Sưu tầm

1. Ghi Chú

Phiên Bản Ngày	Mô Tả/ Ghi Chú	Học Viên
1.0 03/30/2004	Bổ xung phần chương trình minh hoạ	Quí Nguyễn
01/10/2004	Tạo phiên bản đầu tiên	Quí Nguyễn

2. Mục Lục

3. Danh Sách Hình

4. Danh Sách Bảng

5. Lời Mở Đầu

Như một nhà phát triển phần mềm có kinh nghiêm đã nói "quá trình kiểm thử một phần mềm thật sự không bao giờ kết thúc, quá trình này chỉ chuyển từ bạn (một nhà phát triển phần mềm) sang một người khác(khách hàng). Và mỗi khi khách hàng sử dụng chương trình, thì quá trình này lại tiếp diễn. Nhưng bằng cách áp dụng các quá trình kiểm thử có tính chất hệ thống, những kỹ sư phần mềm có thể kiểm định một cách hoàn chỉnh và cũng bằng cách đó quá trình kiểm thử có thể phát hiện và chỉnh sửa phần lớn các lỗi trước khi quá trình kiểm thử mới của khách hàng bắt đầu.

Quá trình thiết kế các trường hợp có khả năng phát hiện các lỗi/sai sót kiếm khuyết của phần mềm có thể phân thành 2 loại kỹ thuật thiết kế : kỹ thuật kiểm thử "white box" và "black box".

"White box" là kỹ thuật tập trung kiểm tra trên cầu trúc điều kiển của chương trình, các trường trường hợp kiểm thử được tạo ra có thể đảm bảo tất cả các câu lệnh được viết trong chương trình được thực thi ít nhất một lần trong quá trình kiểm thử và rằng tất cả các điều kiện logic trong trương trình cũng đã được kiểm thử trong đó.

Trong khi "white box" được mô tả như là một phương pháp kiểm định trên diện nhỏ và là một phương pháp tiêu biểu để kiểm định các thành phần nhỏ của chương trình như module, hay những nhóm nhỏ các module. Thì "black box" lại tập trung vào hướng kiểm định trên diện rộng, các trường hợp được thiết kế tập trung phân tích/phân mãnh vùng thông tin đầu vào và đầu ra của chương trình.

Chúng ta sẽ bàn luân thật kỹ các kỹ thuật này vào chương sau, tiếp theo chúng ta tìm hiểu bài toán kiểm định chương trình và các mục tiêu của bài toán này.

6. Cơ Sở Kiểm Thử Phần Mềm

6.1. Bài Toán Kiểm Thử Phần Mềm

Quá trình phát triển một hệ thống phần mềm bao gồm một chuỗi các hoạt động sản sinh ra mã lệnh, tài liệu. Nơi mà những sai sót của con người có thể xãy ra bất kỳ lúc nào. Một lỗi có thể bắt đầu xuất hiện ngay tại lúc bắt đầu của quá trình phát triển, thiết kế, cài đặc. Do đó quá trình phát triển một phần mềm phải kết hợp với một quá trình kiểm thử.

Trong chương này sẽ thảo luận về các mục tiêu kiểm thử phần mềm. Chủ yếu ở đây là cung cấp các khái niệm và các mục tiêu cho việc kiểm thử phần mềm.

Kiểm thử phần mềm có thể đươc hiểu như là một quá trình bất thường thú vị. Thật sự thì trong giai đoạn ban đầu của quá trình phân tích, thiết kế và phát triển, Những kỹ sư lập trình đã cố gắng xây dựng một phần mềm từ những khái niệm khá trù tượng ngoài thực tế để hình thành một chương trình cụ thể. Và bây giờ đến giai đoạn kiểm thử họ lại tạo ra các trường hợp kiểm thử để nhằm "đánh đổ" phần mềm đã xây dựng. Thật sự thì quá trình kiểm đình là một bước trong quá trình phát triển phần mềm có tình chất tiêu cực nhằm bác bỏ hơn là xây dựng như các bước khác.

