TRƯỜNG CAO ĐẲNG CÔNG THƯƠNG TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN

KIỂM THỬ PHẦN MỀM
(Software Testing)

1

MỤC TIÊU

- **Kiến thức:**
  - Giới thiệu quy trình phát triển phần mềm, vai trò của kiểm thử phần mềm trong mỗi giai đoạn.
  - Giải thích các thuật ngữ, các khái niệm cơ bản trong kiểm thử phần mềm.
  - Phân biệt các phương pháp, các chiến lược kiểm thử phần mềm.
  - Vận dụng một số kỹ thuật cơ bản để thiết kế test case.
  - Giải thích chu kỳ sống của lỗi (bug), các bước xử lý bug.

2

MỤC TIÊU (tt)

- **Kỹ năng:**
  - Biết cách tìm bug, phân tích và viết báo cáo (bug report).
  - Sử dụng một số công cụ quản lý Test Case, công cụ quản lý Bug.
  - Sử dụng được một số công cụ kiểm thử tự động.
  - Rèn luyện kỹ năng làm việc độc lập và làm việc nhóm.

3

TIÊU CHÍ ĐÁNH GIÁ

- **Điểm quá trình: 40%**
  - **Điểm chuyên cần: 10%**
  - **Kiểm tra giữa ký (KT1): 20%**
  - **Kiểm tra cuối kỳ (KT2): 20%**
- **Đồ án môn học (thi) 60%**
  - **Điểm bài báo cáo (file cứng + file mềm): 30%**
  - **Điểm trả lời câu hỏi (lúc báo cáo): 30%**

4

NỘI DUNG

- **Chương 1: Tổng quan về kiểm thử phần mềm.**
- **Chương 2: Cơ sở toán học rời rạc cho kiểm thử**
- **Chương 3: Kỹ thuật kiểm thử phần mềm**
- **Chương 4: Chiến lược kiểm thử phần mềm**

5

TÀI LIỆU HỌC TẬP

- **Giáo trình**
  - Phạm Ngọc Hùng, Trương Anh Hoàng, Đặng Văn Hưng, *Giáo trình kiểm thử phần mềm*, 2014.
  - Nguyễn Xuân Huy (1994), *Công nghệ phần mềm*, Đại học Tổng hợp Tp. Hồ Chí Minh.
  - Roger S. Pressman, *Software Engineering: A Practioner's Approach*. 5th Ed., McGraw-Hill, 2001.
  - Roger S. Pressman, *Introduction to Software Engineering*, Ngô Trung Việt dịch, Nhà xuất bản Giáo dục 1997

6

TÀI LIỆU HỌC TẬP

- **Phần mềm thực hành:**
  - **Katalon Studio, Selenium, Junit,**
  - **Website học tập:**
    - ✓ https://vntesters.com
    - ✓ https://www.testingvn.com
    - ✓ …

7

Chương 1 TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM

- **Phần mềm**
- **Quy trình phát triển phần mềm**
- **Chất lượng phần mềm**
- **Lỗi phần mềm**
- **Tổng quan về kiểm thử phần mềm**

8

1.1 PHẦN MỀM – ĐỊNH NGHĨA

- **Phần mềm ?**

"Software is:

(1)instructions (computer programs) that when executed provide desired features, function, and performance;

(2) data structures that enable the programs to adequately manipulate information, and

(3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs." (Software Engineering, Roger S. Pressman)

Software

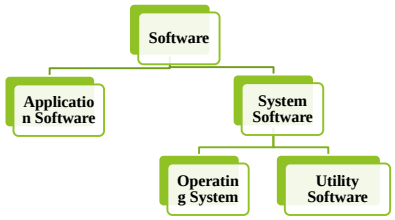
9

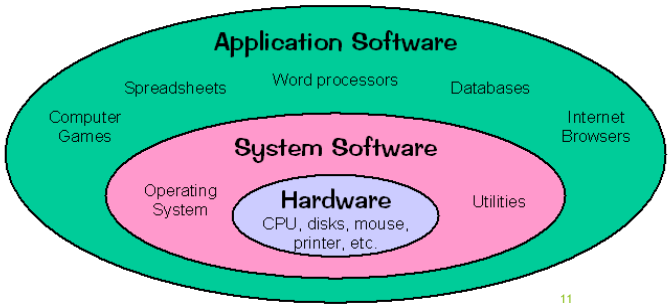
1.1 PHẦN MỀM – ĐẶC TÍNH

- **Đặc tính phần mềm ?**
  - Phần mềm được phát triển (development) hay kỹ nghệ (engineering), nó không được chế tạo (manufacture) theo nghĩa cổ điển.
    - software development và hardware manufacturing
  - Phần mềm không "tự hỏng" nhưng thoái hoá theo thời gian.
  - Phần lớn phần mềm được xây dựng theo yêu cầu của khách hàng.
  - Sự phức tạp và tính luôn thay đổi luôn là bản chất của phần mềm.
  - Phần mềm được phát triển theo nhóm.

10

1.1 PHẦN MỀM – PHÂN LOẠI





11

1.1 PHẦN MỀM – PHÂN LOẠI

- **Phần mềm thời gian thực (Real-time SW)**
- **Phần mềm nghiệp vụ (Business SW)**
- **Phần mềm tính toán KH&KT (Eng.&Scie. SW)**
- **Phần mềm nhúng (Embedded SW)**
- **Phần mềm trên Web (Web-based SW)**
- **Phần mềm trí tuệ nhân tạo (AI SW)**
- **Tiện ích (Utility)**
- **Phần mềm phát triển (Development SW)**

12

1.1 PHẦN MỀM – CÔNG NGHỆ PHẦN MỀM

- **Bauer [1969]:** CNPM là việc thiết lập và sử dụng các nguyên tắc công nghệ đúng đắn dùng để thu được phần mềm một cách kinh tế vừa tin cậy vừa làm việc hiệu quả trên các máy thực
- **IEEE [1993]:** CNPM là
  (1) việc áp dụng phương pháp tiếp cận có hệ thống, bài bản và được lượng hóa trong phát triển, vận hành và bảo trì phần mềm;
  (2) nghiên cứu các phương pháp tiếp cận được dùng trong (1)
- **Pressman [1995]:** CNPM là bộ môn tích hợp cả quy trình, các phương pháp, các công cụ để phát triển phần mềm máy tính.
- …

13

1.1 PHẦN MỀM – CÔNG NGHỆ PHẦN MỀM (tt)

- **Công nghệ phần mềm** *là lĩnh vực khoa học về các phương pháp luận, kỹ thuật và công cụ tích hợp trong quy trình sản xuất và vận hành phần mềm nhằm tạo ra phần mềm với những chất lượng mong muốn.* [Software Engineering is a scientific field to deal with methodologies, techniques and tools integrated in software production-maintenance process to obtain software with desired qualities].

14

1.2 QUY TRÌNH PHÁT TRIỂN PHẦN MỀM

- **Quy trình phát triển phần mềm (Software Development Process - SDP) là tiến trình xây dựng sản phẩm phầm mềm hay nâng cấp phần mềm đang có.**

15

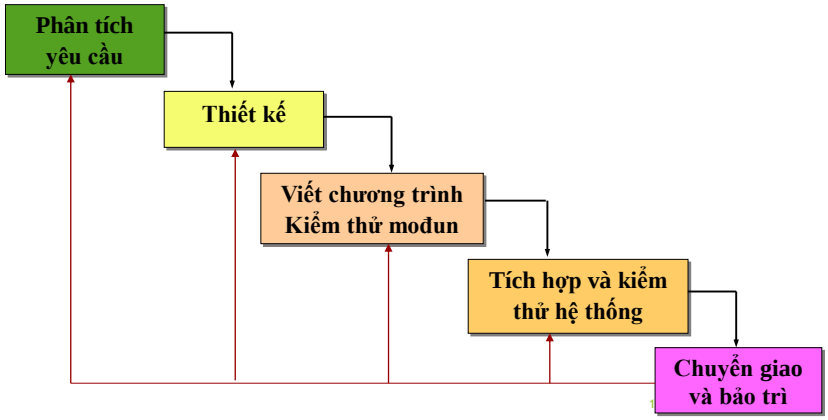
1.2 QUY TRÌNH PHÁT TRIỂN PHẦN MỀM (tt)

- **Có nhiều quy trình phát triển phần mềm khác nhau**
  - **Đóng vai trò quyết định chất lượng PM.**
  - **Các nhóm công việc được triển khai theo những cách khác nhau.**
  - **Có 4 nhóm công việc nền tảng: Đặc tả yêu cầu – Phát triển – Kiểm thử - Cài đặt và bảo trì**
  - **Một PM có thể dùng nhiều mô hình khác nhau.**
  - **Không phải tất cả các mô hình đều thích hợp cho mọi phần mềm ứng dụng.**

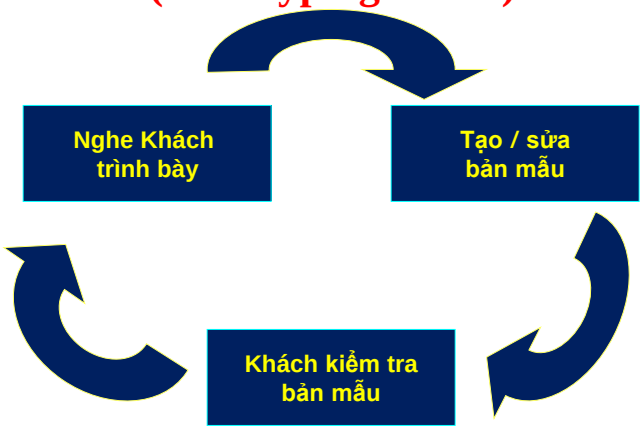
16

1.2 QUY TRÌNH PHÁT TRIỂN PHẦN MỀM (tt)
MÔ HÌNH THÁC NƯỚC
(Waterfall – Boehm 1970)

- Vòng đời (life cycle) phần mềm

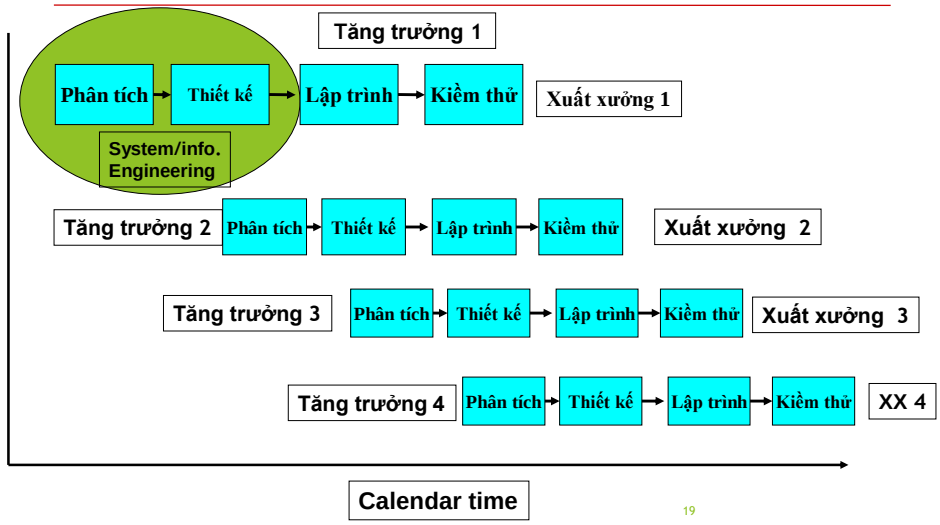


1.2 QUY TRÌNH PHÁT TRIỂN PHẦN MỀM (tt)
MÔ HÌNH CHẾ THỬ
(Prototyping model)



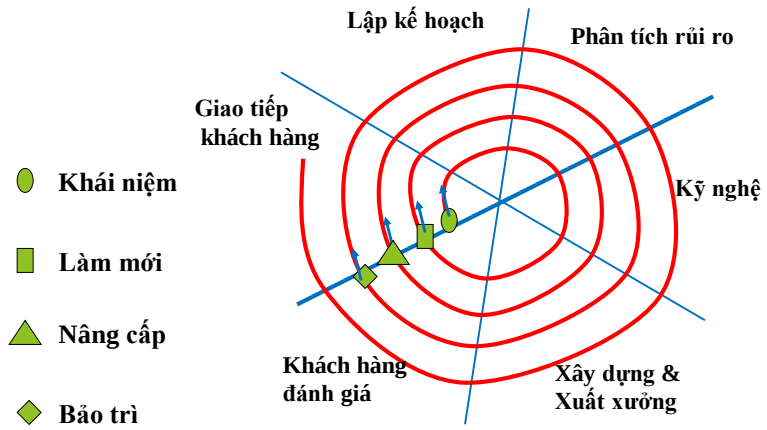
18

1.2 QUY TRÌNH PHÁT TRIỂN PHẦN MỀM (tt)
MÔ HÌNH TĂNG TRƯỞNG
(D.R.Graham 1988)



19

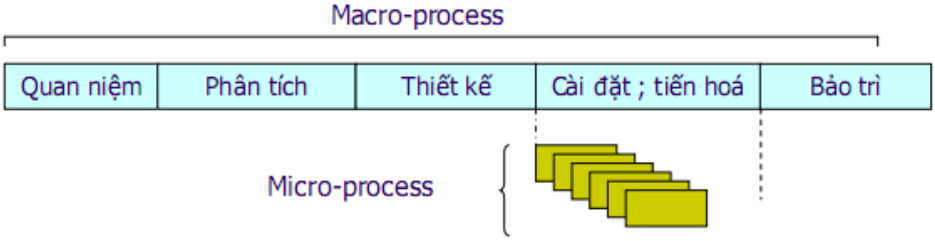
1.2 QUY TRÌNH PHÁT TRIỂN PHẦN MỀM (tt)
MÔ HÌNH XOẮN ỐC
(SPIRAL)



20

1.2 QUY TRÌNH PHÁT TRIỂN PHẦN MỀM (tt)

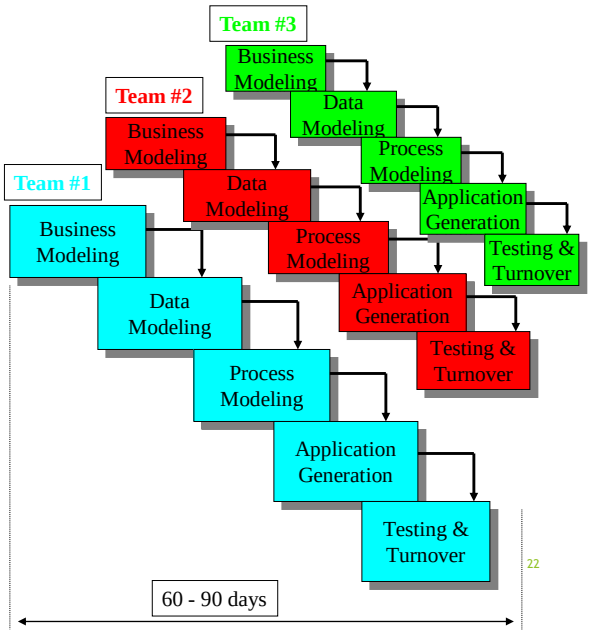
MÔ HÌNH BOOCH (1996)



21

1.2 QUY TRÌNH PHÁT TRIỂN PHẦN MỀM (tt)

MÔ HÌNH PHÁT TRIỂN ỨNG DỤNG NHANH



22

1.2 QUY TRÌNH PHÁT TRIỂN PHẦN MỀM (tt)
TỶ LỆ CÁC CÔNG VIỆC CÁC GIAI ĐOẠN PT

Giai đoạn	Phân tích yêu cầu	Thiết kế sơ bộ	Thiết kế chi tiết	Lập trình và kiểm thử đơn vị	Tích hợp và kiểm thử tích hợp	Kiểm thử hệ thống
Hai thập kỉ 1960 - 1970	10%			80%	10%	
Thập kỉ 1980	20%		60%		20%	
Thập kỉ 1990	40%	30%		30% - 40%		

23

1.3 CHẤT LƯỢNG PHẦN MỀM – ĐỊNH NGHĨA

- Chất lượng của một sản phẩm được thể hiện bằng các đặc trưng phù hợp với đặc tả của nó.
- Chất lượng phần mềm đặc trưng cho "độ tốt, độ tuyệt hảo" của phần mềm, gồm các yếu tố về chất lượng:
  - Tính đúng đắn (hành vi đúng như đặc tả)
  - Tính hiệu quả (tiết kiệm thời gian và tiền bạc)
  - Độ tin cậy
  - Tính khả kiểm thử (kiểm thử được và dễ)
  - Dễ học
  - Dễ sử dụng
  - Dễ bảo trì ....

24

1.3 CHẤT LƯỢNG PHẦN MỀM – ĐẢM BẢO CHẤT LƯỢNG PHẦN MỀM

- Mục đích của nhóm phát triển PM là có PM chất lượng cao.
- Hạn chế thấp nhất việc phát sinh lỗi.
- Đảm bảo chất lượng phần mềm là một hoạt động có hệ thống và kế hoạch, bao gồm các hoạt động
  - Áp dụng các phương pháp kỹ thuật,
  - Tiến hành các cuộc xét duyệt kỹ thuật chính thức,
  - Kiểm thử phần mềm,
  - Buộc tôn trọng các chuẩn,
  - Kiểm soát thay đổi,
  - Đo chất lượng,
  - Báo cáo, lưu giữ kết quả.

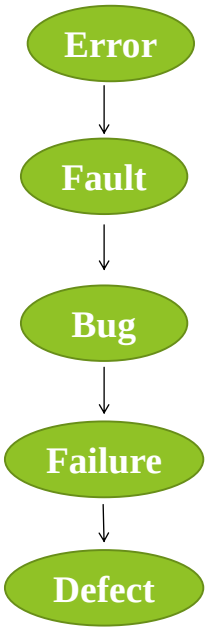
25

1.4 LỖI PHẦN MỀM – CÁC ĐỊNH NGHĨA

- **Lỗi (Error)**: Lỗi là những vấn đề mà con người mắc phải trong quá trình phát triển các sản phẩm phần mềm.
- **Sai (Fault)**: Sai là kết quả của lỗi. Sai là một biểu diễn của lỗi dưới dạng một biểu thức, chẳng hạn chương trình, văn bản, sơ đồ dòng dữ liệu, biểu đồ lớp, ...
- **Thất bại (Failure)**: Thất bại xuất hiện khi một lỗi được thực thi.
- **Sự cố (Incident)**: Sự cố là triệu chứng liên kết với một thất bại và thể hiện cho người dùng hoặc người kiểm thử về sự xuất hiện của thất bại này.

26

1.4 LỖI PHẦN MỀM – CÁC ĐỊNH NGHĨA




```
1   #include<stdio.h>
2
3   int main ()
4   {
5   int value1, value2, ans;
6
7   value1 = 5;
8   value2 = 3;
9
10  ans = value1 - value2;  // ----> Bug
11
12  printf("The addition of 5 + 3 = %d.", ans);
13
14  return 0;
15  }
```

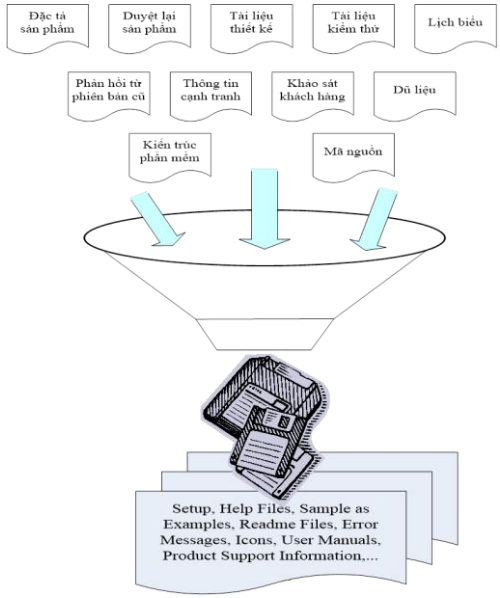
The addition of 5 + 3 = 2.

1.4 LỖI PHẦN MỀM – ĐỊNH NGHĨA

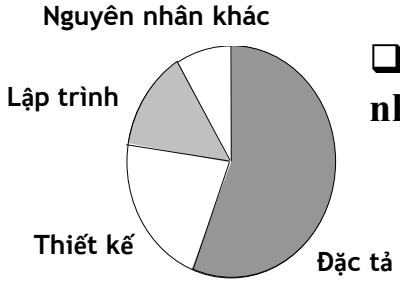
- **Định nghĩa Lỗi phần mềm?**
  - ***Lỗi phần mềm là sự không khớp giữa chương trình và đặc tả của nó.***
- **Lỗi phần mềm xuất hiện theo các dạng sau:**
  - **Sai: Sản phẩm được xây dựng khác với đặc tả.**
  - **Thiếu: Một yêu cầu đã được đặc tả nhưng lại không có trong sản phẩm được xây dựng.**
  - **Thừa: Một yêu cầu được đưa vào sản phẩm mà không có trong đặc tả.**
  - **Phần mềm khó hiểu, khó sử dụng, chậm hoặc dễ gây cảm nhận rằng phần mềm hoạt động không đúng.**

1.4 LỖI PHẦN MỀM

- Lỗi phần mềm xảy ra ở tất cả các công đoạn của qui trình phát triển một sản phẩm phần mềm. Việc kiểm thử cũng vì thế phải được tiến hành trong tất cả các phần tạo nên một sản phẩm phần mềm.



1.4 LỖI PHẦN MỀM – ĐẶC TẢ PHẦN MỀM



Đặc tả: ~ 80%

- Nguyên nhân làm đặc tả nhiều lỗi ?
  - Đặc tả không được viết ra
  - Đặc tả không đủ cẩn thận
  - Đặc tả thay đổi
  - Chưa phối hợp tốt trong toàn nhóm phát triển

30

1.4 LỖI PHẦN MỀM –Ví dụ

- **Ví dụ: chương trình tính tiền lương được đặc tả cho từng nhân viên theo qui định làm tròn đến hàng đơn vị, với công thức (1.1)**

  $Lương_i$ = round(hsl_i*lcb,**0**) **(1.1)**

Nhưng khi lập trình:

Lương$_i$ = round(hsl_i *lcb,**-2**) **(1.2)**

31

1.4 LỖI PHẦN MỀM –Ví dụ (tt)

- **Ví dụ: Như vậy dẫn đến sai số như sau**

Stt	Học và tên	hsl	Chưa tròn	CT(1.1)	CT(1.2)	Chênh lệnh tiền lương
1	Lê A	3.99	1,312,710.47	1,312,710	1,312,700	-10
2	Trần B	2.34	769,859.55	769,860	769,900	+40
3	Nguyễn C	4.32	1,421,280.43	1,421,280	1,421,300	+20
4	Hoàng D	2.67	878,429.95	878,430	878,400	-30
Tổng cộng				4,382,280	4,382,300	+20

32

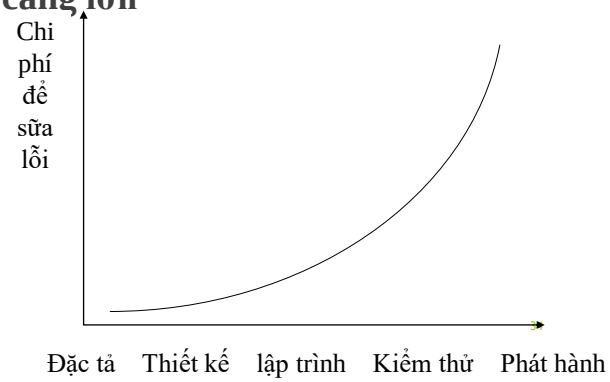
1.4 LỖI PHẦN MỀM – CHI PHÍ SỬA LỖI

- Kiểm thử và sửa lỗi có thể được thực hiện tại bất kỳ giai đoạn nào của vòng đời phần mềm.
- Chi phí cho việc tìm và sửa lỗi tăng một cách đáng kể theo quá trình phát triển:
  - Không đáng kể khi thay đổi yêu cầu ở lần duyệt yêu cầu đầu tiên.
  - Tăng lên gấp bội khi thay đổi yêu cầu lúc đã lập trình.
  - Không đáng kể nếu lập trình viên tự phát hiện lỗi của mình.

