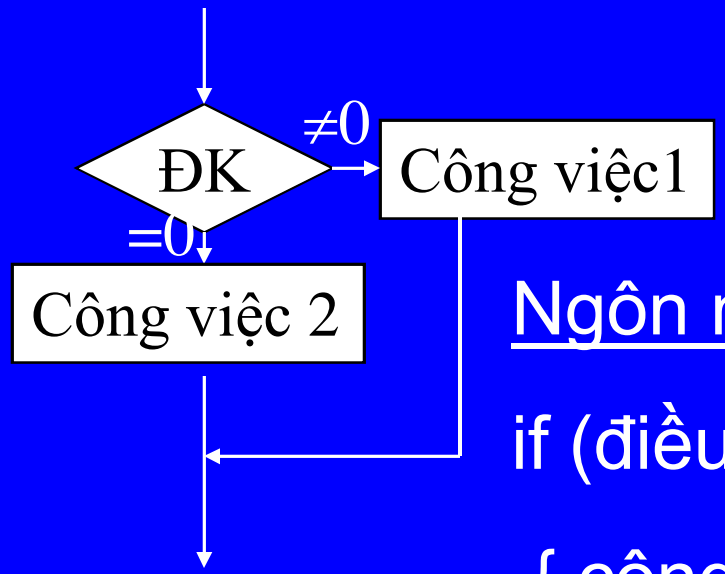
Cấu trúc thuật toán dễ hiểu

- Algorithm
- Structured coding và 9 điểm lưu ý:
  - 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 / môđun nên viết trên 1 trang
  - Tránh viết nhiều lệnh trên 1 dòng

IF THEN / IF THEN ELSE

PASCAL

```
if  điều kiện then
 begin
    công việc 1
 end;
else
 begin
    công việc 2
 end
```



Ngôn ngữ C

```
if (điều kiện)
 { công việc 1}
else
 {công việc 2}
```

CASE / switch

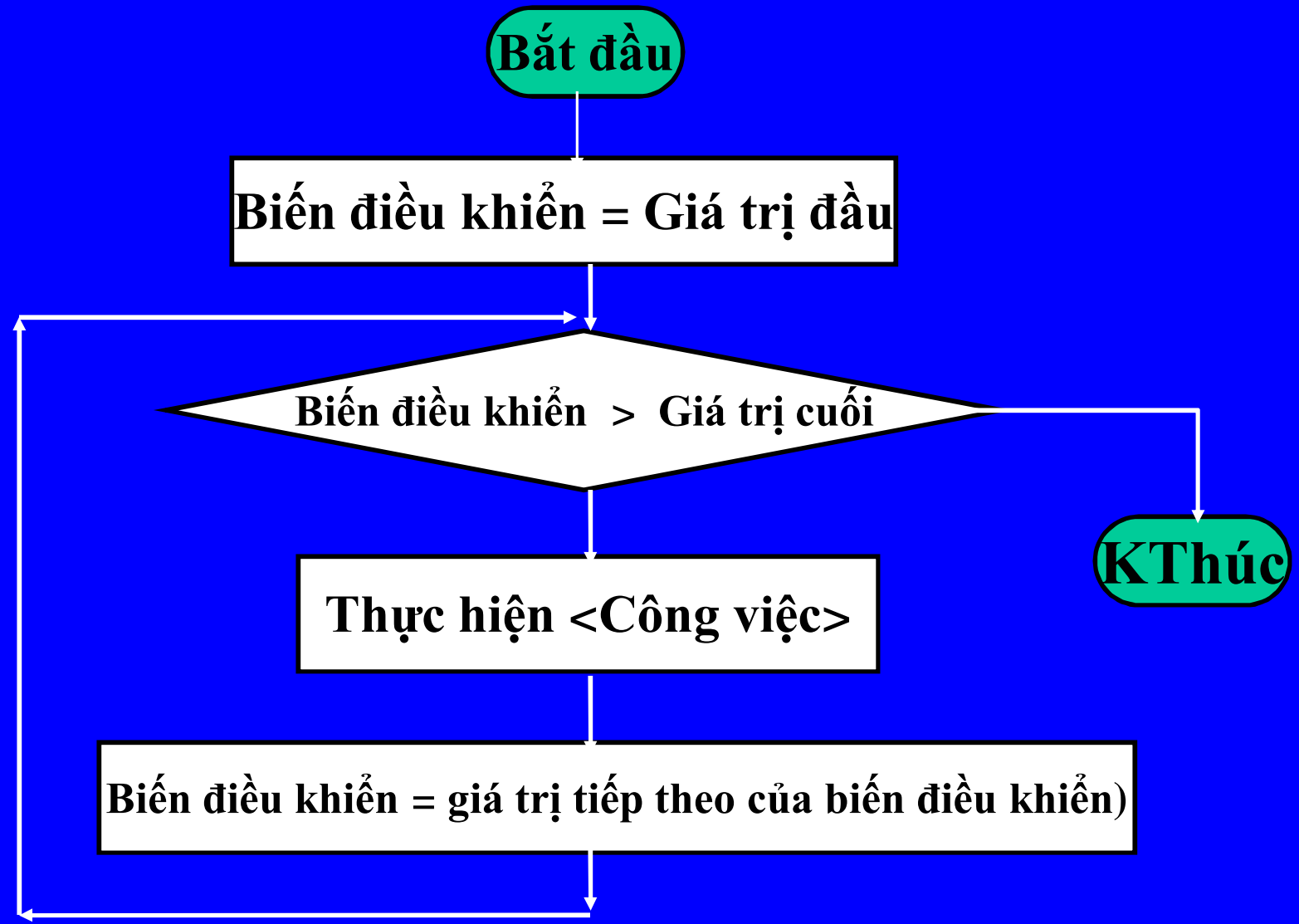
PASCAL

```
CASE <biểu thức>
OF
  gtrị1: <việc 1>;
  gtrị2: <việc 2>;
  ..........
  gtrịN: <việc N>;
ELSE
  <việc N+1>;
END;
```