6.2. Các Mục Tiêu Kiểm Định

- Kiểm định là một quá trình thực thi chương trình với mục đích là tìm ra lỗi/các yếu điểm của chương trình.
- Một trường hợp kiểm thử tốt là một trường hợp có khả năng lớn trong việc tìm ra những lỗi chưa được phát hiện.
- Một trường hợp kiểm không tốt (không thành công) là một trường hợp mà khả năng tìm thấy những lỗi chưa biết đền là rất ít.

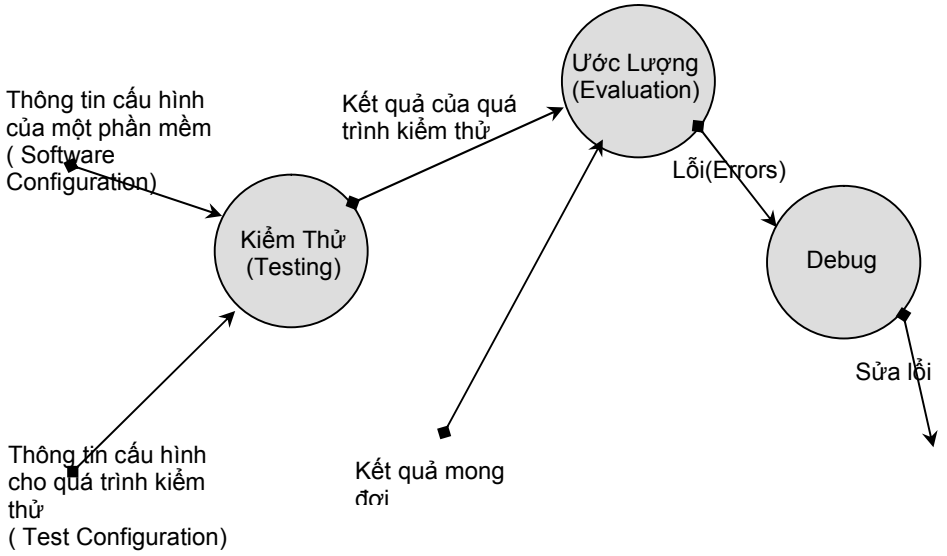
Mục tiêu của kiểm thử phần mềm là thiết kế những trượng hợp kiểm thử để có thể phát hiện một cách có thệ thống những loại lỗi khác nhau và thực hiện việc đó với lượng thời gian và tài nguyên ít nhất có thể.

6.3. Quá Trình Kiểm Định

Một mô hình cho quá trình kiểm định được mô tả dưới **Hình 1**. Thông tin đầu vào được cung cấp cho quá trinh kiểm định gồm :

- Thông tin cấu hình của phần mềm: các thông tin này bao gồm: mô tả về yêu cầu của phần mềm(Software Requirement Specification). Mô tả về thiết kế của chương trình(Design Specification) và mã của chương trình.

- Thông tin cầu hình về kiểm thử bao gồm : kế hoạch kiểm thử, và thủ tục kiểm thử và các chương trình chạy kiểm thử như: chương trình giả lập môi trương, chương trình tạo các trường hợp kiểm thử… Các trương hợp kiểm thử phải đi cùng với kết quả mong muốn, Trong thực tế những thông tin này cũng là một phần của software configuration ở trên.



Hình 1 Quá Trình Kiểm Định

Từ những thông tin đầu vào chương trình được chạy kiểm thử và kết quả của sau bước này sẽ được đánh giá/so sánh với một tập các kết quả mong đợi. khi kết quả của quá trình so sánh thất bại thì một lỗi được phát hiện và quá trình gở lỗi (debugging) bắt đầu. Gở lỗi là một quá trình không thể đoán trước được do một lỗi gây ra bởi sự khác nhau giữa kết quả kiểm thử và kết quả mong đợi có thể tốn một giờ, một ngày hay một tháng để tìm ra nguyên nhân và chỉnh sửa. Và cũng chính sự không chắc chắn cố hữu này mà làm cho quá trình kiểm định rất khó đưa ra một lịch biểu chắc chắn.