33

1.4 LỖI PHẦN MỀM – CHI PHÍ SỬA LỖI

- "Sửa một lỗi trước khi phát hành một phần mềm sẽ tốn chi phí ít hơn rất nhiều so với việc khắc phục nó sau khi đã phát hành. "
- Lỗi được phát hiện càng muộn thì chi phí cho việc sửa lỗi càng lớn



1.5 KIỂM THỬ PHẦN MỀM – CÁC ĐỊNH NGHĨA

- **Kiểm thử**
  - **IEEE: Kiểm thử là tiến trình vận hành hệ thống hoặc thành phần dưới những điều kiện xác định, quan sát hoặc ghi nhận kết quả và đưa ra đánh giá về hệ thống hoặc thành phần đó.**
  - **Myers: Kiểm thử là tiến trình thực thi chương trình với mục đích tìm thấy lỗi (The art of software testing)**

=> Kiểm thử phần mềm là quá trình thực thi một hệ thống phần mềm để xác định xem phần mềm đó có đúng với đặc tả không và thực hiện như mong đợi hay không.

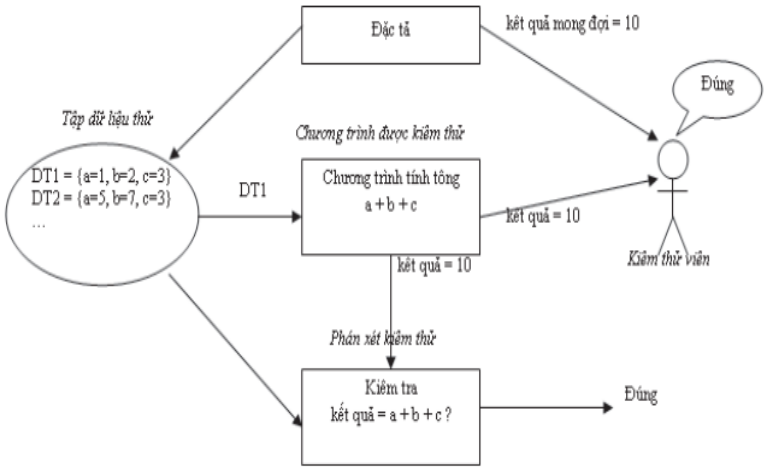
35

1.5 KIỂM THỬ PHẦN MỀM – CÁC ĐỊNH NGHĨA

- **Dữ liệu thử (test data): dữ liệu vào cần cung cấp cho phần mềm trong khi thực thi**
- **Kịch bản kiểm thử (test scenario): các bước thực hiện khi tiến hành kiểm thử**
- **Kết quả kiểm thử (test oracle): đánh giá kết quả của kiểm thử**
  - **Tự động: chương trình**
  - **Thủ công: con người**
- **Kiểm thử viên (tester): người thực hiện kiểm thử**
- **Ca kiểm thử (test case)**
  - **tập dữ liệu thử**
  - **điều kiện thực thi và kết quả mong đợi**

36

1.5 KIỂM THỬ PHẦN MỀM – CÁC ĐỊNH NGHĨA



37

1.5 KIỂM THỬ PHẦN MỀM (tt)

- **Vai trò:**
  - **Đóng vai trò tối quan trọng, quyết định đến việc đánh giá chất lượng phần mềm.**
  - **Thông qua chu trình " kiểm thử - tìm lỗi - sửa lỗi", chất lượng của sản phẩm phần mềm sẽ được cải tiến.**
- **Mục đích:**
  - **Tìm ra lỗi chưa được phát hiện, tìm một cách sớm nhất và đảm bảo rằng lỗi đã được sửa.**
- **Mục tiêu:**
  - **Thiết kế tài liệu kiểm thử một cách có hệ thống và thực hiện nó sao cho có hiệu quả, nhưng tiết kiệm được thời gian, công sức và chi phí.**

38

1.5 KIỂM THỬ PHẦN MỀM (tt)

- Kiểm thử ≠ Gỡ rối (debug)
- Kiểm thử nhằm phát hiện lỗi
- Gỡ rối:
  - Nhằm xác định bản chất lỗi và định vị lỗi trong chương trình.
  - Tiến hành sửa lỗi

TESTING	DEBUGGING
a) Finding and locating of a defect	a) Fixing that defect
b) Done by Testing Team	b) Done by Development team
c) Intention behind is to find as many defect as possible	c) Intention is to remove those defects

1.5 KIỂM THỬ PHẦN MỀM – QUY TRÌNH KIỂM THỬ

- **Phân tích tĩnh:**
- Dựa trên việc khảo sát các tài liệu được xây dựng trong quá trình phát triển sản phẩm như tài liệu đặc tả nhu cầu người dùng, mô hình phần mềm, hồ sơ thiết kế và mã nguồn phần mềm.
- Các phương pháp phân tích tĩnh truyền thống bao gồm việc khảo sát đặc tả và mã nguồn cùng các tài liệu thiết kế.
- Công việc này **không** động đến việc thực thi chương trình mà chỉ duyệt, lý giải về tất cả các hành vi có thể của chương trình khi được thực thi.

1.5 KIỂM THỬ PHẦN MỀM – QUY TRÌNH KIỂM THỬ (tt)

- **Phân tích động:**
  - **Liên quan đến việc thực thi chương trình để phát hiện những thất bại có thể có của chương trình, hoặc quan sát các tính chất nào đó về hành vi và hiệu quả (performance).**
  - **Không thể thực thi chương trình trên tất cả các dữ liệu đầu vào có thể, ta chỉ có thể chọn một tập con các dữ liệu đầu vào để thực thi, gọi là các "ca kiểm thử".**

=> Phân tích tĩnh và động là hai kỹ thuật bổ sung cho nhau và cần được làm lặp đi lặp lại nhiều trong quá trình kiểm thử.

41

1.5 KIỂM THỬ PHẦN MỀM – QUY TRÌNH KIỂM THỬ (tt)

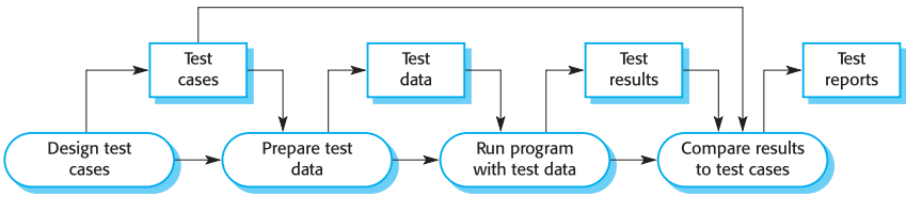
- **Kiểm thử thường bao gồm các bước**
  - **Thiết kế các ca kiểm thử**
  - **Tạo dữ liệu thử: chọn tập các dữ liệu thử đại diện từ miền dữ liệu vào dựa trên các tiêu chuẩn chọn dữ liệu thử**
  - **Thực thi chương trình trên dữ liệu thử**
    - **Cung cấp dữ liệu thử**
    - **Thực thi**
    - **Ghi nhận kết quả**
  - **Quan sát kết quả kiểm thử**
    - **Thực hiện trong khi hoặc sau khi thực thi**
    - **So sánh kết quả nhận được và kết quả mong đợi**

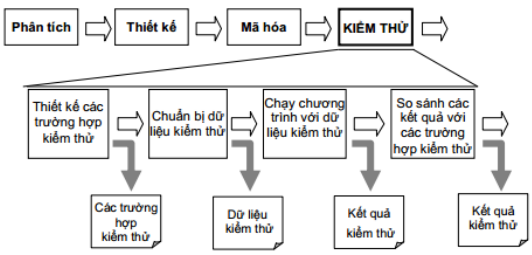
42

1.5 KIỂM THỬ PHẦN MỀM – CÁC HOẠT ĐỘNG KIỂM THỬ

43

1.5 KIỂM THỬ PHẦN MỀM – QUY TRÌNH KIỂM THỬ (tt)





Hình 1.6 – Qui trình kiểm thử phần mềm

1.5 KIỂM THỬ PHẦN MỀM – CÁC MỨC KIỂM THỬ

- **Đơn vị**
  - Tìm lỗi trong từng đơn vị
- **Tích hợp**
  - Tìm lỗi khi ghép các đơn vị
- **Hệ thống**
  - Tìm lỗi khi hệ thống đã tích hợp xong, trước khi phát hành, chuyển giao
- **Chấp thuận**
  - Người sử dụng dùng thử xem hệ thống đáp ứng đúng mong muốn chưa.

45

- **Kiểm thử đơn vị *(unit testing)***
- **Kiểm thử tích hợp *(integration testing)***
- **Kiểm thử hệ thống *(system testing)***
  - **KT chức năng *(functional test: system&interface)***
  - **KT phục hồi *(recovery test)***
  - **KT chịu tải *(extra: stress & load test)***
  - **KT thi hành *(performance test)***
  - **KT an ninh *(security test)***
- **Kiểm thử chấp nhận *(aceptance testing)***
  - **Kiểm thử alpha *(alpha testing)***
    - ***Người dùng thực hiện***
    - ***Trong môi trường được quản lý***
  - **Kiểm thử beta *(beta testing)***
    - ***Người dùng thực hiện***
    - ***Trong môi trường thực***

46

1.5 KIỂM THỬ PHẦN MỀM – PHƯƠNG PHÁP KIỂM THỬ

- **Hai phương pháp phổ biến:**
  - **Kiểm thử hộp trắng (white box)**
  - **Kiểm thử hộp đen (black box)**
- **Các chiến lược Kiểm thử**
  - **Kiểm thử từ trên xuống/dưới lên/lai (tích hợp)**
  - **Kiểm thử vụ nổ lớn (big bang –tích hợp)**
  - **Kiểm thử hồi quy (tích hợp)**
  - **Kiểm thử luồn sợi (hệ thời gian thực)**

47

1.5 KIỂM THỬ PHẦN MỀM – KHÓ KHĂN

- **Liên quan đến tiến trình phát triển:**
  - **Gồm nhiều giai đoạn phát triển**
  - **Cái ra của một giai đoạn là cái vào của giai đoạn khác.**
  - **Mất mát thông tin**
- **Về mặt con người:**
  - **Thiếu đào tạo.**
  - **Ít chú trọng vai trò kiểm thử.**
- **Về mặt kỹ thuật:**
  - **Không tồn tại thuật toán tổng quát có thể chứng minh sự đúng đắn hoàn toàn của bất kỳ một chương trình nào.**

48

CHƯƠNG 2. CƠ SỞ TOÁN HỌC RỜI RẠC CHO VIỆC KIỂM THỬ

- Tập hợp
- Hàm
- Quan hệ
- Đồ thị

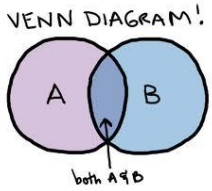
2.1 TẬP HỢP – KHAI BÁO

- Liệt kê phần tử:

  Y = {1812,1813,1814,…,2013,2014}
- Dùng qui tắc:
  - Y = {năm | 1812 ≤năm ≤2014}
  - Y = {năm : 1812 ≤năm ≤2014}
  - Y = {item_number : 80 < item_number <100}
  - Y = {x | x is positive integer}
  - Y = {x | PI(x)}
- Tập rỗng
  - Y = {year : 2014 < year < 1812} = {} = ∅

51

2.1 TẬP HỢP – BIỂU ĐỒ VENN



- Biểu đồ minh họa các quan hệ tập hợp
- Các quan hệ tập hợp:
  - Hợp: A U B = { x | x thuộc A hoặc x thuộc B}
  - Giao: A ∩ B = {x | x thuộc A và x thuộc B}
  - Phần bù: A* = {x | x không thuộc A}
  - Hiệu: A – B = {x | x thuộc A và x không thuộc B}
  - Phần khác nhau (đối xứng): A ⊕ B = (A U B) – (A ∩ B)

52

2.1 TẬP HỢP -QUAN HỆ TẬP HỢP VÀ PHÂN HOẠCH

- **Quan hệ**
  - A là tập con của B: a thuộc A ➔ a thuộc B
  - A tập con chặt của B: A là tập con của B và (B – A) ≠ {}
  - Tập bằng nhau: A = B nếu A là tập con của B và B là tập con của A
- **Phân hoạch**
  - Cho tập B và các tập con A1, A2, An của B, các tập con này là một phân hoạch của B nếu A1 U A2 U .. An = B và Với i ≠ j Ai ∩Aj = {}

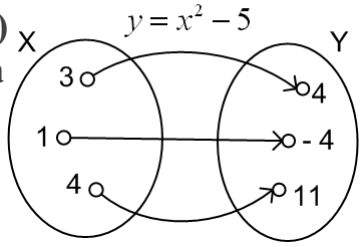
53

2.2 HÀM – ĐỊNH NGHĨA

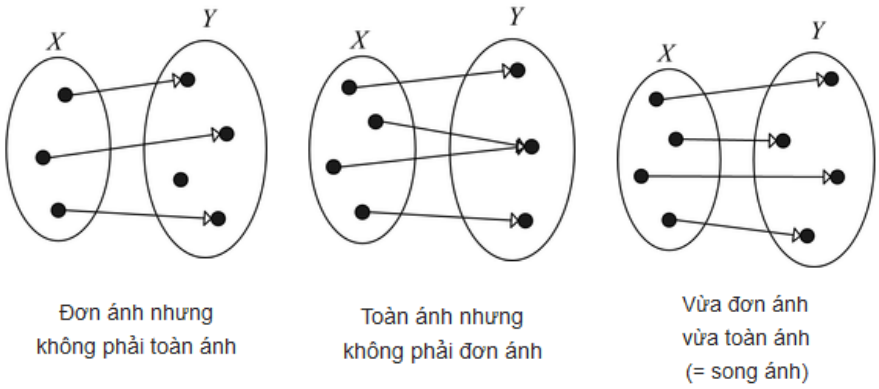
- Cho hai tập A và B, một hàm *f* :A ➔ B là một tập con của tập tích A X B sao cho với mọi a_i, a_j thuộc A, tồn tại b_i, b_j thuộc B sao cho *f*(a_i) = b_i và *f*(a_i) = b_i và b_i khác b_j thì a_i khác a_j
- A là miền xác định (domain) B (range) của

$$f : A \to B$$
$$f \subseteq A \times B$$

$$f : X \to Y$$
$$y = x^2 - 5$$



2.2 HÀM – CÁC DẠNG HÀM SỐ



Đơn ánh nhưng không phải toàn ánh

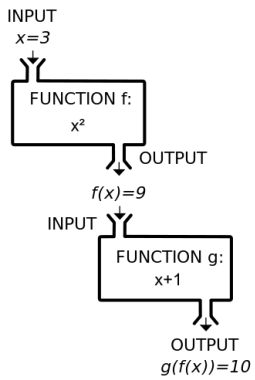
Toàn ánh nhưng không phải đơn ánh

Vừa đơn ánh vừa toàn ánh (= song ánh)

55

2.2 HÀM – HÀM HỢP

- Giả sử có
  - f: A → B
  - g: B → C
- Chúng ta có thể định nghĩa hàm mới w:A → D sao cho $w(a) = (g \circ f)(x) = (g(f(x))$
- Như vậy nếu $f(a) = b$, và $g(b) = c$ thì $w(a) = g(f(a)) = g(b) = c$



56

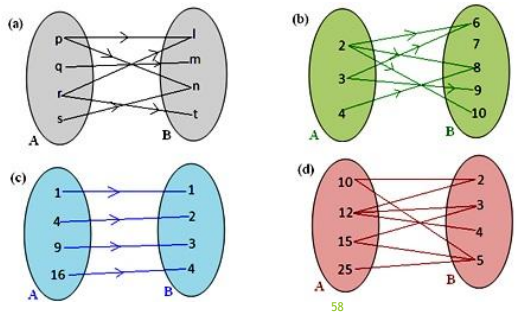
2.3 QUAN HỆ

- Cho hai tập A, B, một quan hệ R là một tập con của tích Đề-các A X B
- Ví dụ: A = {a, b, c, d}; B= {1, 2, 3}. Quan hệ R = { (a,1), (a, 2), (b, 2), (c, 2)}.
- Hàm là một dạng đặc biệt của quan hệ:
  - f(a) là duy nhất
  - R(a) không duy nhất, là tập con của AxB

57

2.3 QUAN HỆ - QUAN HỆ GIỮA CÁC TẬP HỢP

- Quan hệ có lực lượng
- Lực lượng của R có thể là:
  - 1-1
  - nhiều-1
  - 1-nhiều
  - nhiều-nhiều



58

2.3 QUAN HỆ - THAM GIA CỦA QUAN HỆ

- Cho A, B là hai tập hợp, R ⊆ A × B là một quan hệ. Sự tham gia của quan hệ R giữa A và B có thể là:
  - **toàn bộ**: R có mọi phần tử của A
  - **bộ phận**: R không có mọi phần tử của A
  - **lên**: R có mọi phần tử của B
  - **vào**: R không có mọi phần tử của B

59

2.3 QUAN HỆ - QUAN HỆ TRÊN MỘT TẬP HỢP

- Cho A là một tập hợp, R ⊆ A × A là một quan hệ trên A. Có bốn tính chất quan trọng đối với các quan hệ trên A.
- **phản xạ**: nếu và chỉ nếu với mọi a thuộc A <a, a> thuộc R.
- **đối xứng**: nếu và chỉ nếu <a,b> thuộc R thì <b,a> thuộc R
- **phản đối xứng**: nếu <a,b> thuộc R và <b,a> thuộc R thì suy ra a=b
- **bắc cầu**: nếu và chỉ nếu <a,b> thuộc R và <b,c> thuộc R thì <a,c> thuộc R

60

2.3 QUAN HỆ - QUAN HỆ TRÊN MỘT TẬP HỢP

- R là quan hệ **thứ tự** nếu nó là phản xạ, bất đối xứng, và bắc cầu.
- R là quan hệ **tương đương** nếu nó là phản xạ, đối xứng, và bắc cầu.

61

2.4 LOGIC MỆNH ĐỀ

- Một mệnh đề là một câu khẳng định có giá trị đúng (T) hoặc sai (F)
- Mệnh đề đơn giản (ví dụ p, q, r) là một mệnh đề hợp lệ (valid)
- Các phép toán logic dùng để xây dựng các mệnh đề/công thức/biểu thức logic phức tạp hơn từ các mệnh đề đơn giản: hội, tuyển, phủ định, kéo theo,..

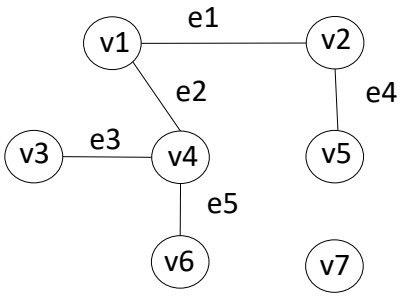
p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \oplus q$	$p \rightarrow q$	$p \leftrightarrow q$
T	T	F	T	T	F	T	T
T	F	F	F	T	T	F	F
F	T	T	F	T	T	T	F
F	F	T	F	F	F	T	T

2.5 ĐỒ THỊ VÔ HƯỚNG - ĐỊNH NGHĨA

- **Một đồ thị G = (V, E) gồm**
  - **Một tập không rỗng, hữu hạn V = {v_1, v_2,.. v_m} các đỉnh**
  - **Một tập cạnh E = {e_1, e_2,.. e_p} trong đó e_k = {v_i, v_j}, với các đỉnh v_i, v_j nào đó thuộc V.**

63

2.5 ĐỒ THỊ VÔ HƯỚNG - VÍ DỤ



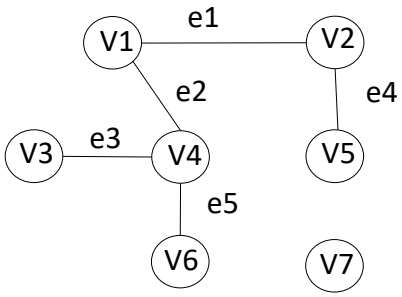
V = {v1, v2, v3, v4, v5, v6, v7}

E = {e1, e2, e3, e4, e5}
= {{v1, v2}, {v1, v4}, {v3, v4}, {v2, v5}, {v4, v6}}

Bậc của một đỉnh: số cạnh gắn với nó.
Ví dụ: deg(v1) = 2, deg(v7) = 0, deg(v4) = 3.

2.5 ĐỒ THỊ VÔ HƯỚNG - MA TRẬN TỚI

- Phần tử tương ứng đỉnh có gắn cạnh có giá trị 1
- Còn lại là 0



	e1	e2	e3	e4	e5
V1	1	1	0	0	0
V2	1	0	0	1	0
V3	0	0	1	0	0
V4	0	1	1	0	1
V5	0	0	0	1	0
V6	0	0	0	0	1
V7	0	0	0	0	0

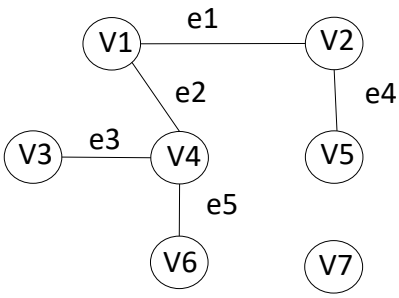
Tổng bậc của tất cả các đỉnh = 2 lần số cạnh. Ký hiệu: Σdeg(v) = 2e

65

http://vi.wikipedia.org/wiki/Ma_tr%E1%BA%ADn_li%C3%AAn_thu%E1%BB%9

2.5 ĐỒ THỊ VÔ HƯỚNG - MA TRẬN KỀ

- Cặp đỉnh có cạnh có giá trị 1
- Còn lại là 0



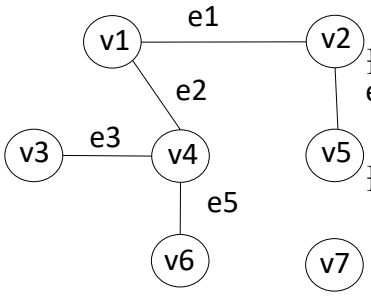
	V1	V2	V3	V4	V5	V6	V7
V1	0	1	0	1	0	0	0
V2	1	0	0	0	1	0	0
V3	0	0	0	1	0	0	0
V4	1	0	1	0	0	1	0
V5	0	1	0	0	0	0	0
V6	0	0	0	1	0	0	0
V7	0	0	0	0	0	0	0

66

http://vi.wikipedia.org/wiki/Ma_tr%E1%BA%ADn_k%E1%BB%81

2.5 ĐỒ THỊ VÔ HƯỚNG - ĐƯỜNG ĐI

- Một đường đi là một dãy các cạnh liên tục
- Có thể mô tả bằng một dãy đỉnh hoặc một dãy cạnh
- Đường đi có đỉnh bắt đầu/nguồn (source) và đỉnh kết thúc/đích (target)



Đường đi từ v1 đến v5:

e1 -> e4

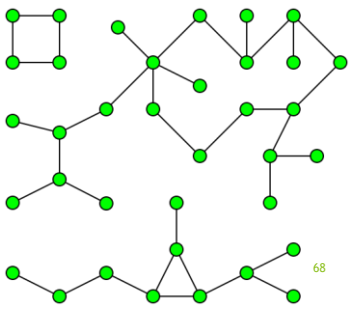
Đường đi từ v6 đến v5:

e5->e2->e1->e4

67

2.5 ĐỒ THỊ VÔ HƯỚNG - LIÊN THÔNG

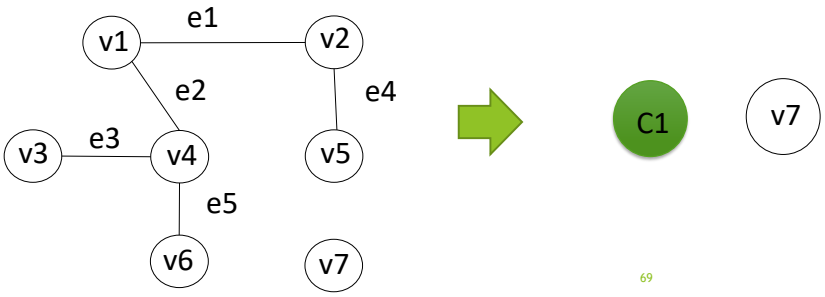
- n_i, n_j là liên thông nếu có đường đi giữa chúng
- Liên thông là quan hệ tương đương trên tập đỉnh của đồ thị
- Liên thông xác định một phân hoạch trên tập đỉnh của đồ thị
- Một thành phần của một đồ thị là tập lớn nhất các đỉnh liên thông