Ngôn ngữ C

```
switch (<bthức>)
{
   case <gtrị1>: <việc1>;[break;]
   case <gtrị2>: <việc2>; [break;]
   case <gtrịN>: <việcN>;
[break;]
   [default : <việcN+1>; [break;] ]
}
```

FOR TO / DOWNTO



PASCAL

FOR biếnđkhiển **:= GTđầu TO GTCuối DO**

begin

<việc>

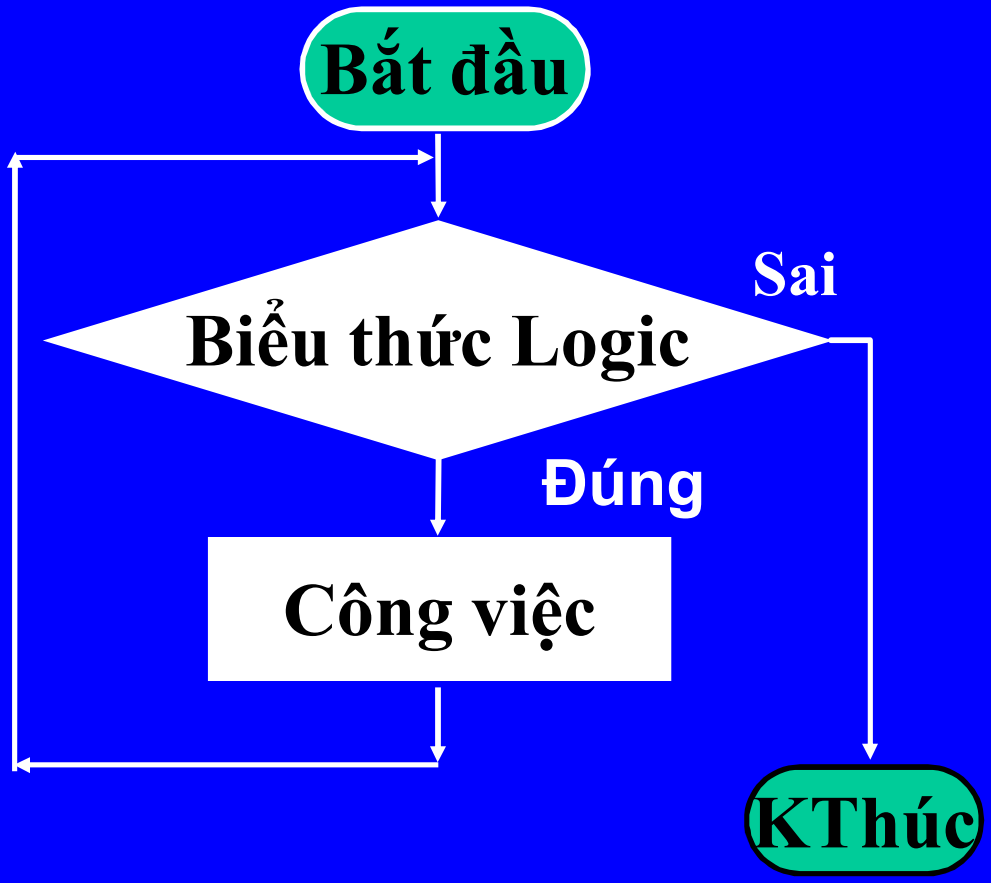
end;