Lúc mà kết quả kiểm định được thống kê và đánh giá. chất lượng và độ tin cậy của một phần mềm được ước lượng. Nếu có những lỗi nghiêm trọng xãy ra thường xuyên những lỗi dẫn đến cần phải thay đổi thiết kế của chương trình thì chất lượng của chương trình rất không tốt. Nhưng nếu ngược lại các module/hàm đều hoạt động đúng đắn như thiết kế ban đầu và những lỗi được tìm thấy có thể chỉnh sửa dễ dàng, thì có 2 kết luận có thể được đưa ra :

- Chất lượng của phần mềm là chấp nhận được
- Những kiểm định có thể không thoả đáng/thích hợp để phát hiện ra những lỗi nghiêm trọng đã đề cập trên.

Vậy thì cuối cùng là nều mà quá trình kiểm định phát hiện không có lỗi thì. Ở đây có một chút nghi ngờ rằng những thông tin cầu hình về kiểm thử không đủ và rằng lỗi vẫn tồn tại trong phần mềm. Những lỗi này sẽ được phát hiện sau này bởi người sử dụng và được chỉnh sửa bởi lập trình viên nhưng ở tại giai đoạn bảo trì và chi phí của những công việc này sẽ tăng lên 60 đến 100 lần so với chi phí cho mỗi chỉnh sửa trong giai đoạn phát triển.

Ta thấy rằng chi phí tiêu tốn quá nhiều cho quá trình bảo trì để chỉnh sửa môt lỗi do đó cần phải có những kỹ thuật hiệu quả để tạo được các trường hợp kiểm thử tốt.

7. Kỹ Thuật Thiết Kế

7.1. Khái Quát

Chương này tập trung vào các kỹ thuật để tạo ra các trường hợp kiểm thử tốt và ít chi phí nhất, tất cả chúng phải thoả những mục tiêu kiểm thử ở chương trước. Nhắc lại các mục tiểu kiểm thử phần mềm là thiết kế các trường hợp kiểm thử có khả năng tìm kiếm nhiều lỗi nhất trong phần mềm và với ít thòi gian và công sức nhất.

Hiện tại phát triển rất nhiều phương thức thiết kế các trường hợp kiểm thử cho phần mềm. Những phương pháp này đều cung cấp một hướng kiểm thử có tính hệ thống. Qua trọng hơn nữa là chúng cung cấp một hệ thống có thể giúp đảm bảo sự hoàn chỉnh của các trường hợp kiểm thử phát hiên lỗi cho phần mềm.

Một sản phẩm đều có thể được kiểm thử theo 2 cách:

- Hiểu rõ một chức năng cụ thể của một hàm hay một module. Các trường hợp kiểm thử có thể xây dựng để kiểm thử tất cả các thao tác đó.

- Hiểu rõ cách hoạt động của một hàm/module hay sản phẩm. Các trường hợp kiểm thử có thể được xây dựng để đảm bảo tất cả các thành phần con khớp với nhau. Đó là tất cả các thao tác nội bộ của hàm dựa vào các mô tả và tất cả các thành phần nôi bộ đã được kiểm thủ một cách thoả đáng.

Cách tiếp cận đầu tiên được gọi là kiểm thử hộp đen (black box testing) và cách tiếp cận thứ hai là gọi là kiểm thử hộp trắng (white box testing).