68

2.5 ĐỒ THỊ VÔ HƯỚNG -ĐỒ THỊ RÚT GỌN

- Đồ thị rút gọn của đồ thị G = (V, E) được xây dựng bằng cách thay các thành phần bằng một đỉnh rút gọn



69

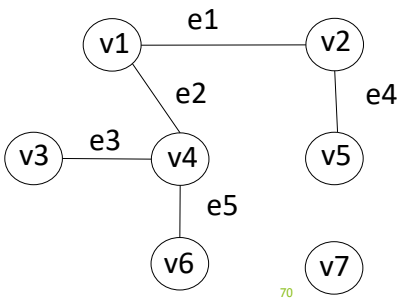
2.5 ĐỒ THỊ VÔ HƯỚNG - ĐỘ PHỨC TẠP

- Độ phức tạp cyclomatic complexity của đồ thị D là

$$V(G) = e - n + p$$

trong đó e, n, p tương ứng là số cạnh, đỉnh, số các thành phần của đồ thị

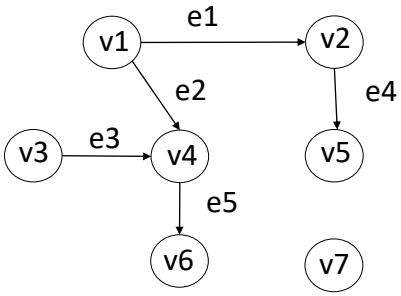
V(G) = 5 -7 +2 = 0



70

2.5 ĐỒ THỊ CÓ HƯỚNG – ĐỊNH NGHĨA

- Đồ thị có hướng D = (V, E) gồm một tập đỉnh V = {n_1, n_2, … n_m}, và một tập cạnh E = {e_1, e_2, e_p}, trong đó mỗi cạnh e_k = <n_i, n_j> là một cặp đỉnh có thứ tự



V = {v1, v2, v3, v4, v5, v6, v7}

E = {e1, 2, e3, e4, e5} =
{<v1, v2>, <v1, v4>, <v3, v4>, <v2, v5>, <v4, v6>}

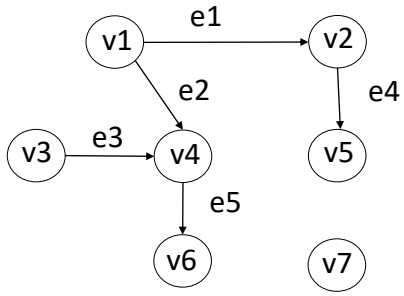
71

2.5 ĐỒ THỊ CÓ HƯỚNG – TÍNH CHẤT

- Bậc vào/ra của một đỉnh là số cạnh đến/đi của đỉnh đó
- Đỉnh có bậc vào/ra bằng 0 là đỉnh nguồn/đích (đầu/cuối)
- Đỉnh có bậc vào và bậc ra đều khác 0 là đỉnh chuyển
- Đỉnh vừa là nguồn và đích là đỉnh cô lập
- Đỉnh vào và đỉnh ra lập thành một biên ngoài của đồ thị

72

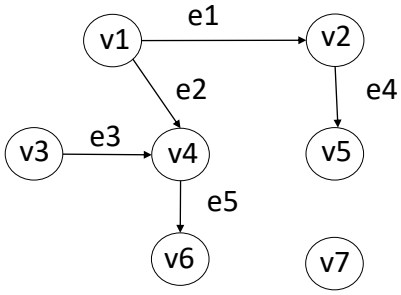
2.5 ĐỒ THỊ CÓ HƯỚNG – TÍNH CHẤT



$$
\begin{array}{llll}
indeg(v_1) & = 0 & outdeg(v_1) & = 2 \\
indeg(v_2) & = 1 & outdeg(v_2) & = 1 \\
indeg(v_3) & = 0 & outdeg(v_3) & = 1 \\
indeg(v_4) & = 2 & outdeg(v_4) & = 1 \\
indeg(v_5) & = 1 & outdeg(v_5) & = 0 \\
indeg(v_6) & = 1 & outdeg(v_6) & = 0 \\
indeg(v_7) & = 0 & outdeg(v_7) & = 0
\end{array}
$$

2.5 ĐỒ THỊ CÓ HƯỚNG – MA TRẬN KỀ

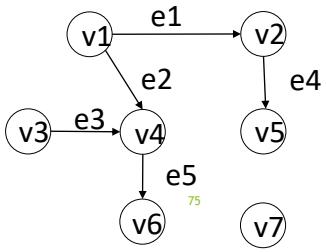
- **Ma trận vuông có kích thước bằng số đỉnh và phần tử ở hàng i, cột j của ma trận là 1 nếu và chỉ nếu có cạnh từ đỉnh i đến đỉnh j, và là 0 ngược lại.**
- **Tổng của hàng/cột là bậc ra/vào**



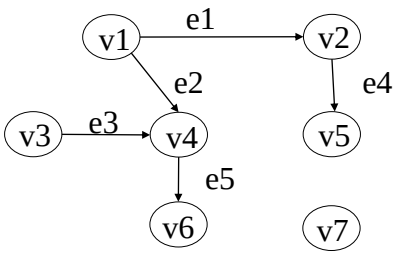
	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	1	0	1	0	0	0
v_2	0	0	0	0	1	0	0
v_3	0	0	0	1	0	0	0
v_4	0	0	1	0	0	1	0
v_5	0	0	0	0	0	0	0
v_6	0	0	0	0	0	0	0
v_7	0	0	0	0	0	0	0

2.5 ĐỒ THỊ CÓ HƯỚNG – ĐƯỜNG ĐI VÀ TỰA ĐƯỜNG ĐI

- Đường đi có hướng là dãy các cạnh mà đỉnh cuối của cạnh trước là đỉnh đầu của cạnh tiếp theo
- Chu trình là đường đi có đỉnh đầu và cuối trùng nhau
- Tựa đường đi có hướng là dãy cạnh trong đó có cặp cạnh có chung đỉnh đầu hoặc chung đỉnh cuối
- Ví dụ:
  - đường đi từ v1 đến v6,
  - tựa đường đi từ v1 và v3; v2 và v4; v5 và v6



75



MA TRẬN TỚI

- Ma trận tới của đồ thị có m đỉnh là ma trận m x m trong đó phần tử (i,j) là 1 khi có đường đi từ i đến j

	v1	v2	v3	v4	v5	v6	v7
V1	0	1	0	1	1	1	0
V2	0	0	0	0	1	0	0
V3	0	0	0	1	0	1	0
V4	0	0	0	0	0	1	0
V5	0	0	0	0	0	0	0
V6	0	0	0	0	0	0	0
v7	0	0	0	0	0	0	0

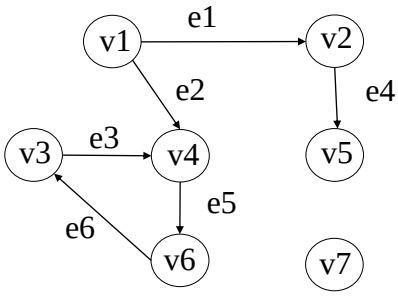
76

2.5 ĐỒ THỊ CÓ HƯỚNG – N – LIÊN THÔNG

- Hai đỉnh n_i và n_j trong một đồ thị có hướng là
  - **0-liên thông**: nếu và chỉ nếu không có đường đi từ n_i đến n_j
  - **1-liên thông**: nếu và chỉ nếu có một tựa đường và không có đường giữa n_i, n_j
  - **2-liên thông**: nếu và chỉ nếu có một đường đi giữa n_i, n_j
  - **3-liên thông**: nếu và chỉ nếu có một đường đi từ n_i đến n_j và một đường đi từ n_j đến n_i
- Một thành phần mạnh của một đồ thị có hướng là tập đỉnh 3-liên thông lớn nhất

77

2.5 ĐỒ THỊ CÓ HƯỚNG – N – LIÊN THÔNG

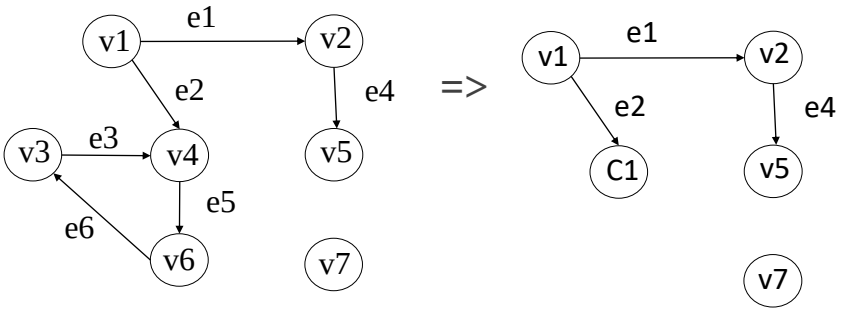


v1 và v7 là 0-liên thông
v2 và v6 là 1-liên thông
v1 và v6 là 2-liên thông
v3 và v6 là 3-liên thông

78

2.5 ĐỒ THỊ CÓ HƯỚNG – ĐỒ THỊ RÚT GỌN

- Là đồ thị với các thành phần liên thông mạnh được thay bằng các đỉnh đơn và các cạnh được giữ nếu có các cạnh giữa hai đỉnh bất kỳ thuộc thành phần liên thông.



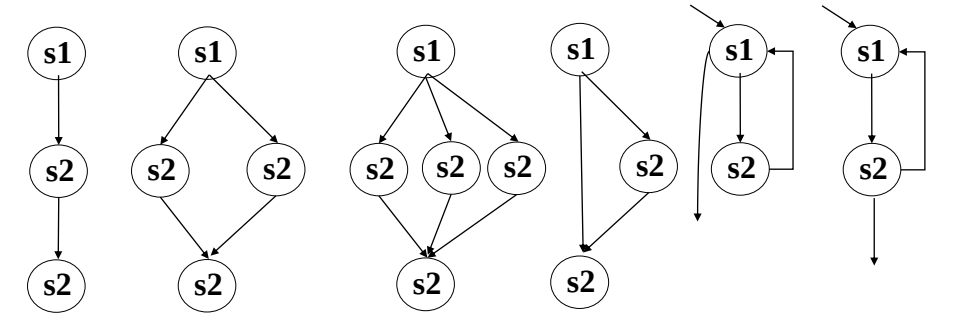
79

2.5 ĐỒ THỊ CHƯƠNG TRÌNH

- Chương trình viết bằng ngôn ngữ mệnh lệnh có đồ thị chương trình là đồ thị có hướng với đỉnh là các lệnh hoặc đoạn lệnh liên tiếp và cạnh là các luồng điều khiển
- Luồng điều khiển của chương trình mệnh lệnh được xác định bằng các cấu trúc tuần tự, lặp và rẽ nhánh

80

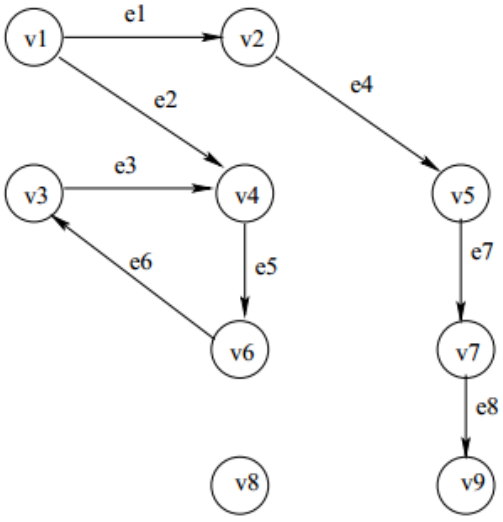
2.5 ĐỒ THỊ CHƯƠNG TRÌNH



81

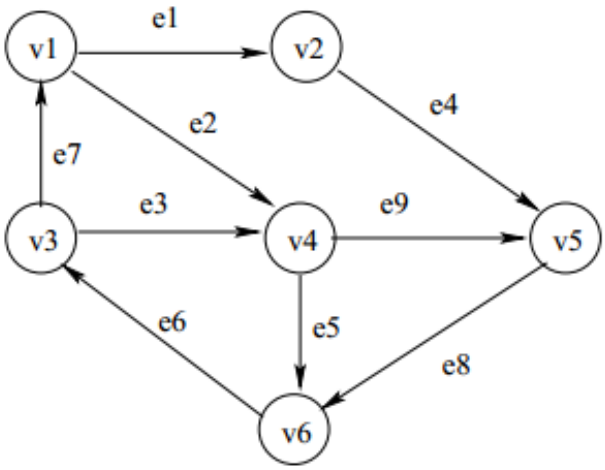
BÀI TẬP

1. Tính độ phức tạp chu trình của đồ thị sau



BÀI TẬP

2.Tính độ phức tạp chu trình của đồ thị sau



84

CHƯƠNG 3. KỸ THUẬT KIỂM THỬ PHẦN MỀM

- Kỹ thuật **kiểm thử chức năng** / **kiểm thử hàm** (functional testing) / **kiểm thử hộp đen**
  - Dựa trên việc khảo sát đặc tả của chương trình
- Kỹ thuật **kiểm thử cấu trúc** (structural testing) / **kiểm thử hộp trắng**
  - Dựa trên việc khảo sát mã nguồn của chương trình

85

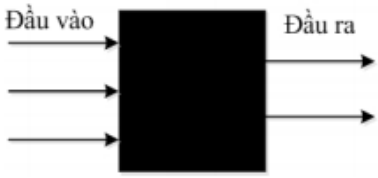
3.1. KỸ THUẬT KIỂM THỬ CHỨC NĂNG

- **Kiểm thử hàm** (functional testing) là các hoạt động kiểm tra chương trình dựa trên tài liệu mô tả chức năng, yêu cầu phần mềm, hay còn gọi là đặc tả chức năng (functional specification).
- Dữ liệu kiểm thử xuất phát từ đặc tả:
  - *Kiểm thử hệ thống: Đặc tả yêu cầu*
  - *Kiểm thử tích hợp: Đặc tả thiết kế*
  - *Kiểm thử đơn vị: Đặc tả chi tiết mô-đun*

86

3.1. KỸ THUẬT KIỂM THỬ CHỨC NĂNG

- Tester xem phần mềm như là một hộp đen:
  - *Không quan tâm đến cấu trúc và hành vi bên trong phần mềm*
  - *Chỉ quan tâm đến các "hoạt động bề ngoài" của phần mềm*



Hình 1.5: Một hộp đen kỹ thuật.

87

3.1. KỸ THUẬT KIỂM THỬ CHỨC NĂNG

- Tester cố gắng tìm ra các LỖI sau:
  - *Chức năng THIẾU hoặc KHÔNG ĐÚNG*
  - *Lỗi giao diện*
  - *Lỗi cấu trúc dữ liệu*
  - *Lỗi truy cập CSDL bên ngoài*
  - *Lỗi thi hành*
  - *Lỗi khởi tạo/kết thúc*
  - *…*

88

3.1. KỸ THUẬT KIỂM THỬ CHỨC NĂNG

- Kiểm thử giá trị biên
- Phân hoạch tương đương
- Bảng quyết định

89

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN

- Các chương trình có thể coi là một hàm (toán học)
  - Các đầu vào chương trình là miền xác định của hàm
  - Các đầu ra là miền giá trị của hàm
- Phân tích giá trị biên (boundary value analysis - BVA) là kỹ thuật kiểm thử hàm phổ biến nhất.
- Mục tiêu của kiểm thử hàm là sử dung kiến thức về hàm để xác định các ca kiểm thử
  - Trước kia chủ yếu tập trung vào miền xác định, nhưng nay đã dựa trên cả miền giá trị của hàm để xác định ca kiểm thử

90

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN

- Phân tích giá trị biên tập trung vào biên của miền xác định để xây dựng ca kiểm thử.
- Lý do là lỗi thường xảy ra ở gần các giá trị biên này.
- Chương trình viết bằng ngôn ngữ không có kiểm tra kiểu mạnh càng cần kiểm thử giá trị biên
  - Javascript, php, Visual Basic

91

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – GIẢ THUYẾT KHIẾM KHUYẾT ĐƠN

- Phân tích giá trị biên dựa trên nguyên lý giả định khuyếm khuyết đơn:
  - "Hỏng hóc xảy ra hiếm khi do hai (hoặc hơn) khiếm khuyết cùng xảy ra"
- Do đó các ca kiểm thử theo phương pháp này được tạo bằng việc lấy các giá trị bình thường của các chiều/biên rồi lần thay mỗi chiều bằng các giá trị cực trị như trên.

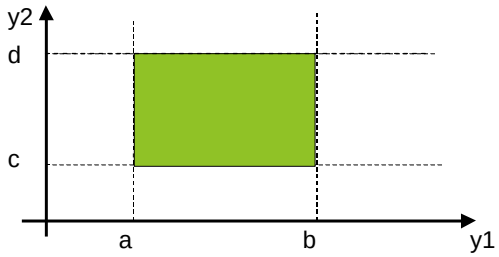
92

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN

- Từ đây chúng ta giả sử có chương trình P nhận hai biến đầu vào là y1 và y2 thỏa mãn $a \le y1 \le b$ and $c \le y2 \le d$

  P(y1, y2) where a <= y1 <= b, c <= y2 <= d

- Chương trình nhận n đầu vào sẽ có không gian đầu vào n chiều



93

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – CHỌN GIÁ TRỊ

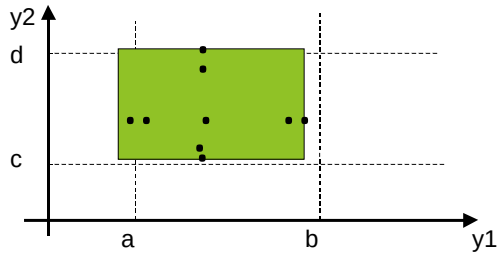
- Phân tích giá trị biên sẽ chọn các giá trị:
  - Giá trị nhỏ nhất
  - Ngay trên giá trị nhỏ nhất
  - Một giá trị bình thường
  - Ngay dưới giá trị lớn nhất
  - Giá trị lớn nhất

Ví dụ:

a <= y1 <=b thì sẽ chọn a, a+1, a+b/2, b-1, b.

94

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – CÁC CA KIỂM THỬ THEO PHÂN TÍCH GIÁ TRỊ BIÊN



$T = \{ \langle y_{1nom}, y_{2min} \rangle, \langle y_{1nom}, y_{2min+} \rangle, \langle y_{1nom}, y_{2nom} \rangle, \langle y_{1nom}, y_{2max-} \rangle, \langle y_{1nom}, y_{2max} \rangle, \langle y_{1min}, y_{2nom} \rangle, \langle y_{1min+}, y_{2nom} \rangle, \langle y_{1max-}, y_{2nom} \rangle, \langle y_{1max}, y_{2nom} \rangle \}$

95

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – BÀI TOÁN TRIANGLE

- Bài toán tam giác nhận ba số nguyên làm các dữ liệu đầu vào; các dữ liệu này là số đo các cạnh của một tam giác. Đầu ra của chương trình là loại của tam giác xác định bởi ba cạnh ứng với các số đo này: tam giác đều, tam giác cân, tam giác thường, hoặc không là tam giác.
- Ta sẽ dùng các từ tiếng Anh làm dữ liệu đầu ra tương ứng cho các loại này như lấy từ ví dụ nguyên thủy: Equilateral, Isosceles, Scalene, hoặc NotATriangle. Bài toán này đôi khi được mở rộng với đầu ra thứ năm là tam giác vuông (right triangle**).**

96

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – CÁC CA KIỂM THỬ CHO BÀI TOÁN TRIANGLE

Case #	a	b	c	Expected Output
1	100	100	1	Isosceles
2	100	100	2	Isosceles
3	100	100	100	Equilateral
4	100	100	199	Isosceles
5	100	100	200	Not a Triangle
6	100	1	100	Isosceles
7	100	2	100	Isosceles
8	100	100	100	Equilateral
9	100	199	100	Isosceles
10	100	200	100	Not a Triangle
11	1	100	100	Isosceles
12	2	100	100	Isosceles
13	100	100	100	Equilateral
14	199	100	100	Isosceles
15	200	100	100	Not a Triangle

97

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – TỔNG QUÁT HÓA

- Có hai cách tổng quát hóa :
  - Theo số biến, sẽ có (4n +1) ca kiểm thử cho n biến
  - Theo loại khoảng của biến
    - Phụ thuộc ngôn ngữ lập trình
    - Tính rời rạc của biến
    - Tính rời rạc không bị chặn (không có cận trên và cận dưới rõ ràng)
    - Biến logic

98

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – ƯU ĐIỂM

- Đơn giản
- Có thể tự động hóa việc sinh các ca kiểm thử khi đã xác định được các biên của các biến.

99

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – HẠN CHẾ

- BVA hiệu quả với các chương trình có các đầu vào độc lập nhau và biểu diễn đại lượng vật lý bị chặn
- BVA lấy các ca kiểm thử mà không tính đến chức năng của hàm, hay ý nghĩa của các biến

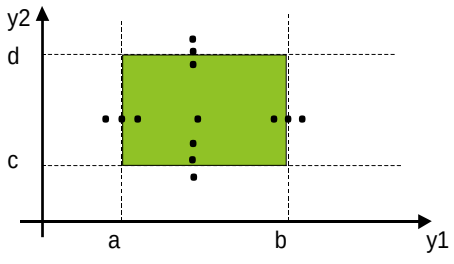
100

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – KIỂM THỬ BIÊN MẠNH

- Kiểm thử biên mạnh (robustness testing) là một mở rộng đơn giản của BVA
- Ngoài năm giá trị biên bổ sung thêm hai giá trị ngoài biên:
  - Giá trị ngay trên giá trị cực đại (max+) và
  - Giá trị ngay dưới giá trị cực tiểu (min-).
- Mục đích chính là xem chương trình có kiểm tra giá trị hợp lệ của đầu vào không.

101

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – KIỂM THỬ BIÊN MẠNH



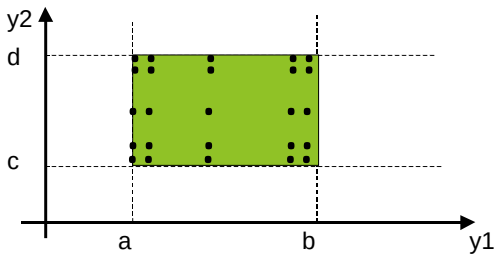
102

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – KIỂM THỬ GIÁ TRỊ TỔ HỢP BIÊN

- Điều gì xảy ra khi nhiều hơn một biến nhận các giá trị (gần) cực trị?
- Khi các biến có tương tác với nhau thì cần kiểm tra các bộ giá trị kết hợp các cực trị này
- Có thể kết hợp với kiểm thử mạnh để có bộ kiểm thử trường hợp xấu nhất mạnh

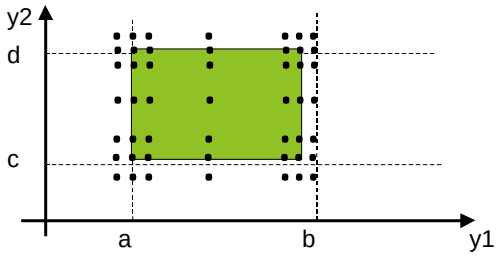
103

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – KIỂM THỬ GIÁ TRỊ TỔ HỢP BIÊN



104

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – KIỂM THỬ GIÁ TRỊ TỔ HỢP BIÊN MẠNH



105

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – KIỂM THỬ GIÁ TRỊ ĐẶC BIỆT

- Kiểm thử giá trị đặc biệt là phương pháp được thực hiện nhiều nhất trên thực tế, nó cũng trực quan nhất, và không có dạng cố định nhất
- Sử dụng kỹ nghệ và kiến thức miền ứng dụng để phán đoán và đưa ra ca kiểm thử
- Mặc dù mang tính chủ quan cao, đây vẫn là phương pháp hiệu quả để phát hiện khiếm khuyết của chương trình

106

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – HƯỚNG DẪN ÁP DỤNG

- Trừ kiểm thử giá trị đặc biệt, kiểm thử giá trị biên là phương pháp thô sơ nhất
- Cân nhắc các tình huống để chọn phương pháp phù hợp:
  - Các biến có độc lập hay phụ thuộc nhau không
  - Có cần các giá trị mạnh hay giá trị thường
  - Lỗi thường do 1 khiếm khuyết hay nhiều?
- Có thể áp dụng BVA cho miền giá trị (đầu ra), hay các biến khác trong chương trình như biến đếm vòng lặp, biến trung gian, chỉ số mảng, con trỏ,..