Ngôn ngữ C

for ([biểuthức1] ; [biểuthứcĐK]; [biểuthức2])
{ <việc>; }

Đặc biệt: có các lệnh thoát break; continue; exit

DO WHILE



PASCAL

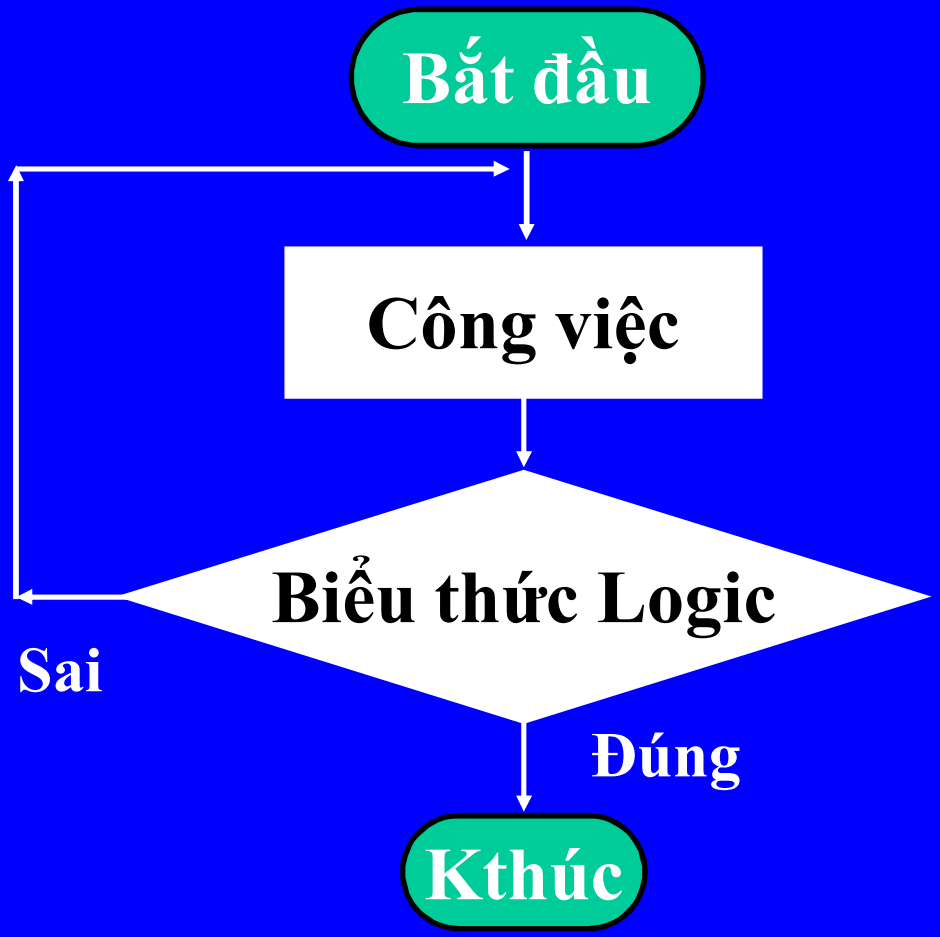
```
While BiểuthứcBoolean DO
 begin
   <Công việc>
 end;
```

Ngôn ngữ C

```
while (<biểuthứcĐK>)
   { <Công việc>; }
```

- **Kiểm tra điều kiện trước khi thực hiện**
- **Lỗi thường gặp: Lặp vô hạn**

REPEAT UNTIL



PASCAL

```
Repeat
  <Công việc>
until Biểu_thức_Boolean;
```

Ngôn ngữ C

```
do {
        <Công việc>;
    } while (<biểuthứcĐK>);
```

- **Có sự khác nhau giữa hai ngôn ngữ?**

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