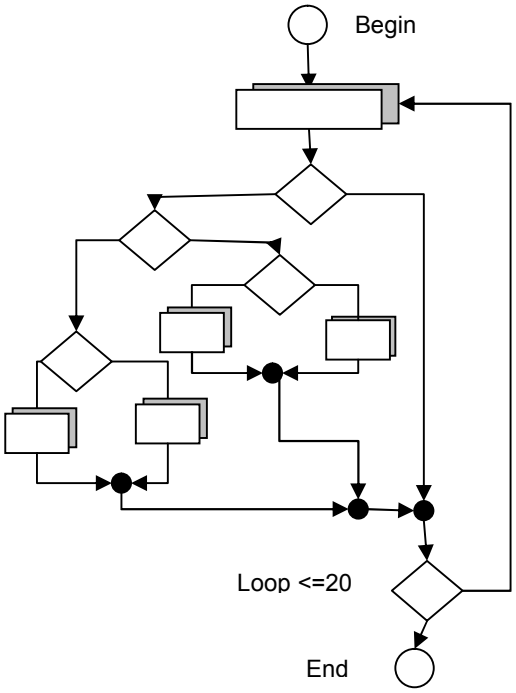
Khi đề cập đến kiểm thử phần mềm, black box testing còn được biết như là kiểm thử ở mức giao diện (interface). Mặc dù thật sự thì chúng được thiết kế để phát hiện lỗi. Black box testing còn được sử dùng để chứng minh khả năng hoạt động của hàm hay module chương trình và có thể cả một chương trình lớn: các thông số đầu vào được chấp nhận như mô tả của hàm, giá trị trả về cũng hoạt đông tốt, đảm bảo các dữ liệu từ bên ngoài ví dụ như file dữ liệu được giữ/đảm bảo tính nguyên vẹn của dữ liệu khi thực thi hàm.

While box testing là kỹ thuật tập trung vào khảo sát chặc chẻ thủ tục một cách chi tiết. Tất cả những đường diễn tiến logic trong chương trình được kiểm tra bằng những trường hợp kiểm thử kiểm tra trên các tập điều kiện và cấu trúc lặp cụ thể. kỹ thuật này sẽ kiểm tra trạng thái của chương trình tại rất nhiều điểm trong chương trình nhằm xác giá trị mong đợi tại các điểm nay có khớp với giá trị thực tế hay không.

Với tất cả các mục tiêu kiểm định trên thì kỹ thuật while box testing có lẽ sẻ dẫn đến một chương trình chính xác tuyệt đối. Tất cả những gì chúng ta cần bây giờ là thiết kế tất cả các đường logic của chương trình và sau đó là cài đặt tất cả các

trường hợp kiểm định có được. Tuy nhiên việc kiểm định một cách thấu đáo tất cả các trường hợp là một bài toán quá lớn và tốn rất nhiều chi phi. Chúng ta hay xem xét ví dụ sau

Hình 2 FlowChart



Bên trái là flowchart cho một chương trình đơn giản được viết bằng khoản 100 dòng mã với một vòng lặp chính thực thi đoạn mã bên trong và lăp lại không quá 20 lần. Tuy nhiên khi tính toán cho thấy đối với chương trình này có đến khoảng 10^{14} đường có thể được thực hiện.

Chúng ta làm tiếp một phép tính nhanh để thấy được chi phí dùng để kiểm thử đoạn chương trình nay một cách thấu đáo và chi tiết. Ta giả sử rằng để kiểm định một trường hợp cần chạy trung bình tồn một giây. Và chương trình kiểm thử sẽ được chạy 24 giờ một ngày và chạy suốt 365 ngày một năm. Vây thì để chạy kiểm thử cho tất cả các trường hợp này cũng cần phải tốn khoản 3170 năm.

Do đó kiểm thử một cách thấu đáo là một việc bất khả thi cho những hệ thống lớn.