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – BÀI TOÁN CHIA HOA HỒNG

- Một người bán hàng chuyên bán khóa, báng và nòng súng trường cho một cửa hàng.
- Giá của Khóa = $ 45, báng = $ 30, và nòng = $ 25
- Mỗi người bán hàng phải bán ít nhất một bộ đầy đủ mỗi tháng (100 đô)
- Người bán giỏi nhất bán được 70 khóa, 80 báng, 90 nòng một tháng

3.1.1. KIỂM THỬ GIÁ TRỊ BIÊN – BÀI TOÁN CHIA HOA HỒNG

- Hàng tháng mỗi người bán sẽ gửi báo cáo về cho cửa hàng với tổng số hàng bán được cho mỗi thị trấn anh ta đến
- Số thành phố đến được mỗi tháng là từ 1 đến 10.
- Người bán nhận được:
  - 10% nếu số tiền bán được <= 1000 đô,
  - 15% trên 800 đô tiếp theo, và
  - 20% của số tiền vượt quá 1800.

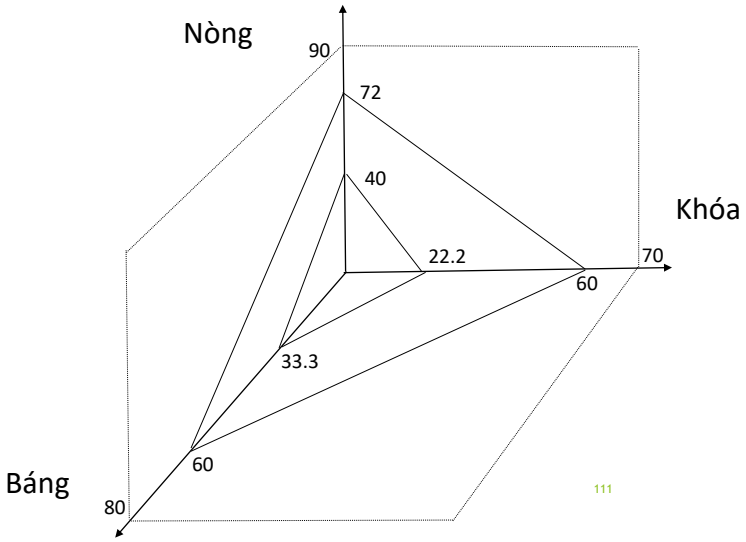
109

3.1.1. BÀI TOÁN CHIA HOA HỒNG – CÁC CA KIỂM THỬ THEO MIỀN GIÁ TRỊ

Case #	Locks	Stocks	Barrels	Sales	Comm.	Comments
1	1	1	1	100	10	min
2	10	10	9	975	97.5	border-
3	10	9	10	970	97	border-
4	9	10	10	955	95.5	border-
5	**10**	**10**	**10**	**1000**	**100**	**border**
6	10	10	11	1025	103.75	border+
7	10	11	10	1030	104.5	border+
8	11	10	10	1045	106.75	border+

110

3.1.1. BÀI TOÁN CHIA HOA HỒNG – CÁC CA KIỂM THỬ THEO KHOẢNG ĐẦU RA



111

3.1.1. BÀI TOÁN CHIA HOA HỒNG – CÁC CA KIỂM THỬ GIÁ TRỊ ĐẶC BIỆT

TT	Khóa	Báng	Nòng	Số bán	Hoa hồng	Ghi chú
1	10	11	9	1005	100.75	border+
2	18	17	19	1795	219.25	border+
3	18	19	17	1805	221	border+

112

3.1.1. BÀI TOÁN CHIA HOA HỒNG – HÀM TÍNH NGÀY KẾ TIẾP

- NextDate là một hàm có ba biến biểu diễn ngày, tháng và năm là day, month và year. Hàm này trả về ngày kế tiếp của ngày đầu vào. Các biến day, month, year có các giá trị số thỏa mãn các ràng buộc: $1 \leq day \leq 31$, $1 \leq month \leq 12$, $1812 \leq year \leq 2012$.

→ Áp dụng kiểm thử giá trị biên cho hàm này?

113

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG

- **Kiểm thử lớp tương đương (ECT)**
  - **Kiểm thử lớp tương đương yếu**
  - **Kiểm thử lớp tương đương mạnh**
  - **Kiểm thử lớp tương đương đơn giản**

114

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – VÍ DỤ

- **Ví dụ: Xét chương trình đọc số nguyên có giá trị hợp lệ trong đoạn 0..1000 (tt)**
- **Như vậy dữ liệu thử nào sau đây có khả năng phát hiện lỗi cao hơn:**
  - ***-7, 396, 1800* ?**
  - ***140, 1680, 360?***

115

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – VAI TRÒ

- **Các lớp tương đương tạo thành một phân hoạch của miền dữ liệu**
  - **Hợp của tất cả các lớp bằng miền đầu vào**
    - **Cảm giác đã kiểm thử hết**
  - **Hai lớp bất kỳ không giao nhau**
    - **Không dư thừa**

116

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – VAI TRÒ

- **Việc kiểm thử tất các đầu vào của PM là KHÔNG THỂ**
- **Nên giới hạn một tập con tất cả các trường hợp đầu vào có thể có**
- **Mục đích: lựa chọn một tập con đúng**
  - ***Có xác suất cao nhất phát hiện hầu hết các lỗi***

117

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – Ý TƯỞNG

- **Ý tưởng của ECT là chỉ kiểm thử với một phần tử của mỗi miền tương đương**
  - **Giảm rất nhiều dư thừa tiềm tàng nếu các lớp tương đương được chọn hợp lý**
- **Mấu chốt là làm sao chọn được quan hệ tương đương để từ đó xác định được các lớp tương đương (phân hoạch)**

118

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – CHỌN PHÂN HOẠCH

- Thường là "thủ công" (craft):
  - Không dựa trên mã nguồn, chỉ dựa trên đặc tả
  - Cần hiểu biết về miền xác định, thường không thể xác định dựa vào đặc tả thiết kế giao diện
  - Phải hiểu đầu vào phụ thuộc nhau như thế nào

119

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – CHỌN PHÂN HOẠCH

- Thiết kế DL thử bằng phân hoạch tương đương theo 2 bước như sau:
  - *Phân hoạch các miền đầu vào/ra thành các lớp tương đương*
  - *Thiết kế các trường hợp kiểm thử đại diện cho mỗi lớp.*

120

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – VÍ DỤ

- Xét chương trình P có ba biến đầu vào: a, b và c với các miền xác định là A, B, and C.
- Phân hoạch của các miền này giả sử là:

$$A = A1 \cup A2 \cup A3$$
$$B = B1 \cup B2 \cup B3 \cup B4$$
$$C = C1 \cup C2$$

121

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – VÍ DỤ (tt)

- Gọi ai thuộc Ai là một phần tử đại diện của lớp
  - Ví dụ lấy phần tử giữa của 1 khoảng
- Tương tự có bi và ci.
- Các ca kiểm thử sẽ được xây dựng từ các phần tử đại diện này
- Ý tưởng ở đây là phần tử đại diện này cũng tốt như các phần tử khác ở trong cùng lớp tương đương đó

122

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – ECT YẾU

- ECT yếu chỉ lấy tất cả các phần tử đại diện ít nhất một lần
- Số ca kiểm thử tối thiểu sẽ bằng số lớp của phân hoạch có nhiều tập con nhất
  - Trong ví dụ trước là 4

#	a	b	c
WE1	a_1	b_1	c_1
WE2	a_2	b_2	c_2
WE3	a_3	b_3	c_1
WE4	a_1	b_4	c_2

123

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – ECT MẠNH

- ECT mạnh dựa trên tích Đề-các của các lớp con
- Với ví dụ trước ta có: 3 * 4 * 2 = 24 ca kiểm thử
- Cách này xét đến tất cả các tương tác của các giá trị đại diện

124

#	a	b	c
SE1	a1	b1	c1
SE2	a1	b1	c2
SE3	a1	b2	c1
SE4	a1	b2	c1
SE5	a1	b3	c1
SE6	a1	b3	c2
SE7	a1	b4	c1
SE8	a1	b4	c2
SE9	a2	b1	c1
SE10	a2	b1	c2
SE11	a2	b2	c1
SE12	a2	b2	c2

#	a	b	c
SE13	a2	b3	c1
SE14	a2	b3	c2
SE15	a2	b4	c1
SE16	a2	b4	c2
SE17	a3	b1	c1
SE18	a3	b1	c2
SE19	a3	b2	c1
SE20	a3	b2	c2
SE21	a3	b3	c1
SE22	a3	b3	c2
SE23	a3	b4	c1
SE24	a3	b4	c2

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – ECT ĐƠN GIẢN

- Chỉ phân biệt lớp giá trị hợp lệ và không hợp lệ
- Ví dụ nếu A là khoảng 1 đến 200 các số nguyên thì miền hợp lệ là các giá trị từ 1 đến 200, miền không hợp lệ gồm hai miền. Miền thứ nhất là các giá trị từ 0 trở xuống. Miền còn lại là các giá trị từ 201 trở lên.
- Sau khi đã có các lớp tương đương theo cách đơn giản này chiến lược kiểm thử có thể áp dụng tương tự kiểm thử tương đương mạnh và yếu.

127

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – BÀI TOÁN TRIANGLE DỰA TRÊN KẾT QUẢ ĐẦU RA

- Miền đầu ra có các giá trị:
  - Không là tam giác
  - Tam giác đều
  - Tam giác cân
  - Tam giác thường

128

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – BÀI TOÁN TRIANGLE DỰA TRÊN KẾT QUẢ ĐẦU RA

- Chúng ta sử dụng chúng để xác định lớp tương đương
  - R1= {<a, b, c> | ba cạnh a, b, c tạo thành tam giác cân}
  - R2= {<a, b, c> | ba cạnh a, b, c tạo thành tam giác đều}
  - R3= {< a, b, c> | ba cạnh a, b, c tạo thành tam giác thường}
  - R4= {a, b, c> | ba cạnh a, b, c không tạo thành tam giác}

129

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – BÀI TOÁN TRIANGLE

TT	a	b	c	Kết quả mong đợi
OE1	5	5	5	Đều
OE2	2	2	3	Cân
OE3	3	4	5	Thường
OE4	4	1	2	Không là tam giác

130

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – BÀI TOÁN TRIANGLE DỰA TRÊN KẾT QUẢ ĐẦU VÀO

- Dựa trên đầu vào sẽ cho tập ca kiểm thử lớn hơn.
- Các lớp tương đương:
  - D1= {<a,b,c> | a=b=c}
  - D2= {<a,b,c> | a=b, a≠c}
  - D3= {<a,b,c> | a=c, a≠b}
  - D4= {<a,b,c> | b=c, a≠b}
  - D5= {<a,b,c> | a≠b, a≠c, b≠c}
  - D6= {<a, b, c> | a ≥ b+c}
  - D7= {<a, b, c> | b ≥ a+c}
  - D8= {<a, b, c> | c ≥ a+b}

131

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – BÀI TOÁN TRIANGLE DỰA TRÊN KẾT QUẢ ĐẦU VÀO

- Có thể thêm các lớp để loại trừ ba cạnh không tạo thành tam giác
  - D6= {<a, b, c> | a ≥ b+c}
  - D7= {<a, b, c> | b ≥ a+c}
  - D8= {<a, b, c> | c ≥ a+b}
- D6 có thể tách tiếp thành
  - D6'= {<a, b, c> | a = b+c} và D6"= {<a, b, c> | a > b+c}

132

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – BÀI TOÁN NEXTDATE DỰA TRÊN ĐẦU VÀO

- NextDate là hàm của ba biến với miền xác định:
  - $1 \leq month \leq 12$
  - $1 \leq day \leq 31$
  - $1812 \leq year \leq 2012$

133

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – CÁC CA ECT TRUYỀN THỐNG

- Các lớp dữ liệu hợp lệ:
  - M1 = {month | $1 \leq month \leq 12$}
  - D1 = {day | $1 \leq day \leq 31$}
  - Y1 = {year | $1812 \leq year \leq 2012$}

- Các lớp dữ liệu không hợp lệ:
  - M2 = {month | month < 1}
  - M3 = {month | month > 12}
  - D2 = {day | day < 1}
  - D3 = {day | day > 31}
  - Y2 = {year | year < 1812}
  - Y3 = {year | year > 2012}

134

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – CÁC CA ECT TRUYỀN THỐNG

	Month	Day	Year	KQ mong đợi
TE1	6	15	1912	6/16/1912
TE2	-1	15	1912	Invalid
TE3	13	15	1912	Invalid
TE4	6	-1	1912	Invalid
TE5	6	32	1912	Invalid
TE6	6	15	1811	Invalid
TE7	6	15	2013	Invalid

135

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – MỘT PHÂN HOẠCH MỊN HƠN

- **Nếu chọn quan hệ tương đương khéo hơn ta có các lớp tương đương có ích hơn**
  - **M1= {month | tháng có 30 ngày }**
  - **M2= {month | tháng có 31 ngày}**
  - **M3= {month | tháng Hai}**
  - **D1= {day | 1 ≤ day ≤ 28}**
  - **D2= {day | day = 29}**
  - **D3= {day | day = 30}**
  - **D4= {day | day=31}**
  - **Y1= {year | year = 1900}**
  - **Y2= {year | 1812 ≤ year ≤ 2012**

  AND year ≠ 1900 AND (0 = year mod 4)}
  - **Y3= {year | 1812 ≤ year ≤ 2012 AND (0 ≠ year mod 4)}**

136

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – CÁC CA ECT MẠNH

CASE ID	Month	Day	Year	Output
SE1	6	14	1900	6/15/1900
SE2	6	14	1912	6/15/1912
SE3	6	14	1913	6/15/1913
SE4	6	29	1900	6/30/1900
SE5	6	29	1912	6/30/1912
SE6	6	29	1913	6/30/1913
SE7	6	30	1900	7/1/1900
SE8	6	30	1912	7/1/1912
SE9	6	30	1913	7/1/1913
SE10	6	31	1900	ERROR
SE11	6	31	1912	ERROR
SE12	6	31	1913	ERROR
SE13	7	14	1900	7/15/1900
SE14	7	14	1912	7/15/1912
SE15	7	14	1913	7/15/1913
SE16	7	29	1900	7/30/1900
SE17	7	29	1912	7/30/1912
SE18	7	29	1913	7/30/1913

137

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – CÁC CA ECT MẠNH

CASE ID	Month	Day	Year	Output
SE19	7	30	1900	7/31/1900
SE20	7	30	1912	7/31/1912
SE21	7	30	1913	7/31/1913
SE22	7	31	1900	8/1/1900
SE23	7	31	1912	8/1/1912
SE24	7	31	1913	8/1/1913
SE25	2	14	1900	2/15/1900
SE26	2	14	1912	2/15/1912
SE27	2	14	1913	2/15/1913
SE28	2	29	1900	ERROR
SE29	2	29	1912	3/1/1912
SE30	2	29	1913	ERROR
SE31	2	30	1900	ERROR
SE132	2	30	1912	ERROR
SE33	2	30	1913	ERROR
SE34	2	31	1900	ERROR
SE35	2	31	1912	ERROR
SE36	2	31	1913	ERROR

138

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – CÂN NHẮC SỬ DỤNG

- ECT đơn giản khá yếu, kém hơn ETC yếu, bản thân đã kém hơn ETC mạnh
- Nếu cần kiểm tra dữ liệu không hợp lệ thì cần thêm các lớp tương đương ngoài khoảng xác định
- ECT phù hợp với dữ liệu đầu vào là khoảng hoặc tập các giá trị rời rạc

139

3.1.2. KIỂM THỬ LỚP TƯƠNG ĐƯƠNG – CÂN NHẮC SỬ DỤNG (tt)

- Chức năng của hàm sẽ giúp xác định các lớp tương đương
- Tương đương mạnh dựa trên giả định là các biến độc lập, nếu không sẽ có những ca kiểm thử "lỗi"
- Có thể kết hợp với giá trị biên
  - Sử dụng lại các khoảng đã xác định
  - Không xét các phần tử ở biên lớp tương đương
  - Cần mở rộng ECT để có các yêu cầu như BVA

140

BÀI TẬP

- Hãy áp dụng kỹ thuật kiểm thử lớp tương đương cho miền dữ liệu đầu vào và đầu ra của chương trình giải phương trình bậc hai? Biết a, b, c thuộc [0,100]

141

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH

- Yêu cầu chức năng có thể mô tả bằng bảng quyết định (DT)
- DT là một cách chính xác và gọn để mô tả logic phức tạp
  - Gắn các điều kiện với các hành động tương ứng
  - Giống lệnh if-then-else và switch-case
- DT có thể liên kết nhiều điều kiện độc lập với vài hành động một cách dễ hiểu
  - Khác các cấu trúc điều khiển trong các ngôn ngữ lập trình

142

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – VÍ DỤ

Khắc phục sự cố máy in

Điều kiện	**Máy in không in**	**Y**	**Y**	**Y**	**Y**	**N**	**N**	**N**	**N**
	Đèn đỏ nhấp nháy	**Y**	**Y**	**N**	**N**	**Y**	**Y**	**N**	**N**
	Không nhận ra máy in	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**
Hành động	**Kiểm tra dây nguồn**			**X**					
	Kiểm tra dây tín hiệu	**X**		**X**					
	Kiểm tra phần mềm in đã cài đúng	**X**		**X**		**X**		**X**	
	Kiểm tra/thay mực	**X**	**X**			**X**	**X**		
	Kiểm tra kẹt giấy		**X**		**X**				

143

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – SỬ DỤNG BẢNG QUYẾT ĐỊNH

- Để quan sát tất cả các điều kiện dễ dàng
- Có thể dùng để
  - Mô tả logic phức tạp
  - Sinh ca kiểm thử, còn gọi là *kiểm thử dựa trên logic*
- Kiểm thử dựa trên logic được xem là:
  - Kiểm thử cấu trúc khi áp dụng cho các cấu trúc chương trình. Ví dụ luồng điều khiển
  - Kiểm thử hàm khi áp dụng cho đặc tả.

144

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – CẤU TRÚC BẢNG QUYẾT ĐỊNH

1. Các điều kiện	2. Các giá trị điều kiện
3. Hành động	4. Xảy ra hay không

1. Mỗi điều kiện tương ứng với một biến, một quan hệ, hoặc một mệnh đề (predicate)
2. Các giá trị của điều kiện
   – Chỉ là True/False – Bảng quyết định hạn chế
   – Một số giá trị – Bảng quyết định mở rộng
   – Giá trị không quan tâm
3. Mỗi hành động là một thủ tục hoặc thao tác phải thực hiện
4. Đánh dấu hành động có/không xảy ra

145

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – VÍ DỤ BẢNG QUYẾT ĐỊNH TÍNH LƯƠNG

	Conditions/ Courses of Action	Rules					
		1	2	3	4	5	6
Condition Stubs	Employee type	S	H	S	H	S	H
	Hours worked	<40	<40	40	40	>40	>40
Action Stubs	Pay base salary	X		X		X	
	Calculate hourly wage		X		X		X
	Calculate overtime						X
	Produce Absence Report		X				

146

Cách tính lương

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – PHƯƠNG PHÁP XÂY DỰNG BẢNG

1. Xác định các điều kiện và giá trị của chúng
2. Xác định số luật tối đa
3. Xác định các hành động
4. Đánh số các luật nếu cần
5. Đánh số các hành động thích hợp cho mỗi luật
6. Kiểm tra chính sách
7. Đơn giản hóa các luật (gộp cột)

147

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – SỬ DỤNG BẢNG QUYẾT ĐỊNH

- Bảng thích hợp khi:
  - Đặc tả có thể chuyển về dạng bảng
  - Thứ tự các hành động xảy ra không quan trọng
  - Thứ tự các luật không ảnh hưởng đến hành động
  - Khi một luật thỏa mãn và được chọn thì không cần xét luật khác
- Các hạn chế trên không ảnh hưởng đến việc sử dụng bảng
  - Trong hầu hết các ứng dụng thứ tự các mệnh đề được xét là không quan trọng

148

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – MỘT SỐ VẤN ĐỀ CẦN LƯU Ý

- Trước khi sử dụng bảng cần đảm bảo:
  - Các luật phải đầy đủ
    - Có mọi tổ hợp
  - Các luật phải nhất quán
    - Mọi tổ hợp giá trị chân lý chỉ gây ra một hoặc một tập hành động

149

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – THIẾT KẾ CA KIỂM THỬ

- Khi đặc tả đã được kiểm tra, mục tiêu là chứng tỏ chương trình thực hiện các hành động đúng cho mọi tổ hợp các giá trị của mệnh đề
  - Nếu có k luật và n mệnh đề đúng/sai, thì có **ít nhất k trường hợp** và **nhiều nhất là 2^n trường hợp** phải xét.
- Có thể dựa trên các luật chưa mở rộng hoặc các luật đã mở rộng với 2^n ca
  - Xác định đầu vào cho mỗi ca

150

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – THIẾT KẾ CA KIỂM THỬ (tt)

- Để xác định ca kiểm thử, chúng ta chuyển các điều kiện thành đầu vào, hành động thành đầu ra.
- Một số điều kiện sẽ tham chiếu đến các lớp tương đương đầu vào, và hành động tham chiếu đến các phần xử lý chức năng chính của cột đang xét.
- Các luật được chuyển thành các ca kiểm thử

151

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – BẢNG QUYẾT ĐỊNH TRIANGLE

Conditions											
C1: a < b+c?	F	T	T	T	T	T	T	T	T	T	T
C2: b < a+c?	-	F	T	T	T	T	T	T	T	T	T
C3: c < a+b?	-	-	F	T	T	T	T	T	T	T	T
C4: a=b?	-	-	-	T	T	T	T	F	F	F	F
C5: a=c?	-	-	-	T	T	F	F	T	T	F	F
C6: b=c?	-	-	-	T	F	T	F	T	F	T	F
Actions											
A1: Not a Triangle	X	X	X								
A2: Scalene											X
A3: Isosceles							X		X	X	
A4: Equilateral				X							
A5: Impossible					X	X		X			

152

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – BẢNG QUYẾT ĐỊNH TRIANGLE (tt)

Case ID	a	b	c	Expected Output
DT1	4	1	2	Not a Triangle
DT2	1	4	2	Not a Triangle
DT3	1	2	4	Not a Triangle
DT4	5	5	5	Equilateral
DT5	?	?	?	Impossible
DT6	?	?	?	Impossible
DT7	2	2	3	Isosceles
DT8	?	?	?	Impossible
DT9	2	3	2	Isosceles
DT10	3	2	2	Isosceles
DT11	3	4	5	Scalene

153

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – BẢNG QUYẾT ĐỊNH NEXTDATE

- Chúng ta có thể chia thành các lớp tương đương sau:

  M1= {month | month has 30 days}

  M2= {month | month has 31 days}

  M3= {month | month is February}

  D1= {day | 1 ≤ day ≤ 28}

  D2= {day | day = 29}

  D3= {day | day = 30}

  D4= {day | day=31}

  Y1= {year | year = 1900}

154

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – BẢNG QUYẾT ĐỊNH NEXTDATE

- Y2= {year | 1812 ≤ year ≤ 2012 AND year ≠ 1900 AND (0 = year mod 4}
- Y3= {year | 1812 ≤ year ≤ 2012 AND 0 ≠ year mod 4}

→Khi đó ta có bảng quyết định như sau

155

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – BẢNG QUYẾT ĐỊNH NEXTDATE

Conditions	1	2	3	4	5	6	7	8
C1: month in	M1	M1	M1	M1	M2	M2	M2	M2
C2: day in	D1	D2	D3	D4	D1	D2	D3	D4
C3: year in	-	-	-	-	-	-	-	-
Rule count	3	3	3	3	3	3	3	3
Actions								
A1: Impossible				X				
A2: Increment day	X	X			X	X	X	
A3: Reset day			X					X
A4: Increment month			X					?
A5: reset month								?
A6: Increment year								?

156

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – BẢNG QUYẾT ĐỊNH NEXTDATE (tt)

Conditions	9	10	11	12	13	14	15	16
C1: month in	M3	M3	M3	M3	M3	M3	M3	M3
C2: day in	D1	D1	D1	D2	D2	D2	D3	D3
C3: year in	Y1	Y2	Y3	Y1	Y2	Y3	-	-
Rule count	1	1	1	1	1	1	3	3
Actions								
A1: Impossible				X		X	X	X
A2: Increment day		X						
A3: Reset day	X		X		X			
A4: Increment month	X		X		X			
A5: reset month								
A6: Increment year								

157

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – BẢNG QUYẾT ĐỊNH NEXTDATE (lần 2)

- Xét một cách phân hoạch khác:

M1= {month | month has 30 days}
M2= {month | month has 31 days}
M3= {month | month is December}
M4= {month | month is February}

D1= {day | 1 ≤ day ≤ 27}
D2= {day | day = 28}
D3= {day | day = 29}
D4= {day | day = 30}
D5= {day | day=31}

Y1= {year | year is a leap year}
Y2= {year | year is a common year}

M1= {month | month has 30 days}
M2= {month | month has 31 days}
M3= {month | month is February}

D1= {day | 1 ≤ day ≤ 28}
D2= {day | day = 29}
D3= {day | day = 30}
D4= {day | day=31}

Y1= {year | year = 1900}
Y2= {year | 1812 ≤ year ≤ 2012
AND year ≠ 1900
AND (0 = year mod 4}
Y3= {year | 1812 ≤ year ≤ 2012
AND 0 ≠ year mod 4}

158

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – BẢNG QUYẾT ĐỊNH NEXTDATE (lần 2)

Conditions	1	2	3	4	5	6	7	8	9	10
C1: month in	M1	M1	M1	M1	M1	M2	M2	M2	M2	M2
C2: day in	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5
C3: year in	-	-	-	-	-	-	-	-	-	-
Actions										
A1: Impossible					X					
A2: Increment day	X	X	X			X	X	X	X	
A3: Reset day				X						X
A4: Increment month				X						X
A5: reset month										
A6: Increment year										

159

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – BẢNG QUYẾT ĐỊNH NEXTDATE (lần 2)

Conditions	11 22	12	13	14	15	16	17	18	19	20	21
C1: month in	M3	M3	M3	M3	M3	M4	M4	M4	M4	M4	M4
C2: day in	D1	D2	D3	D4	D5	D1	D2	D2	D3	D3	D4	D5
C3: year in	-	-	-	-	-	-	Y1	Y2	Y1	Y2	-	-
Actions												
A1: Impossible										X	X	X
A2: Increment day	X	X	X	X		X	X					
A3: Reset day					X			X	X			
A4: Increment month								X	X			
A5: reset month					X							
A6: Increment year					X							

160

3.1.3. KIỂM THỬ BẢNG QUYẾT ĐỊNH – QUAN SÁT VÀ HƯỚNG DẪN

- Bảng quyết định phù hợp khi
  - Có nhiều quyết định đưa ra
  - Có các quan hệ logic quan trọng giữa các biến đầu vào
  - Có các tính toán liên quan đến các tập con của các biến đầu vào
  - Có quan hệ nhân quả giữa đầu vào và đầu ra
  - Có logic tính toán phức tạp (độ phức tạp đồ thị cyclomatic cao)
- Bảng quyết định không dễ mở rộng (scale up)
- Bảng quyết định có thể làm mịn, cải tiến dần

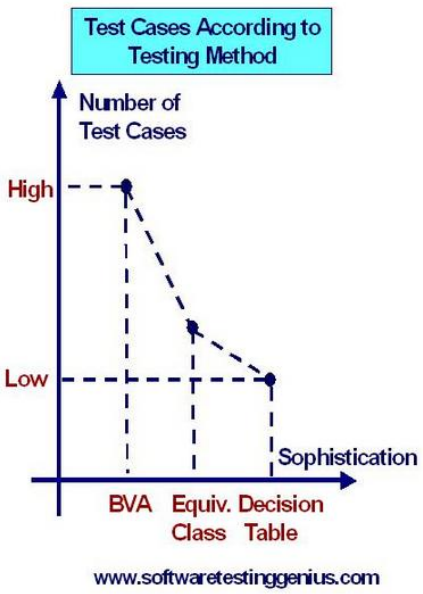
161

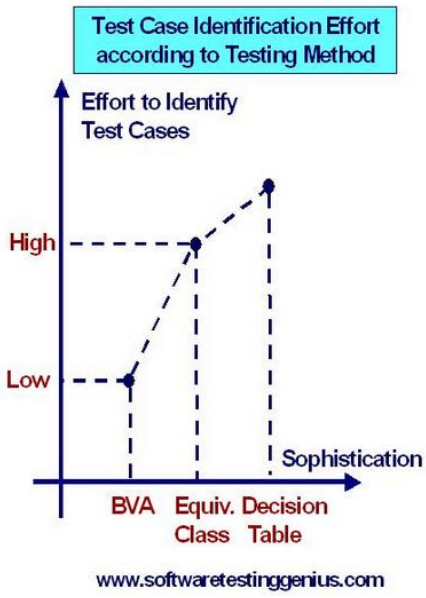
BÀI TẬP

- Hãy áp dụng các kỹ thuật bảng quyết định cho chương trình giải phương trình bậc hai?

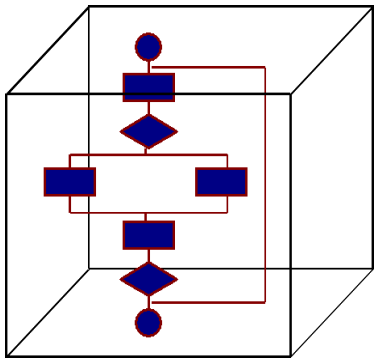
162

3.1.4 SO SÁNH CÁC PHƯƠNG PHÁP KIỂM THỬ HỘP ĐEN



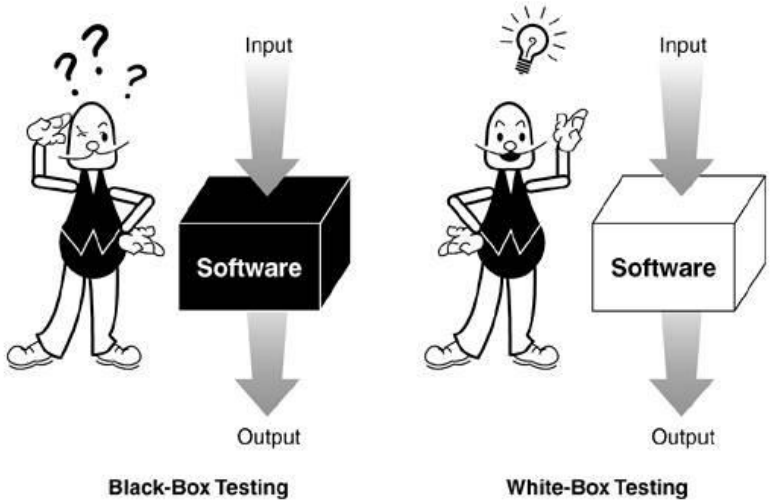


CHƯƠNG 3:
CÁC KỸ THUẬT KIỂM THỬ
KIỂM THỬ HỘP TRẮNG



164

DẪN NHẬP



NỘI DUNG

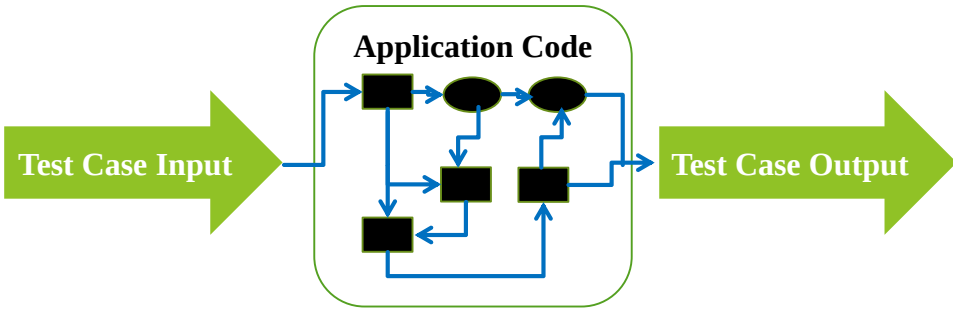
- Tổng quan kiểm thử hộp trắng
- Các kỹ thuật thực hiện
  - Kiểm thử theo dòng điều khiển
  - Kiểm thử theo dòng dữ liệu
  - Phương pháp kết hợp
- Tổng kết về kiểm thử hộp trắng

3.2.1 TỔNG QUAN KIỂM THỬ HỘP TRẮNG

- Kiểm thử hộp trắng (**White box testing**) là kỹ thuật kiểm thử sử dụng thông tin về bản chất hoạt động bên trong của chương trình để chọn dữ liệu kiểm thử.
- Các tên gọi khác: kiểm thử hộp kính (**Glass box testing**), hộp trong (**Clear box testing**), hộp mở (**Open box testing**), hay kiểm thử cấu trúc (**structural testing**).

167

3.2.1 TỔNG QUAN KIỂM THỬ HỘP TRẮNG (tt)



WHITE BOX TESTING APPROACH

Kiểm thử hộp trắng dựa trên mã nguồn để xây dựng các ca kiểm thử và kiểm tra đầu ra.

168

3.2.1 TỔNG QUAN KIỂM THỬ HỘP TRẮNG (tt)

- Kiểm thử cấu trúc dựa trên:
  - Các định nghĩa chặt chẽ liên quan đến ngữ nghĩa của ngôn ngữ lập trình: **luồng điều khiển**, **luồng dữ liệu**, **mục tiêu**, **tiêu chuẩn bao phủ**.
  - Phân tích toán học: **phân tích đồ thị, đường đi trong đồ thị**.
  - Các phép đo chính xác: **bao phủ**

169

3.2.1 TỔNG QUAN KIỂM THỬ HỘP TRẮNG (tt)

- Các kỹ thuật thực hiện
  - **Kiểm thử theo dòng điều khiển**
  - **Kiểm thử theo dòng dữ liệu**

170

3.2.2 KIỂM THỬ DÒNG ĐIỀU KHIỂN

- Kiểm thử dòng điều khiển (**Control flow testing**) tập trung kiểm thử tính đúng đắn của các giải thuật sử dụng trong các chương trình/đơn vị phần mềm.
- Sử dụng đồ thị dòng điều khiển và các độ đo kiểm thử để đo mức độ bao phủ chương trình của một tập ca kiểm thử cho trước.

171

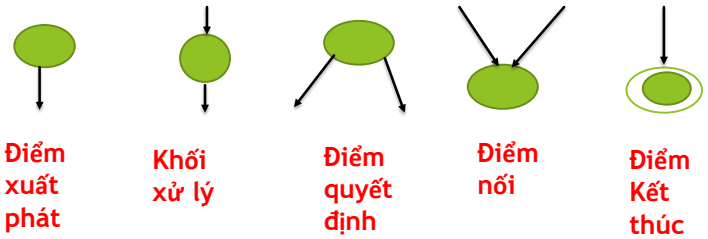
3.2.2.1 ĐỒ THỊ DÒNG ĐIỀU KHIỂN

- Là một đồ thị có nhãn và có hướng biểu diễn một chương trình, trong đó:
  - **Đỉnh** biểu diễn một lệnh tuần tự hoặc khối lệnh tuần tự.
  - **Cung** biểu diễn luồng điều khiển.
  - Một đỉnh vào và một đỉnh ra được thêm vào đồ thị để biểu diễn cho điểm vào ra tương ứng của chương trình.

172

3.2.2.1 ĐỒ THỊ DÒNG ĐIỀU KHIỂN

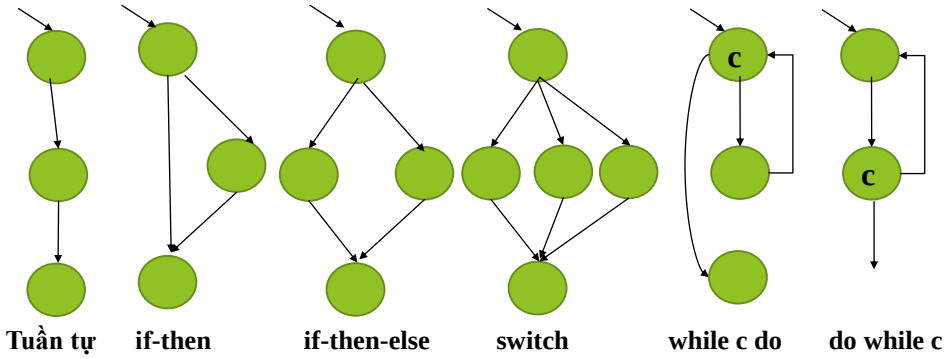
- Nếu i và j là hai đỉnh thì có một cạnh từ i đến j trong đồ thị chương trình nếu và chỉ nếu lệnh ở đỉnh j có thể chạy ngay sau (các) lệnh ở đỉnh i.
- Các thành phần cơ bản



173

3.2.2.1 ĐỒ THỊ DÒNG ĐIỀU KHIỂN

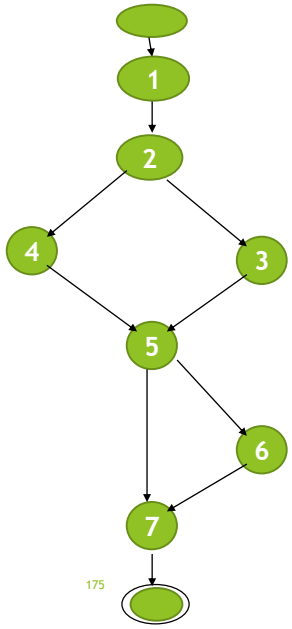
Các cấu trúc điều khiển phổ biến của chương trình



174

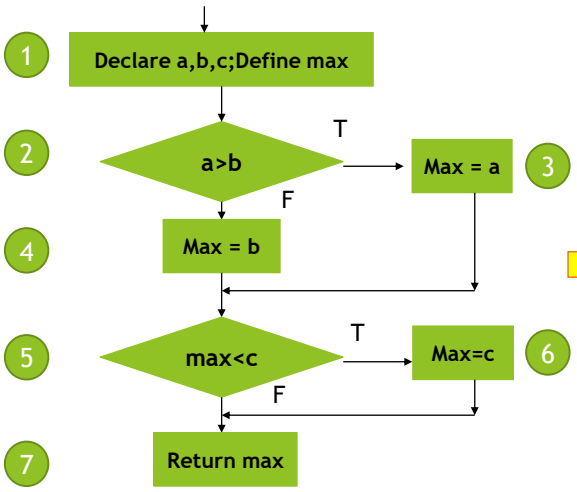
3.2.2.1 ĐỒ THỊ DÒNG ĐIỀU KHIỂN

```
int find_max ( int a, int b, int c)
   {
1.     int max;
2.     if ( a>b ) then
3.        max=a;
4.     else max=b;
5.     if ( max<c ) then
6.        max=c;
7.     return max;
   }
```

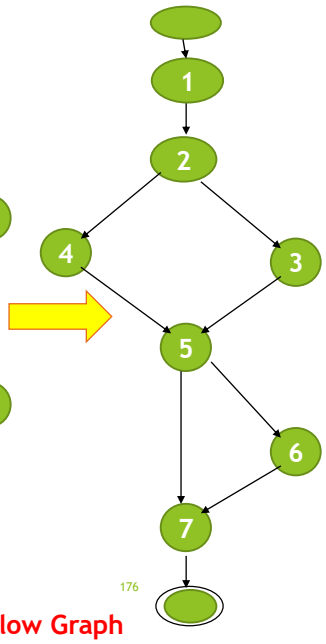


175

3.2.2.1 ĐỒ THỊ DÒNG ĐIỀU KHIỂN



Flow Chart



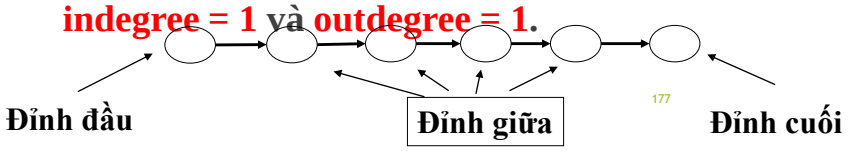
176

Flow Graph

3.2.2.1 ĐỒ THỊ DÒNG ĐIỀU KHIỂN - RÚT GỌN ĐỒ THỊ

- Dạng phổ biến nhất trong kiểm thử hộp trắng dựa trên đường đi **quyết định-đến-quyết định** (**decision-to-decision**)
- Đường đi DD là một dãy của đồ thị chương trình trong đó:
  - Một dãy là một đường đi với đỉnh đầu khác đỉnh cuối/kết thúc.
  - Mọi đỉnh trung gian có bậc vào và bậc ra bằng 1

    indegree = 1 và **outdegree = 1**.



177

3.2.2.1 ĐỒ THỊ DÒNG ĐIỀU KHIỂN - RÚT GỌN ĐỒ THỊ (tt)

- Chính xác hơn đường đi DD là dãy trong đồ thị chương trình thỏa mãn:
  - **Loại 1**: đỉnh đơn với bậc vào bằng 0
  - **Loại 2**: đỉnh đơn với bậc ra bằng 0
  - **Loại 3**: đỉnh đơn với bậc vào lớn hơn 1 hoặc bậc ra lớn hơn 1
  - **Loại 4**: đỉnh đơn với bậc vào bằng bậc ra và bằng 1
  - **Loại 5**: dãy với chiều dài lớn hơn 0

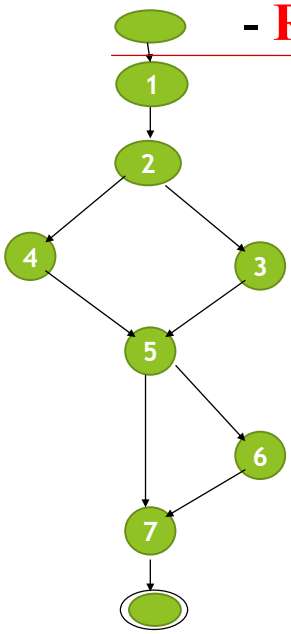
178

3.2.2.1 ĐỒ THỊ DÒNG ĐIỀU KHIỂN - RÚT GỌN ĐỒ THỊ (tt)

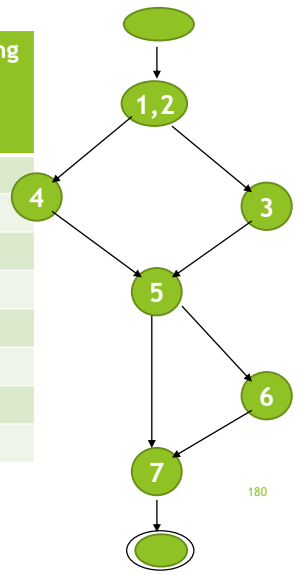
- Đồ thị đường đi DD của một chương trình trong ngôn ngữ mệnh lệnh là đồ thị có nhãn và có hướng mà các đỉnh là các đường đi DD và các cạnh là luồng điều khiển giữa các đường đi DD kế tiếp.
- Đường đi DD bản chất là một đồ thị thu gọn.

179

3.2.2.1 ĐỒ THỊ DÒNG ĐIỀU KHIỂN - RÚT GỌN ĐỒ THỊ (tt)



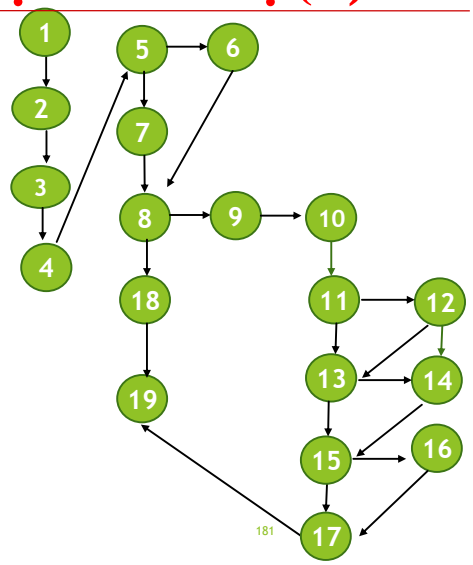
Các đỉnh đồ thị	Tên đường đi quyết định	Trường hợp
start	first	1
1-2	1,2	5
3	3	4
4	4	4
5	5	3
6	6	4
7	7	3
8	last	2



180

3.2.2.1 ĐỒ THỊ DÒNG ĐIỀU KHIỂN - RÚT GỌN ĐỒ THỊ (tt)

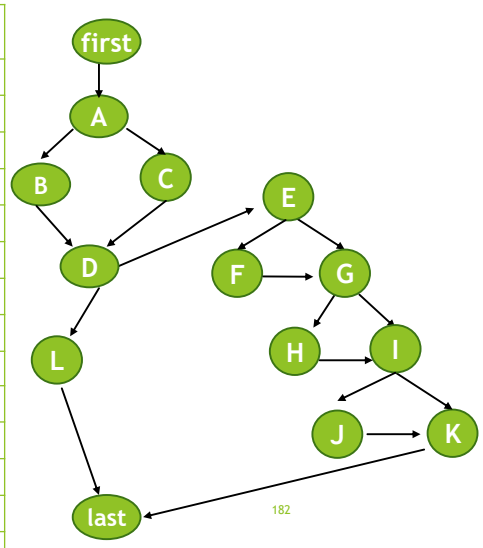
Các đỉnh đồ thị	Tên đường đi DD	Trường hợp
1	first	1
2-5	A	5
6	B	4
7	C	4
8	D	3
9-11	E	5
12	F	4
13	G	3
14	H	4
15	I	3
16	J	4
17	K	3
18	L	4
19	last	2



181

3.2.2.1 ĐỒ THỊ DÒNG ĐIỀU KHIỂN - RÚT GỌN ĐỒ THỊ (tt)

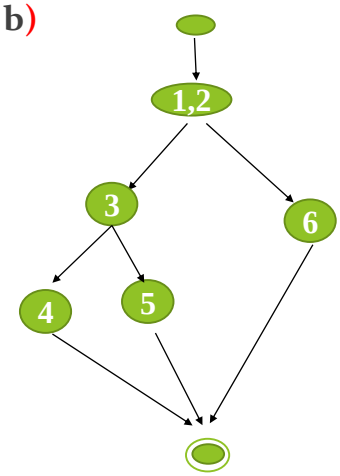
Các đỉnh đồ thị	Tên đường đi DD	Trường hợp
1	first	1
2-5	A	5
6	B	4
7	C	4
8	D	3
9-11	E	5
12	F	4
13	G	3
14	H	4
15	I	3
16	J	4
17	K	3
18	L	4
19	last	2



182

3.2.2.1 ĐỒ THỊ DÒNG ĐIỀU KHIỂN

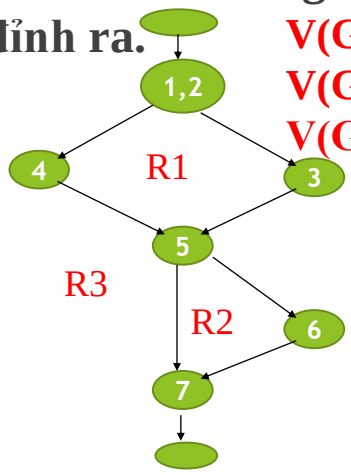
```
1.  void linearEquation( int a,int b)
    {
2.    if(a = = o)
3.      if(b = = o)
4.        cout<<"Vo so nghiem";
        else
5.        cout<<"Vo nghiem";
      else
6.      cout<<"x ="<<-b/a;
    }
```



183

3.2.2.1 ĐỒ THỊ DÒNG ĐIỀU KHIỂN- ĐƯỜNG ĐI KIỂM THỬ (TEST PATH)

- Là đường đi xuất phát từ đỉnh vào đi qua các đỉnh và cung khác và kết thúc tại đỉnh ra.



V(G)=Số cạnh – số đỉnh + 2
V(G)= Số đỉnh quyết định + 1
V(G) = Số miền phẳng

1. 1,2 → 3 → 5 → 7
2. 1,2 → 4 → 5 → 7
3. 1,2 → 3 → 5 → 6 → 7

184

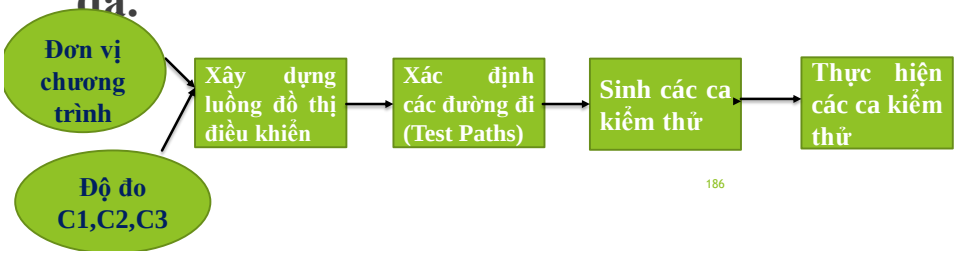
3.2.2.2 ĐỘ ĐO BAO PHỦ

- **Độ đo bao phủ** là dụng cụ để đo mức độ bộ kiểm thử phủ (**cover**) chương trình đến đâu? Độ bao phủ càng lớn thì độ tin cậy của bộ kiểm thử càng cao.
- Các độ đo bao phủ:
  - Phủ tất cả các lệnh (đỉnh)
  - Phủ tất các các quyết định (cung)
  - Phủ tất cả các điều kiện
  - Phủ tất cả các lộ trình

185

3.2.2.2 ĐỘ ĐO BAO PHỦ - KIỂM THỬ DỰA TRÊN ĐỘ ĐO

- Kiểm thử dựa trên độ đo là phương pháp chạy mã nguồn sao cho bao phủ một độ đo nào đó.
- Mục tiêu là kiểm thử với số ca kiểm thử tối thiểu nhưng đạt được độ bao phủ tối đa.



186

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ TẤT CẢ CÁC LỆNH – C1 (STATEMENT COVERAGE)

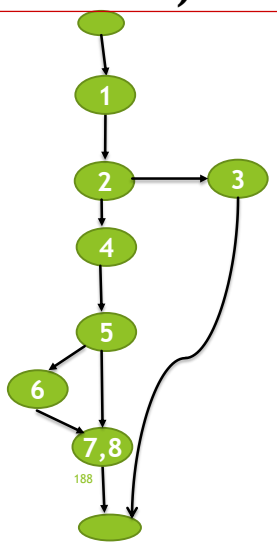
- Mục đích của phủ tất cả các lệnh (đỉnh) là mỗi lệnh phải được thực thi ít nhất một lần sau khi chạy các ca kiểm thử (test cases).
- Đây là độ đo khá tốt và việc đảm bảo độ đo này trong thực tế cũng khá tốn kém.

187

3.2.2.2 ĐỘ ĐO BAO PHỦ - KIỂM THỬ DỰA TRÊN ĐỘ ĐO C_1 (STATEMENT COVERAGE)

```
float foo (int a, int b, int c, int d, float e)
{
    float e;
    if (a = = 0) {
      return 0;
    }
    int x = 0;
    if ((a==b) || (c == d)){
      x=1;
    }
    e = 1/x;
    return e;
}
```

statement



188

3.2.2.2 ĐỘ ĐO BAO PHỦ - KIỂM THỬ DỰA TRÊN ĐỘ ĐO C_1 (STATEMENT COVERAGE)

- Các đường đi:
  - 1->2->4-> 5-> 6-> 7->8
  - 1->2-> 3
  - Để đạt được 100% độ phủ của độ đo C_1, ta cần kiểm tra tất cả các lệnh/khối lệnh (1-8) đều được xuất hiện ít nhất một lần trong các đường đi này.

189

3.2.2.2 ĐỘ ĐO BAO PHỦ - KIỂM THỬ DỰA TRÊN ĐỘ ĐO C_1 (STATEMENT COVERAGE)

- Sinh ca kiểm thử C_1

ID	Test Path	Inputs	EO	RO	Note
tc1	1; 2; 4; 5; 6; 7,8	2, 2, 3, 5	1		
tc2	1; 2; 3	0, 3, 2, 7	0		

190

3.2.2.2 ĐỘ ĐO BAO PHỦ - KIỂM THỬ DỰA TRÊN ĐỘ ĐO C_1 (STATEMENT COVERAGE)

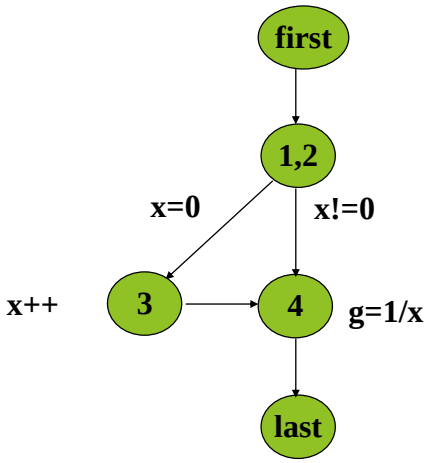
- Ví dụ: cho đoạn chương trình sau

```
cout<<"nhap gia tri: ";
cin>>x;
if(x==0)
   x++;
g=1/x;
```

Hãy thiết kế bộ kiểm thử theo độ đo C_1 cho đoạn chương trình trên ?

3.2.2.2 ĐỘ ĐO BAO PHỦ - KIỂM THỬ DỰA TRÊN ĐỘ ĐO C_1 (STATEMENT COVERAGE)

- Vậy lộ trình nào phủ tất cả các đỉnh ?



ID	Test Path	Inputs	EO
tc1	1; 2; 3; 4	0	1

Nhận xét: Tiêu chuẩn phủ tất cả các lệnh (đỉnh) thì không phủ các cung

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ TẤT CẢ CÁC CUNG – C_2 (DECISION COVERAGE)

- Các đỉnh quyết định trong đồ thị dòng điều khiển của đơn vị kiểm thử đều được thực hiện ít nhất một lần cả hai nhánh đúng và sai.
- Phủ tất cả các cung sẽ kéo theo phủ tất cả các đỉnh.

193

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ TẤT CẢ CÁC CUNG – C_2 (DECISION COVERAGE)

- Các trường hợp cần kiểm thử của độ đo C2 với hàm foo

ID	Test Path	Inputs	EO	RO	Note
tc1	1; 2; 4; 5; 6; 7,8	2, 2, 3, 5	1		
tc2	1; 2; 3	0, 3, 2, 7	0		

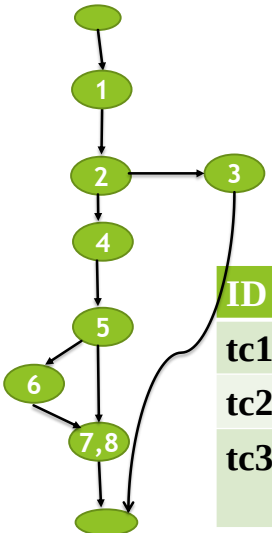
Điểm quyết định	Điều kiện tương ứng	Đúng	Sai
2	a= =0	tc2	tc1
5	(a = = 0) || (c = = d)	tc2	?

ID	Inputs	EO	RO	Note
tc1	0, 1, 2, 3	0		
tc2	1, 1, 2, 3	1		
tc3	1, 2, 1, 2	Lỗi chia cho 0		

```
float foo (int a, int b, int c, int d, float e)
{
  float e;
  if (a = = 0) {
    return 0;
  }
  int x = 0;
  if ((a==b) || (c == d)){
    x=1;
  }
  e = 1/x;
  return e;
}
```

194

3.2.2.2 ĐỘ ĐO BAO PHỦ - KIỂM THỬ DỰA TRÊN ĐỘ ĐO C_2 (DECISION COVERAGE)



- 1-> 2-> 4-> 5-> 6-> 7->8
- 1->2-> 3
- 1->2-> 4-> 5-> 7->8

ID	Test Path	Inputs	EO
tc1	1, 2, 4, 5, 6, 7, 8	2, 2, 3, 5	1
tc2	1, 2, 3	0, 3, 2, 7	0
tc3	1, 2, 4, 5, 7, 8	2, 3, 4, 5	lỗi chia cho 0

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ TẤT CẢ CÁC CUNG – C_2 (DECISION COVERAGE)

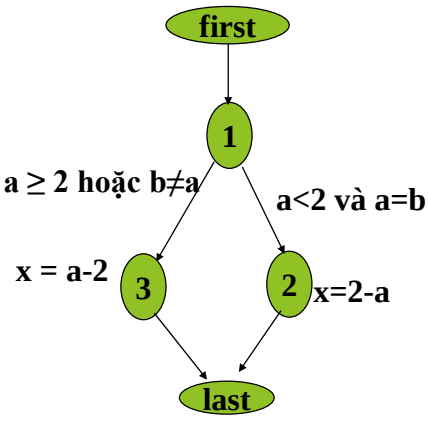
Ví dụ: cho đoạn chương trình sau

```
if(a < 2 && a==b)
   x= 2- a;
else
   x= a-2;
```

Hãy thiết kế bộ kiểm thử theo độ đo C_2 cho đoạn chương trình trên ?

196

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ TẤT CẢ CÁC CUNG – C_2 (DECISION COVERAGE)



Nếu ta chọn tập giá trị thử của cặp (a, b) là: {(1,1) và (3,3)} thì sẽ thỏa mãn tất cả các cung. Tuy nhiên, tập dữ liệu không phủ tất cả các điều kiện.

197

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ TẤT CẢ CÁC ĐIỀU KIỆN – C_3 (CONDITION COVERAGE)

- Tiêu chuẩn phủ tất các điều kiện được thỏa mãn khi và chỉ khi tiêu chuẩn phủ tất cả các cung được thỏa mãn và mỗi biểu thức điều kiện được thử với tất cả các giá trị có thể.
- Các điều kiện con thuộc các điều kiện phức tạp tương ứng với các điểm quyết định trong đồ thị dòng điều khiển của đơn vị cần kiểm thử đều được thực hiện ít nhất một lần cả hai nhánh đúng và sai.

198

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ TẤT CẢ CÁC ĐIỀU KIỆN – C_3 (CONDITION COVERAGE)

```
if(A && B) // điều kiện 1
  F1();
else
  F2();
if(C || D) // điều kiện 2
  F3()
else
  F4();
```

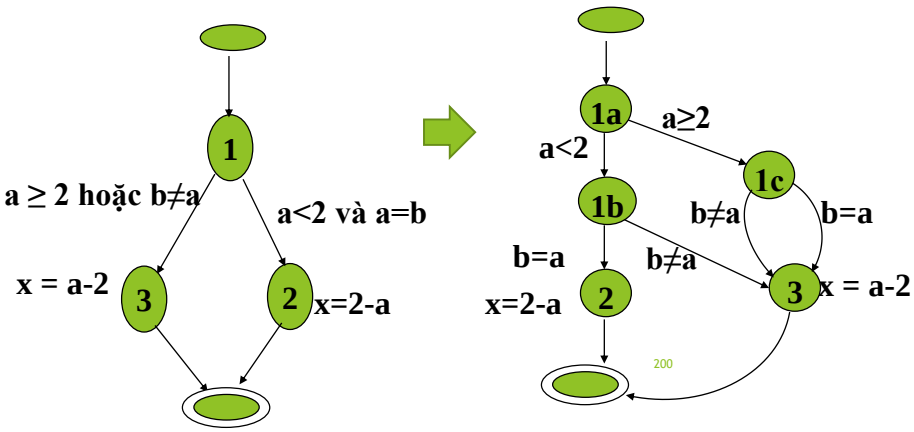
❖ Các ca kiểm thử

Cho điều kiện 1	Cho điều kiện 2
A=T, B=T A=T, B=F A=F, B=T A=F, B=F	C=T, D=T C=T, D=F C=F, D=T C=F, D=F

199

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ TẤT CẢ CÁC ĐIỀU KIỆN – C_3 (CONDITION COVERAGE)

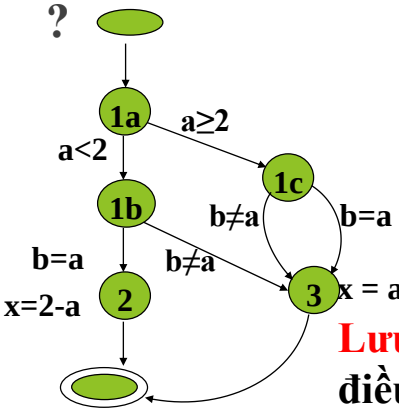
Đồ thị luồng điều khiển ở ví dụ trước được trình bày lại để phủ tất cả các điều kiện như sau:



200

3.2.2.2 ĐỘ ĐO BAO PHỦ - KIỂM THỬ DỰA TRÊN ĐỘ ĐO C_3 (CONDITION COVERAGE)

- Vậy lộ trình nào phủ tất cả các điều kiện ?

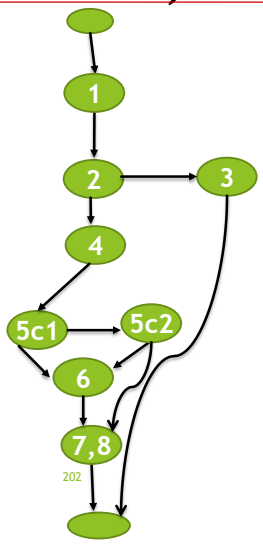


ID	Test Path	Inputs	EO
tc1	1a, 1b, 2	1, 1	1
tc2	1a, 1b, 3	1, 0	-1
tc3	1a, 1c, 3	3, 0	1
tc4	1a, 1c, 3	3, 3	1

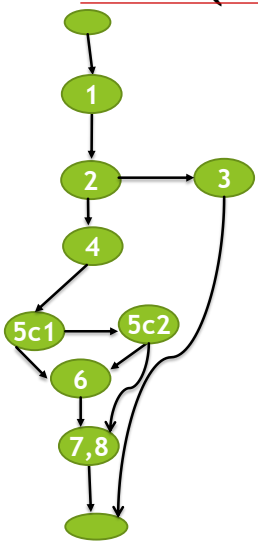
Lưu ý: Trường hợp biểu thức điều kiện phức tạp sẽ dẫn đến sự "bùng nổ" sự kết hợp.

3.2.2.2 ĐỘ ĐO BAO PHỦ - KIỂM THỬ DỰA TRÊN ĐỘ ĐO C_3 (CONDITION COVERAGE)

```
float foo (int a, int b, int c, int d, float e)
{
    1. float e;
    2. if (a = = 0) {
    3.      return 0;
    }
    4. int x = 0;
    5. if ((a==b) || (c == d)) {
      5c1      5c2
    6.      x=1;
    }
    7. e = 1/x;
    8. return e;
}
```



3.2.2.2 ĐỘ ĐO BAO PHỦ - KIỂM THỬ DỰA TRÊN ĐỘ ĐO C_3 (CONDITION COVERAGE)



1. 1; 2; 4; 5c1; 6; 7,8
2. 1; 2; 4; 5c1; 5c2; 6; 7,8
3. 1; 2; 4; 5c1; 5c2; 7,8
4. 1; 2; 3

ID	Test Path	Inputs	EO
tc1	1, 2, 4, 5c1, 6, 7, 8	0, 2, 3, 5	0
tc2	1, 2, 4, 5c1, 5c2, 6, 7, 8	2, 2, 2, 7	0
tc3	1, 2, 4, 5c1, 5c2, 7, 8	2, 3, 4, 5	lỗi chia 0
tc4	1, 2, 3	2, 3, 4, 4	1

203

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ TẤT CẢ LỘ TRÌNH(PATH COVERAGE)

```
Double average (int [] Arr, int min, int max)
{
        int sum =0, count = 0, i=0;
    double avg = 0.0;
    while (Arr[i] != -999 && i <100)
    {
        if (Arr[i] >= min && Arr[i] <= max)
        {
                sum += Arr[i];
            count++;}
                i++;}
        if (count>0)  avg = (double) sum/count;
    else avg = -999;
    return avg;}
```

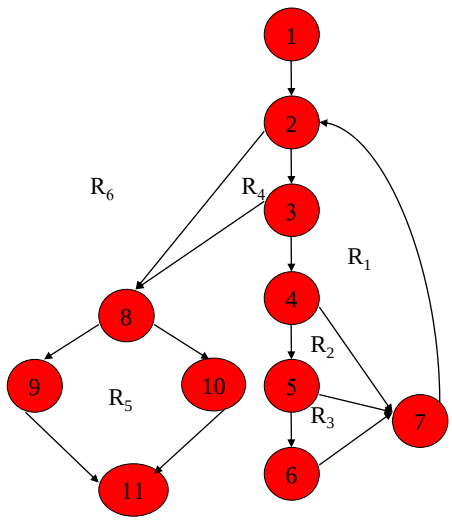
204

Hãy xác định tập dữ liệu thử để phủ tất cả các đỉnh, phủ tất cả các cung?

```
Double average (int [] Arr, int min, int max)
{
      /*1*/ int sum =0, count = 0, i=0;
      double avg = 0.0;
      while (/*2*/Arr[i] != -999 && /*3*/i <100)
      {
        if (/*4*/Arr[i] >= min && /*5*/Arr[i] <= max){
            /*6*/sum += Arr[i];
                count++;}
    /*7*/i++;}
      if (/*8*/count>0)  /*9*/avg = (double) sum/count;
      else /*10*/avg = -999;
/*11*/return avg;}
```

205

VẼ ĐỒ THỊ LUỒNG ĐIỀU KHIỂN



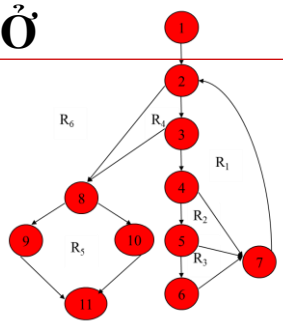
206

BƯỚC 2: TÍNH ĐỘ PHỨC TẠP CYCLOMATIC

- Công thức 1: V(G) = R = 6
- Công thức 2: V(G) = P + 1 = 5 + 1 = 6
- Công thức 3: V(G) = E – N + 2 = 15 – 11 + 2 = 6

207

BƯỚC 3: TÌM TẬP CÁC LỘ TRÌNH CƠ SỞ



- Lộ trình 1: 1-2-8-9-11
- Lộ trình 2: 1-2-8-10-11
- Lộ trình 3: 1-2-3-8-9-11
- Lộ trình 4: 1-2-3-4-7-2- …
- Lộ trình 5: 1-2-3-4-5-7-2- …
- Lộ trình 6: 1-2-3-4-5-6-7-2- …

- Các dấu (…) phía sau các lộ trình 4,5,6 có nghĩa là một lộ trình (hợp lệ) bất kỳ đi qua các phần còn lại của cấu trúc điều khiển đều có thể chấp nhận.
- Quan trọng là việc chỉ ra các đỉnh điều kiện
- Trong ví dụ này, các đỉnh điều kiện là 2, 3, 4, 5, 8.

208

BƯỚC 4: THIẾT KẾ CÁC TEST CASE

- **Lộ trình 1: 1-2-8-9-11**
  - **Input: Arr = {3,5,11,-999}, min = 0, max = 100**
  - **Expected Result: average = (3+5+11)/3 = 6.0**
  - **Mục đích: Kiểm thử việc tính trung bình chính xác**
- **Lộ trình 2: 1-2-8-10-11**
  - **Input: Arr = {-999}, min = 0, max = 0**
  - **Expected Result : average = -999**
  - **Mục đích: Kiểm thử việc tạo đầu ra average = -999**
- **Lộ trình 3: 1-2-3-8-9-11**
  - **Input: Arr = {3,5,7,8,10,….,80} (nhập đủ 101 số hợp lệ), min = 0, max = 100**
  - **Expected Result: Trung bình của 100 số đầu tiên trong mảng Arr**
  - **Mục đích: Chỉ tính trung bình cho 100 số hợp lệ đầu tiên.**

209

BƯỚC 4: THIẾT KẾ CÁC TEST CASE

- **Lộ trình 4: 1-2-3-4-7-2- …**
  - **Input: Arr = {7,23,-5,4,34,6,-999}, min = 0, max =100**
  - **Expected Result : average = (7+23+4+34+6)/5**
  - **Mục đích: Kiểm thử biên dưới (Arr[i]<min, i<100)**
- **Lộ trình 5: 1-2-3-4-5-7-2- …**
  - **Input: Arr = {7,23,104,4,34,6,-999}, min = 0, max =100**
  - **Expected Result : average = (7+23+4+34+6)/5**
  - **Mục đích: Kiểm thử biên trên (Arr[i]>max, i<100)**
- **Lộ trình 6: 1-2-3-4-5-6-7-2- …**
  - **Input: Arr = {7,32,99,23,86,2-999}, min = 0, max = 100**
  - **Expected Result : average = (7+32+99+23+86+2)/6**
  - **Mục đích: Kiểm thử việc tính trung bình là đúng.**

210

BÀI TẬP 2

```
void In_phantu(float A[],int  n)
{
   int i, j, dem, dem_tong = 0;
   cout<<"\nCac phan tu khac nhau la";
   for(i=0; i< n-1; i++){
     dem=0;
     for(j=i+1; j<n; j++){                    if(A[i]= =A[j]) dem++;
                          if(dem= =0){
              cout<<"   "<<A[i];                    dem_tong++;}
         }}
    if(dem_tong = =0) cout<<"\nKhong co so nao ca";
    else cout<<"   "<<A[i];
    return;}
```

211

Hãy xác định các tập kiểm thử để phủ tất cả các lộ trình?

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ VÒNG LẶP (LOOP COVERAGE)

- Trong thực tế, lỗi hay xảy ra ở các vòng lặp. Với mỗi đơn vị chương trình có vòng lặp cần quan tâm đến ba trường hợp sau:
  - **Lệnh lặp đơn giản**: đơn vị chương trình chỉ chứa đúng một vòng lặp(thân của vòng lặp không chứa các vòng lặp khác).
  - **Lệnh lặp liền kề**: đơn vị chương trình chỉ chứa các lệnh lặp kế tiếp nhau.
  - **Lệnh lặp lồng nhau**: đơn vị chương trình chỉ chứa các vòng lặp chứa các lệnh lặp khác.

212

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ VÒNG LẶP (LOOP COVERAGE)

- Cần sinh thêm bảy ca kiểm thử ứng với bảy trường hợp sau:

1. Vòng lặp thực hiện 0 lần
2. Vòng lặp thực hiện 1 lần
3. Vòng lặp thực hiện 2 lần
4. Vòng lặp thực hiện k lần, 2 < k < n - 1, với n là số lần lặp tối đa của vòng lặp
5. Vòng lặp thực hiện n - 1 lần
6. Vòng lặp thực hiện n lần
7. Vòng lặp thực hiện n + 1 lần

213

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ VÒNG LẶP (LOOP COVERAGE)

- Một số trường hợp có thể không xác định được số lần lặp tối đa của các vòng lặp, chỉ cần sinh bốn ca kiểm thử đầu tiên.
- Trong một số các trường hợp khác không thể sinh ca kiểm thử để vòng lặp thực hiện n + 1 lần (trường hợp thứ 7). Khi đó, chỉ cần sinh sáu ca kiểm thử còn lại (các trường hợp từ 1–6).

214

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ VÒNG LẶP (LOOP COVERAGE)

- Với các chương trình/đơn vị chương trình có các vòng lặp liền kề, tiến hành kiểm thử tuần tự từ trên xuống. Mỗi vòng lặp được kiểm thử bằng bảy ca kiểm thử như vòng lặp đơn giản.
- Trường hợp các vòng lặp lồng nhau, tiến hành kiểm thử tuần tự các vòng lặp theo thứ tự từ trong ra ngoài (mỗi vòng lặp cũng dùng bảy ca kiểm thử như đã mô tả ở trên).

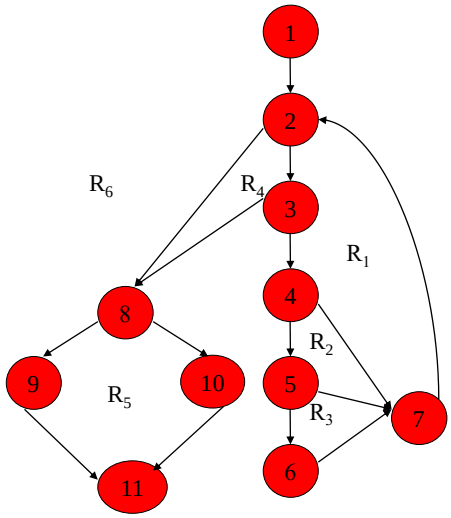
215

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ VÒNG LẶP (LOOP COVERAGE)

```
Double average (int [] Arr, int min, int max)
{
      /*1*/ int sum =0, count = 0, i=0;
     double avg = 0.0;
     while (/*2*/Arr[i] != -999 && /*3*/i <100)
     {
       if (/*4*/Arr[i] >= min && /*5*/Arr[i] <= max){
          /*6*/sum += Arr[i];
                count++;}
   /*7*/i++;}
      if (/*8*/count>0)  /*9*/avg = (double) sum/count;
      else /*10*/avg = -999;
/*11*/return avg;}
```

216

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ VÒNG LẶP (LOOP COVERAGE)



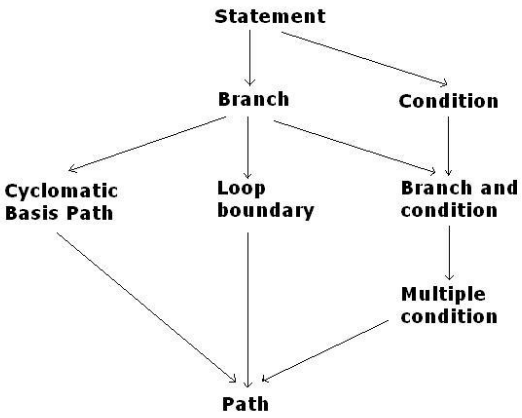
217

3.2.2.2 ĐỘ ĐO BAO PHỦ - PHỦ VÒNG LẶP (LOOP COVERAGE)

ID	Số lần lặp	Inputs	EO
Tcl0	0	[-999, ...], 1, 2	-999
Tcl1	1	[1,-999], 1, 2	1
Tcl2	2	[1,2,-999], 1, 2	1.5
Tclk	5	[1,2,3,4,5,-999], 1, 10	3
Tcl(n-1)	99	[1,2,...,99,-999], 1, 100	50
Tcln	100	[1,2,...,100], 1, 2	50.5
Tcl(n+1)			

218

QUAN HỆ BAO HÀM CHO LUỒNG ĐIỀU KHIỂN



219

3.2.3 KIỂM THỬ DÒNG DỮ LIỆU- Ý TƯỞNG

- Phương pháp kiểm thử dòng dữ liệu xem đơn vị chương trình gồm các đường đi tương ứng với các dòng dữ liệu nơi mà các biến được khai báo, được gán giá trị, được sử dụng để tính toán và trả lại kết quả mong muốn của đơn vị chương trình ứng với đường đi này.
- Với mỗi đường đi, chúng ta sẽ sinh một ca kiểm thử để kiểm tra tính đúng đắn của nó.

```
read (z)
x = 0
y = 0
if (z ≥ 0)
{
      x = sqrt (z)
      if (0 ≤ x && x ≤ 5)
            y = f (x)
      else
            y = h (z)
}
y = g (x, y)
print (y)
```

220

3.2.3 KIỂM THỬ DÒNG DỮ LIỆU- PHÂN LOẠI

- Quá trình kiểm thử dòng dữ liệu được chia thành hai pha riêng biệt: **kiểm thử dòng dữ liệu tĩnh** (static data flow testing) và **kiểm thử dòng dữ liệu động** (dynamic data flow testing).
- Kiểm thử dòng dữ liệu tĩnh, áp dụng các phương pháp phân tích mã nguồn mà không cần chạy chương trình/đơn vị chương trình nhằm phát hiện các vấn đề về khai báo, khởi tạo giá trị cho các biến và sử dụng chúng.
- Kiểm thử dòng dữ liệu động, tiến hành chạy các ca kiểm thử nhằm phát hiện các lỗi tiềm ẩn mà kiểm thử tĩnh không phát hiện được.

221

3.2.3 KIỂM THỬ DÒNG DỮ LIỆU- STATIC DATA FLOW TESTING

- Các vấn đề phổ biến về dòng dữ liệu có thể được phát hiện bằng phương pháp kiểm thử dòng dữ liệu tĩnh:
  - Gán giá trị rồi gán tiếp giá trị
  - Chưa gán giá trị nhưng được sử dụng

```
.
.
.
x = f1(y);
x = f2(z);
.
.
.
y=f(x1);
int z;
x=y+z;
.
.
.
```

  - Đã được khai báo và gán giá trị nhưng không được sử dụng

222

3.2.3.1 KIỂM THỬ DÒNG DỮ LIỆU-DYNAMIC DATA FLOW TESTING

- Hai lý do phải tiến hành kiểm thử dòng dữ liệu động của chương trình:
  - Nhằm chắc chắn một biến phải được gán đúng giá trị (xác định được một đường đi của biến từ điểm nó được định nghĩa đến điểm mà biến đó được sử dụng).
  - Ngay cả khi gán đúng giá trị cho biến thì các giá trị được sinh ra chưa chắc đã chính xác do tính toán hoặc các biểu thức điều kiện sai (biến được sử dụng sai).
- Kiểm thử luồng dữ liệu động liên quan đến việc chọn đường đi với mục tiêu bao phủ các cặp gán (definition) và dùng (use) dữ liệu, được gọi là các tiêu chuẩn luồng dữ liệu.

223

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU-DYNAMIC DATA FLOW TESTING

- Qui trình tổng quát của kiểm thử luồng dữ liệu là:
  - Vẽ đồ thị luồng dữ liệu cho chương trình
  - Chọn tiêu chuẩn kiểm thử luồng dữ liệu
  - Xác định các đường đi trong đồ thị để thỏa mãn tiêu chuẩn lựa chọn .
  - Tạo các ca kiểm thử cho các đường đi đã xác định
  - Thực hiện các ca kiểm thử để xác định các lỗi (có thể có) của chương trình
  - Sửa các lỗi (nếu có) và thực hiện lại tất cả các ca kiểm thử trong trường hợp bước trên phát hiện ra lỗi.

224

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU- ĐỒ THỊ DÒNG DỮ LIỆU

```
int VarTypes(int x, int y){
1.    int i;
2.    int *iptr;
3.     i = x;
4.    iptr = malloc(sizeof(int));
5.    *iptr = i + x;
6.    if (*iptr > y)
7.    return (x);
8.    else {
9.    iptr = malloc(sizeof(int));
10.   *iptr = x + y;
11.   return(*iptr);
12.   }
}
```

- **Định nghĩa của một biến**: một câu lệnh thực hiện việc gán giá trị cho một biến, ký hiệu là **def** . Ví dụ, lệnh i = x (dòng 3). Ngược lại gọi là **undef** với biến đó. Ví dụ, iptr = malloc(sizeof(int)) (dòng 4).
- **Biến được sử dụng**: một câu lệnh sử dụng một biến (để tính toán hoặc để kiểm tra các điều kiện) được gọi là **use** của biến đó.
  - Sử dụng để tính toán: **c-use** . Ví dụ: *iptr = i + x; c-use với i và x.
  - Sử dụng để kiểm tra các điều kiện: **p-use**. Ví dụ: if (*iptr > y), p-use với iptr và y.

225

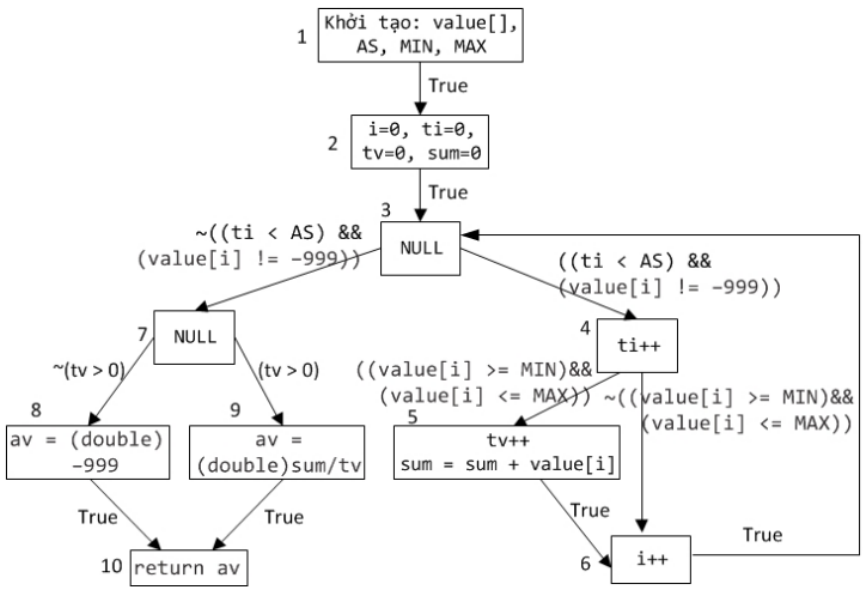
3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU- ĐỒ THỊ DÒNG DỮ LIỆU

- Đồ thị dòng dữ liệu của một chương trình/đơn vị chương trình là một đồ thị có hướng G = <N, E>:
- N là **tập các đỉnh** tương ứng với các **câu lệnh def** hoặc **c-use** của các biến được sử dụng trong đơn vị chương trình. Đồ thị G có hai đỉnh đặc biệt là đỉnh bắt đầu (tương ứng với lệnh def của các biến tham số) và đỉnh kết thúc đơn vị chương trình.
- E là **tập các cạnh** tương ứng với các **câu lệnh p-use** của các biến.

226

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU- ĐỒ THỊ DÒNG DỮ LIỆU

```
public static double ReturnAverage(int value[], int AS, int MIN, int MAX) {
        int i, ti, tv, sum;
        double av;
        i = 0; ti = 0; tv = 0; sum = 0;
        while (ti < AS && value [i] != -999) {
                ti++;
                if (value[i] >= MIN && value[i] <= MAX) {
                    tv++;
                 sum = sum + value[i];
               }
               i++;
        }
        if (tv > 0)
           av = (double) sum/tv;
        else
           av = (double) -999;
        return (av)
}
```

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU- KHÁI NIỆM DÒNG DL

Đỉnh i	def (i)	c-use (i)
1	{value, AS, MIN, MAX}	{}
2	{i, ti, tv, sum}	{}
3	{}	{}
4	{ti}	{ti}
5	{tv, sum}	{tv, i, sum, value}
6	{i}	{i}
7	{}	{}
8	{av}	{}
9	{av}	{sum, tv}

- **Global c-use:** Giả sử biến x được sử dụng để tính toán (**c-use**) tại đỉnh i của đồ thị dòng dữ liệu. Việc sử dụng biến x tại đỉnh i được gọi là Global c-use nếu x đã được định nghĩa ở các đỉnh trước đó. Ví dụ: Biến tv tại đỉnh 9 là Global c-use vì biến này đã được định nghĩa tại các đỉnh 2 và 5.

229

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU- KHÁI NIỆM DÒNG DL

Đỉnh i	def (i)	c-use (i)
1	{value, AS, MIN, MAX}	{}
2	{i, ti, tv, sum}	{}
3	{}	{}
4	{ti}	{ti}
5	{tv, sum}	{tv, i, sum, value}
6	{i}	{i}
7	{}	{}
8	{av}	{}
9	{av}	{sum, tv}
10	{}	{av}

Def -clear path: Giả sử biến x được định nghĩa (def) tại đỉnh i và được sử dụng tại đỉnh j . Một đường đi từ i đến j ký hiệu là (i- n_1- ... - n_m - j) với $m \geq 0$ được gọi là Def -clear path ứng với biến x nếu biến này không được định nghĩa tại các đỉnh từ n_1 đến n_m. Ví dụ, đường đi (2 - 3 - 4 - 5) là một Def -clear path ứng với biến tv, được định nghĩa tại đỉnh 2, được sử dụng tại đỉnh 5, và không được định nghĩa tại các đỉnh 3 và 4.

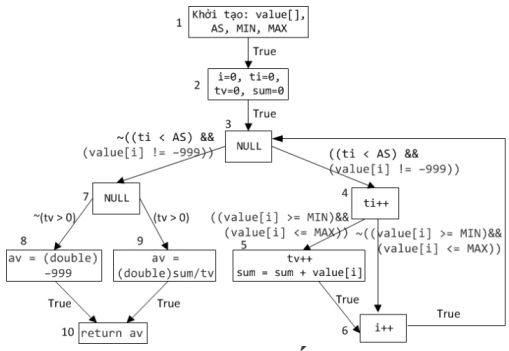
230

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU- KHÁI NIỆM DÒNG DL

Đỉnh i	def (i)	c-use (i)
1	{value, AS, MIN, MAX}	{}
2	{i, ti, tv, sum}	{}
3	{}	{}
4	{ti}	{ti}
5	{tv, sum}	{tv, i, sum, value}
6	{i}	{i}
7	{}	{}
8	{av}	{}
9	{av}	{sum, tv}
10	{}	{av}

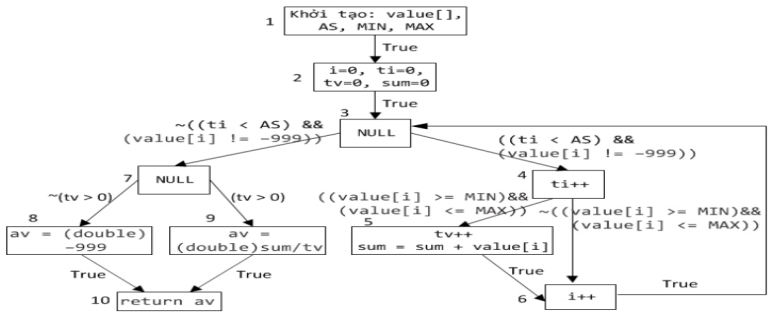
- **Global def** :Một đỉnh i được gọi là global def của biến x nếu đỉnh này định nghĩa biến x (def) và có một Def -clear path của x từ đỉnh i tới đỉnh chứa một Global c-use hoặc cạnh chứa một p-use của biến này.

231



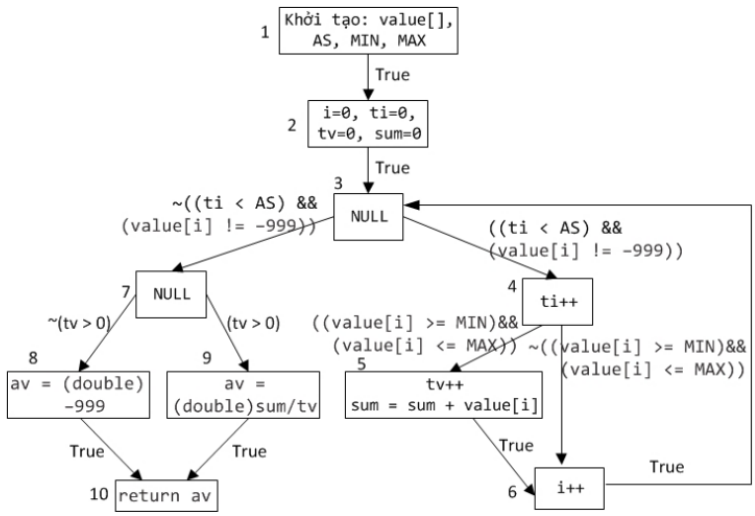
- **Simple path:** các đỉnh chỉ xuất hiện đúng một lần trừ đỉnh đầu và đỉnh cuối. Ví dụ các đường đi (2 - 3 - 4 - 5) và (3 - 4 - 6 - 3) là các Simple paths.
- **Loop-free path:** các đỉnh chỉ xuất hiện đúng một lần.
- **Complete-path**: có điểm bắt đầu và điểm kết thúc chính là điểm bắt đầu và điểm kết thúc của đồ thị dòng dữ liệu.

232



- **Du-path: Một đường đi (n_1- n_2- ... - n_j - n_k) được gọi là một Du-path (definition-use path) ứng với biến x nếu đỉnh n_1 là Global def của biến x và:**
- **đỉnh nk có một global c-use với biến x và (n_1- n_2- ... - n_j - n_k) là một Def-clear simple path với biến x, hoặc**
- **cạnh (nj, nk) có p-use với biến x và (n_1- n_2- ... - n_j) là Def –clear loop-free path với biến này.**

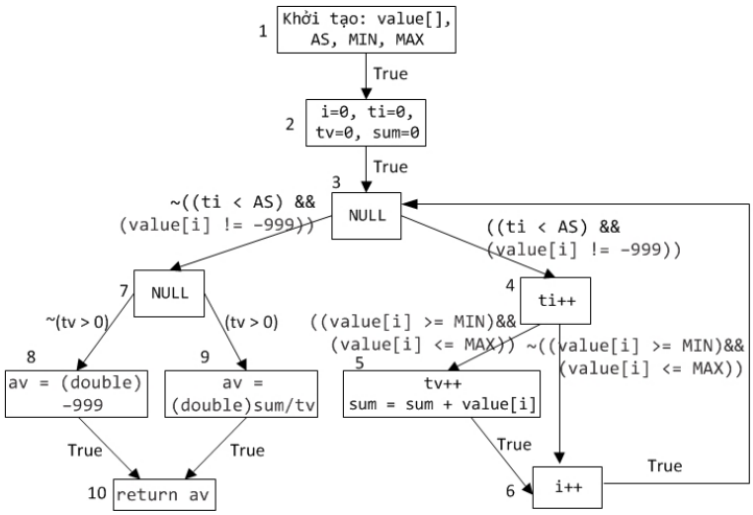
233



Ví dụ Du-path:

- **Đường đi (2 - 3 - 4 - 5) là một Du-path ứng với biến tv vì đỉnh 2 là Global def của biến tv, đỉnh 5 có Global c-use với biến này và (2 - 3 - 4 - 5) là một Def -clear simple path với biến tv.**

234



- **Đường đi (2 - 3 - 7 - 9) cũng là một du-path ứng với biến tv vì đỉnh 2 là Global def của biến này, cạnh (7, 9) có p-use với biến tv và (2 - 3 - 7) là một Def -clear loop-free path với tv.**

235

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU - CÁC ĐỘ ĐO KIỂM THỬ

- **Ý tưởng**
  - **Sử dụng thông tin def-use và tiêu chuẩn cụ thể để nhận được các đường đi cụ thể trong đồ thị dòng dữ liệu.**
  - **Từ đó xác định các ca kiểm thử**
- **Giả sử T là tập các đường đi đầy đủ và khả thi trong đồ thị dòng dữ liệu của chương trình P và V là tập tất cả các biến trong P**

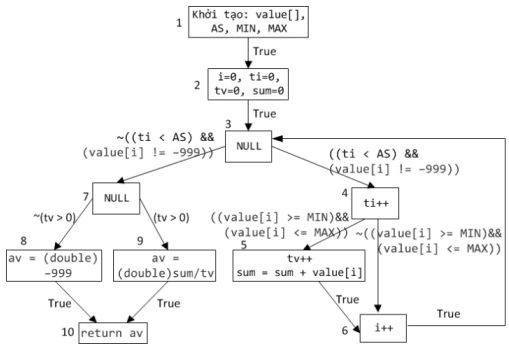
236

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU - CÁC ĐỘ ĐO KIỂM THỬ

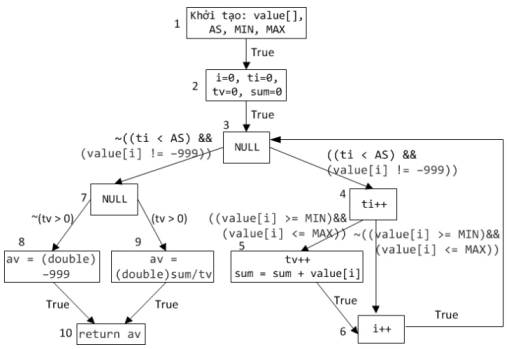
Biến tv có hai Global def tại các đỉnh 2 và 5.
Xét tại đỉnh 2:

- **Có một Global c-use của biến tv tại đỉnh 5**
- **Def -clear: path (2 - 3 - 4 - 5) từ đỉnh 2 tới đỉnh 5**
- **Complete-path:**

(1 - 2 - 3 - 4 - 5 - 6 - 3 - 7 - 9 - 10)
Vậy:
(2 - 3 - 4 - 5) thỏa mãn độ đo All-defs

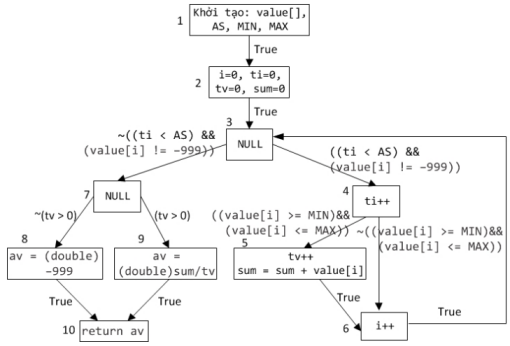


- **All-Defs: với mỗi một định nghĩa (def) của x tại một đỉnh ta cần ít nhất một đường đi xuất phát từ đỉnh đó tới một đỉnh khác sử dụng biến x sao cho đường đi này chứa một Def -clear path của biến đó và thuộc về một Complete-path nào đó.** 237



Đường đi (2 - 3 - 7 - 8) của biến tv cũng thỏa mãn độ đo All-defs

- **Có một Global def tại đỉnh 2, có một p-use tại cạnh (7, 8),**
- **Có một Def -clear path là (2 - 3 - 7 - 8) từ đỉnh 2 tới cạnh (7, 8)**
- **Thuộc về Complete-path (1 - 2 - 3 - 7 - 8 -10)** 238



Xét global def của tv tại đỉnh 5:

- **Có một Global c-use tại đỉnh 9**
- **(5 - 6 - 3 - 7 - 9) là một def -clear path**
- **Complete-path là (1- 2 - 3 - 4 - 5 - 6 - 3 - 7 - 9 - 10) chứa đường đi này**

Vì vậy (5 - 6 - 3 - 7 -9) cũng thỏa mãn độ đo All-defs.

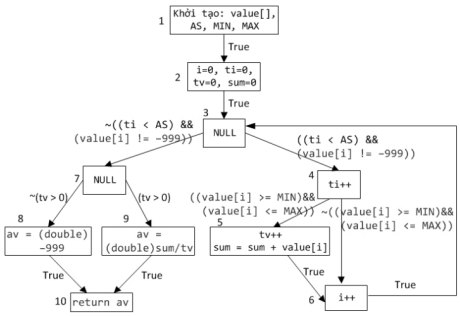
239

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU - CÁC ĐỘ ĐO KIỂM THỬ

Tìm tất cả các đường đi thỏa mãn độ đo All-c-uses ứng với biến ti:

- **Global def của biến này tại các đỉnh 2 và 4.**
- **Xét đỉnh 2:**
- **Global c-use của biến ti tại đỉnh 4**
- **Def -clear path tới đỉnh 4 là (2 - 3 - 4)**
- **Bốn Complete-path từ đồ thị dòng dữ liệu chứa đường đi này như sau:**

(1 - 2 - 3 - 4 - 5 - 6 - 3 - 7 - 8 - 10)
(1 - 2 - 3 - 4 - 5 - 6 - 3 - 7 - 9 - 10)
(1 - 2 - 3 - 4 - 6 - 3 - 7 - 8 - 10)
(1 - 2 - 3 - 4 - 6 - 3 - 7 - 9 - 10).



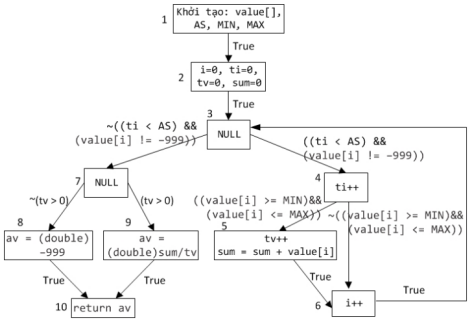
- **All-c-uses: với mỗi định nghĩa (def) của x ta tìm tất cả các đường đi xuất phát từ def của x tới tất cả các c-use của biến x sao cho các đường đi này có chứa một def–clear path của x và thuộc về một Complete-path nào đó.**

240

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU - CÁC ĐỘ ĐO KIỂM THỬ

Tìm tất cả các đường đi thỏa mãn độ đo All-p-uses ứng với biến tv:

- **Global def của biến này tại các đỉnh 2 và 5.**
- **Xét đỉnh 2:**
- **Hai p-use tại các cạnh (7, 8) và (7, 9)**
- **Def -clear path từ đỉnh 2 tới cạnh (7,8) là (2 - 3 – 7-8) và từ đỉnh 2 tới cạnh (7,9) là (2 - 3 - 7 – 9).**
- **Hai Complete-path từ đồ thị dòng dữ liệu chứa đường đi này như sau:**
  (1 - 2 - 3 - 7 - 8 - 10)
  (1 - 2 - 3 - 7 - 9 - 10)

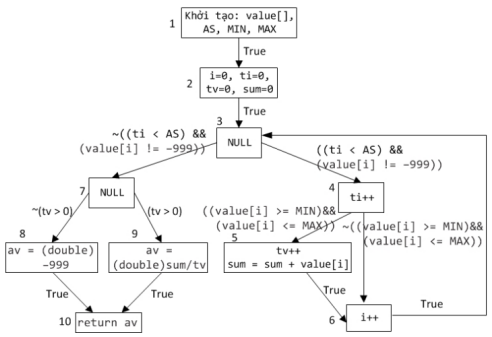


- **All-p-uses: với mỗi định nghĩa (def) của x ta tìm tất cả các đường đi xuất phát từ def của x tới tất cả các p-use của biến đó sao cho các đường đi này có chứa một Def -clear path của x và thuộc về một Complete-path nào đó.**

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU - CÁC ĐỘ ĐO KIỂM THỬ

Tìm tất cả các đường đi thỏa mãn độ đo All-p-uses/Some c-uses ứng với biến i? Global def của biến này tại các đỉnh 2 và 6. Dễ thấy rằng không có p-use của biến này. Xét đỉnh 2:

- **Global c-use của i tại đỉnh 6**
- **Def -clear path (2 - 3 - 4 – 5 -6).**
- **Chọn Complete-path (1 - 2 - 3 - 4 - 5 - 6 - 3 - 7 - 9 - 10) để thỏa mãn độ đo All-p-uses/Some-c-uses cho biến i.**



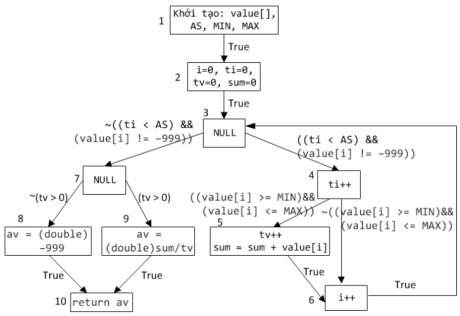
- **All-p-uses/Some-c-uses: Với mỗi một biến x và mỗi đỉnh i sao cho i là global def với biến x, chọn các Complete-path bao gồm các Def –clear path từ đỉnh i tới một số đỉnh j sao cho j là Global c-use của x.**

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU - CÁC ĐỘ ĐO KIỂM THỬ

Tìm tất cả các đường đi thỏa mãn độ đo All-c-uses/Some p-uses ứng với biến AS? Global def của biến này tại các đỉnh 1. Dễ thấy rằng không có Global c-use của biến này.

- **Các p-use của AS tại các cạnh (3, 7) và (3, 4).**
- **Các Def –clear path tương ứng với hai cạnh này là (1 - 2 - 3 - 7) và (1 - 2 - 3 - 4).**
- **Có rất nhiều Complete-path chứa hai đường đi này, ví dụ:**

(1 - 2 - 3 - 4 - 5 - 6 - 3 -7 - 9 - 10)



- **All-c-uses/Some-p-uses: Với mỗi một biến x và mỗi đỉnh i sao cho i là Global def với biến x, chọn các Complete-path bao gồm các Def –clear path từ đỉnh i tới một số cạnh**

  (j,k) sao cho có một p-use của x tại cạnh này.

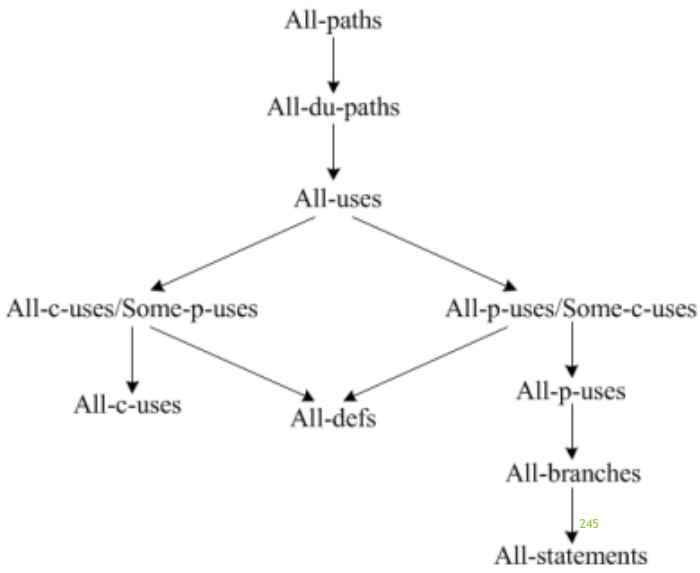
243

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU - CÁC ĐỘ ĐO KIỂM THỬ

- **All-uses: Độ đo này bao gồm các đường đi được sinh ra từ các độ đo All p-uses và All-c-uses. Điều này có nghĩa là với mỗi việc sử dụng (c-use hoặc p-use) của một biến thì có một đường đi từ định nghĩa (def) của biến đó tới các sử dụng của nó.**
- **All-du-paths: Với mỗi một biến x và mỗi đỉnh i sao cho i là Global def với biến x, chọn các Complete-path chứa các tất cả các Du-path từ đỉnh i tới:**
  - **Tất cả các đỉnh j sao cho có một Global c-use của biến x tại j , và**
  - **Tất cả các cạnh (j; k) sao cho có một p-use của biến x tại (j,k).**

244

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU - MỐI QUAN HỆ GIỮA CÁC ĐỘ ĐO



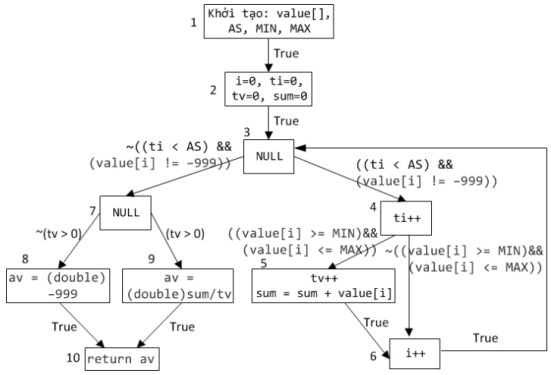
245

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU – SINH CA KIỂM THỬ

- Với độ đo kiểm thử C, xác định tất cả các đường đi đầy đủ (Complete-paths) thỏa mãn độ đo này.
- Để lựa chọn các bộ đầu vào ứng với các đường đi đầy đủ thỏa mãn một độ đo cho trước, chúng ta phải đảm bảo rằng các đường đi này là thực thi được.
- Như vậy, bài toán còn lại hiện nay là làm thế nào để sinh được bộ đầu vào cho từng đường đi đầy đủ trên. Bộ đầu vào này cùng với giá trị đầu ra mong đợi sẽ là ca kiểm thử cho đường đi này.

246

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU – SINH CA KIỂM THỬ



- Xét đường đi đầy đủ (1 - 2 - 3 - 4 - 5 - 6 - 3 - 7 - 9) từ đồ thị dòng dữ liệu . Từ đường đi này, ta sẽ xác định các biểu thức thuộc các p-uses nằm trên các cạnh của nó. Cụ thể, ta có các biểu thức sau:

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU – SINH CA KIỂM THỬ

- 1. ((ti<AS) && (value[i]!=-999)) (thuộc cạnh (3, 4)),
- 2. ((value[i] >= MIN) && (value[i] <= MAX)) (thuộc cạnh (4, 5)),
- 3. ~((ti<AS) && (value[i] != -999)) (thuộc cạnh (3, 7)), và
- 4. tv > 0 (thuộc cạnh (7, 9))

- Chúng ta sẽ sinh ra các giá trị đầu vào và kiểm tra các giá trị đó sao cho thỏa mãn hết các điều kiện trên, chẳng hạn bộ đầu vào ([1, -999], 2, 1, 2)

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU – BÀI TẬP 1

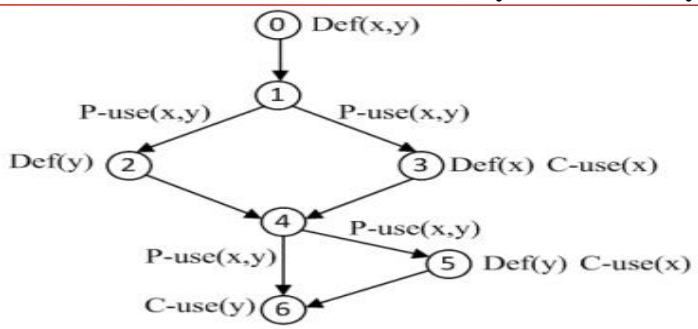
```
int calFactorial (int n){
      int result = 1;
      int i=1;
      while (i <= n){
            result = result*i;
            i++;
      }//end while
      return result;
}//the end
```

Hình 7.15: Mã nguồn C của hàm calFactorial.

- Hãy liệt kê các câu lệnh ứng với các khái niệm def , c−use, và p−use ứng với các biến được sử dụng trong hàm này.
- Hãy vẽ đồ thị dòng dữ liệu của hàm này.

249

3.2.3.2 KIỂM THỬ DÒNG DỮ LIỆU – BÀI TẬP 2



- Hãy xác định tất cả các Def -clear-path của các biến x và y.
- Hãy xác định tất cả các du-paths của các biến x và y.
- Dựa vào các chuẩn của kiểm thử dòng dữ liệu hãy xác định tất cả các All-p-uses/Some-c-uses và All-c-uses/Some-p-uses.
- Biểu thức của các p-use(x; y) tại cạnh (1,3) và (4,5) lần lượt là x + y = 4 và $x^2 + y^2 > 17$. Đường đi (0 - 1 - 3 - 4 - 5 - 6) có thực thi được không? Giải thích.

250

251

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM

252

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KHÁI NIỆM

- Chiến lược kiểm thử là sự **tích hợp các kỹ thuật thiết kế ca kiểm thử** tạo thành một dãy các bước nhằm hướng dẫn quá trình kiểm thử phần mềm thành công.
- Nó phác thảo lộ trình để:
  - **Nhà phát triển** tổ chức việc bảo đảm chất lượng,
  - **Khách hàng** hiểu được công sức, thời gian và nguồn lực cần cho kiểm thử.

253

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – YÊU CẦU

- **Chiến lược kiểm thử phải:**
  - Tích hợp được việc lập kế hoạch, thiết kế ca kiểm thử, tiến hành kiểm thử, thu thập và đánh giá các thông tin kết quả.
  - Đủ mềm dẻo để cổ vũ óc sáng tạo, đáp ứng được yêu cầu khách hàng.
  - Thích ứng với mức kiểm thử cụ thể.
  - Đáp ứng các đối tượng quan tâm khác nhau

254

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – ĐẶC TRƯNG

- **Các đặc trưng có tính khuôn mẫu:**
  - **Bắt đầu ở mức môđun và tiếp tục cho đến khi tích hợp ở mức hệ thống trọn vẹn.**
  - **Các kỹ thuật kiểm thử khác nhau là thích hợp cho những thời điểm khác nhau.**
  - **Được cả người phát triển và nhóm kiểm thử độc lập cùng tiến hành.**
  - **Kiểm thử đi trước gỡ lỗi, song việc gỡ lỗi phải thích ứng với từng chiến lược kiểm thử .**

255

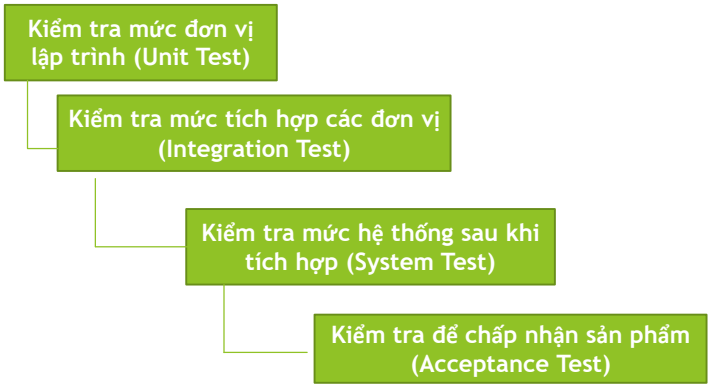
CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – ĐẶC TRƯNG

- **Chiến lược cần thích ứng với từng mức kiểm thử:**
  - **Kiểm thử mức thấp:** ***xác minh*** **từng khúc mã nguồn có tương ứng và thực thi đúng đắn không?**
  - **Kiểm thử mức cao:** ***xác minh và thẩm định*** **các chức năng hệ thống chủ yếu có** ***đúng đặc tả và đáp ứng yêu cầu*** **của khách hàng không?**

256

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – TIẾN TRÌNH KT

- **Tiến trình thực hiện kiểm thử tương ứng với tiến trình phát triển (theo từng mô hình).**
- **Tiến trình kiểm thử thông thường (mô hình chữ V):**



CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ ĐƠN VỊ

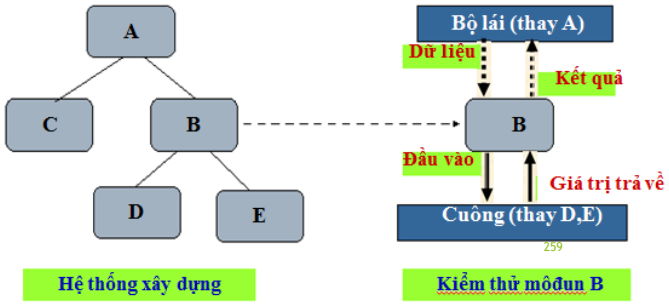
- **Nội dung kiểm thử:**
  - Chủ yếu sử dụng phương pháp KT hộp trắng để thực hiện
  - Giao diện
  - Cấu trúc dữ liệu sử dụng cục bộ
  - Đường điều khiển (đường độc lập)
  - Điều kiện lôgic
  - Phép toán xử lý
- **Câu hỏi:**

  Định lượng/dạng gì (biến, mô đun qua giao diện)?
  Yếu tố nào cần (vào/ra dữ liệu)?
  Sai xử lý, lôgíc (phép toán, biểu thức)?
  Sai điều khiển (vòng lặp, giá trị biên**)?**
  Sai tiềm ẩn (ghi chép, mô tả)?

258

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ ĐƠN VỊ

- Môđun không phải là chương trình cô lập, nên cần phát triển các phần mềm **bộ lái** và/hoặc **cuống** cho kiểm thử mỗi đơn vị.
- **Bộ lái (driver)** là một hàm "main" điều khiển việc đưa dữ liệu vào và nhận kết quả ra của module.
- **Cuống (stub)** dùng để thay cho các module là thứ cấp của nó.



CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ TÍCH HỢP

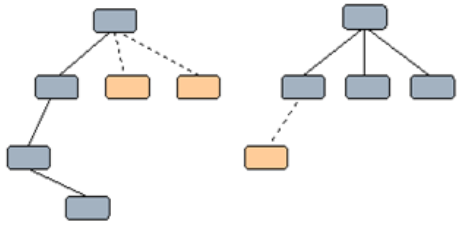
- Kiểm thử tích hợp *(integration testing)* nhằm nhận được 1 bộ phận chức năng hay 1 hệ con tốt.
- Là một kỹ thuật có tính hệ thống để xây dựng cấu trúc chương trình.
- Từ các môđun đã kiểm thử đơn vị, xây dựng cấu trúc chương trình đảm bảo tuân theo thiết kế.
- Có hai cách tích hợp
  - Tích hợp tăng dần: **từ trên xuống, dưới lên**
  - Tích hợp đồng thời 1 lúc: **big bang**

260

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ TÍCH HỢP

Chiến lược tích hợp từ trên xuống (Top-Down)

- Là một cách tiện lợi để xây dựng và kiểm soát cấu trúc chương trình.
- Gộp dần các môđun từ trên xuống theo trật tự cấp bậc điều khiển, bắt đầu từ môđun điều khiển "main", gắn từng môđun phụ trợ vào môđun điều khiển thương cấn.
- Có thể theo 2 cách:
  - Theo chiều "sâu trước"
  - Theo chiều "rộng trước".



261

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ TÍCH HỢP

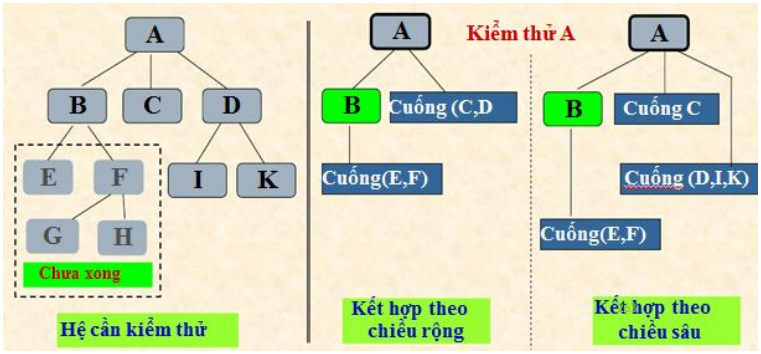
- Tích hợp từ trên xuống thực hiện theo các bước sau:

1. Môđun điều khiển chính được dùng như bộ lái kiểm thử (test driver) và tất cả các môđun phụ trợ trực tiếp được thay thế bởi các cuống (stub).
2. Thay thế dần từng cuống bởi môđun thực thi tương ứng.
3. Sau khi tích hợp môđun đó, tiến hành các kiểm thử tương ứng.
4. Khi hoàn thành các kiểm thử này thì thay một cuống khác bằng môđun thực (nghĩa là quay lại bước 2).

262

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ TÍCH HỢP

4. **Có thể kiểm thử lại (toàn bộ hoặc một phần các kiểm thử trước – kiểm thử hồi quy) để bảo đảm rằng không có sai mới nào được sinh ra.**
5. **Tiếp tục lặp lại từ bước 2 cho tới khi toàn bộ cấu trúc chương trình được xây dựng.**



CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ TÍCH HỢP

Chiến lược tích hợp từ dưới lên (Bottom- Up)

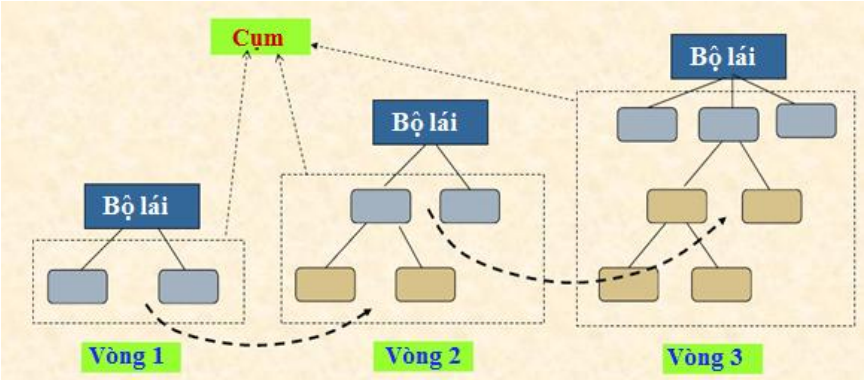
- **Bắt đầu xây dựng và kiểm thử từ các môđun nguyên tố.**
- **Thực hiện bước:**

1. **Các môđun mức thấp được tổ hợp vào trong các cụm (cluster) thực hiện một chức năng phụ trợ đặc biệt.**
2. **Một bộ lái được viết để phối hợp đầu vào và đầu ra của ca kiểm thử. Kiểm thử cụm đó.**
3. **Tháo bỏ các driver & các cụm được tổ hợp ngược lên trong cấu trúc chương trình**

264

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ TÍCH HỢP

Phương pháp tích hợp Bottom- Up:



265

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ TÍCH HỢP

- **Phương pháp trên-xuống:**
  - **Ưu điểm: kiểm thử ngay chức năng điều khiển hệ thống.**
  - **Nhược điểm: cần các cuống, những khó khăn kèm theo cuống.**
- **Phương pháp dưới –lên:**
  - **Ưu điểm: thiết kế ca kiểm thử dễ và không cần cuống**
  - **Nhược điểm: luôn chưa có chương trình chỉnh thể**

266

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ HỆ THỐNG

- **Mục đích System Test là kiểm thử thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không.**
- **System Test bắt đầu khi tất cả các bộ phận của PM đã được tích hợp thành công. Trọng tâm là đánh giá về hoạt động, thao tác, sự tin cậy và các yêu cầu khác liên quan đến chất lượng của toàn hệ thống.**
- **System Test thường được thực hiện bởi một nhóm kiểm thử viên hoàn toàn độc lập với nhóm phát triển dự án.**

267

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ HỆ THỐNG

- **Các loại kiểm thử hệ thống:**
  - **Kiểm thử chức năng mức hệ thống (Functional Test)**
  - **Kiểm thử phục hồi (Recovery Testing)**
  - **Kiểm thử an ninh (Security Test)**
  - **Kiểm thử thi hành (Performance Test)**
  - **Kiểm thử chịu tải (Stress Test hay Load Test)**

268

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ CHẤP NHẬN

- Sau khi thực hiện kiểm thử tích hợp, các kiểm thử cuối cùng bắt đầu - kiểm thử chấp nhận hay thẩm định (acceptation/validation testing).
- Mục đích của Acceptance Test để chứng minh PM thỏa mãn tất cả yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm.
- Thông thường sẽ thông qua hai loại kiểm thử gọi là Alpha Test và Beta Test.

269

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ CHẤP NHẬN

Kiểm thử Alpha

- Kiểm thử alpha được bên phát triển tiến hành:
  - Phần mềm được người dùng thực hiện trong bối cảnh "tự nhiên".
  - Người phát triển "nhòm qua vai" người sử dụng để báo cáo các sai và các vấn đề sử dụng (vì thế còn gọi là kiểm thử sau lưng).
- Kiểm thử alpha được tiến hành trong một môi trường được điều khiển (theo kế hoạch của người phát triển).
- Dữ liệu cho kiểm thử Alpha thường là dữ liệu môphỏng

270

CHƯƠNG 4. CÁC CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM – KIỂM THỬ CHẤP NHẬN

Kiểm thử Bêta

- Kiểm thử bêta được nhiều người đặt hàng tiến hành , không có mặt người phát triển.
- Được áp dụng trong môi trường thực, không có sự kiểm soát của người phát triển.
- Khách hàng sẽ báo cáo tất cả các vấn đề mà họ gặp phải trong quá trình kiểm thử cho người phát triển một cách định kỳ.
- Theo các báo đó người phát triển cải tiến và chuẩn bị phân phối bản hoàn thiện cho toàn bộ những người đặt hàng.

271

272

TÀI LIỆU THAM KHẢO

- **Phạm Ngọc Hùng, Trương Anh Hoàng, Đặng Văn Hưng, *Giáo trình kiểm thử phần mềm*, 2014.**
- **Glenford J. Myers, Corey Sandler, Tom Badgett, *The Art of Software Testing*, 2011.**
- **http://www.uet.vnu.edu.vn/~hoangta/**
- **http:// www.istqb.org**