Mặc dù kỹ thuật này không thể hiện thực được trong thực tế với lượng tài nguyên có hạn, tuy nhiên với một số lượng có giới hạn các đường diễn tiến logic quan trọng có chọn lựa trước để kiểm thử. Phương pháp này có thể là rất khả thi

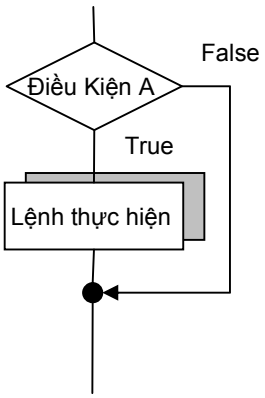
Ngoài ra các trường hợp kiểm thử còn có thể là sự kết hợp của cả hai kỹ thuật trên nhằm đạt được các mục tiêu của việc kiểm thử.

Và bây giờ chúng ta sẽ đi và chi tiết thảo luận về kỹ thuật kiểm thử hộp trắng.

7.2. Kỹ Thuật Thiết Kế Hộp Trắng (White Box)

Trước tiên ta thảo luận một số khái niệm cần thiết cho các phần trình bày sau : Khái niệm một đường diễn tiến của chương trình là một tập hợp lệnh được thực thi có thứ tự.trong chương trình. Để đơn giản hơn có thể hiểu một đoạn chương trình hay một chương trình chứa rất nhiều các đường diễn tiến tại một lênh điều kiện rẽ nhánh tạo ra một tập đường mới

Hình 3 Đường Diễn Tiến



Vi dụ trên ta sẽ có 2 đường một đường khi điều kiện A nhận giá trị đúng và một đường khi điều kiện A mang giá trị sai.

Trong kiểm thử hộp trắng, các trường hợp kiểm thử được thiết kế để xem xét trên cấu trúc nội bộ của module và cấu trúc logic và cấu trúc điều kiển. Các trường hợp kiểm thử sẽ duyệt qua tất cả các lệnh trong chương trình.Tuy nhiên điều này cũng gặp các khó khăn như trình bày ở trên bởi số lượng công việc phải làm. Vậy tại sao ta không tập trung vào chỉ thiết kế các trường hợp kiểm thử dựa trên kỹ thuật kiểm thử hộp đen. Câu trả lỏi nằm trong nhưng yếu điểm tự nhiên của phần mềm.

- Những lỗi về lý luận và những giả sử không chính xác có xác xuất xảy ra tương đương vói những trường hợp đúng. Những lỗi có khuynh hướng xuất hiện khi chúng ta thiết kế và cài đặc chương trình, các biểu thức điều kiện, hoặc các biểu thức điều kiển, và các lỏi thường có khuynh hướng xuất hiện ở các trường hợp đặc biệt.

- Chúng ta thường tin rằng một đường diễn tiến nào đó sẽ không được thực thi. Tuy nhiên thực tế thì nó có thể được thực thi. Luồng diễn tiến của chương trình đôi khi chỉ là mang tính trực giác, có thể hiểu là một giả định tưởng tượng của người lập trình về luồng điều kiển và dữ liệu đã làm cho chúng ta tạo ra lỗi. Lỗi loại này có thể được phát hiện bằng một trương hợp kiểm thử trên một đường diễn tiến.

- Những lỗi về cài đặt sai do lỗi gõ phím là ngẫu nhiên và có thể xuất hiên tại bất kỳ đâu trong chương trình. Khi một chương trình được chuyển đổi từ ý tưởng thiết kế sang thành mã chương trình. một số lỗi do dánh sai hiểu sai xuất hiện. Phần lớn có thể được phát hiện bỏi những hệ thống kiểm tra cú pháp của ngôn ngữ, nhưng một số khác sẽ không được phát hiên cho đến khi chạy kiểm thử.

Mỗi một lý do giải thích tại sao phải tạo ra các trường hợp kiểm thử dựa trên kỹ thuật hộp trắng. Hộp đen củng được nhưng có thể một số loại lỗi ở trên sẽ không được phát hiện bởi các trường hợp sử dụng phương pháp này..

. Vậy cho nên thiết kế các trường hợp kiểm thử này cần phải xem xét đến sự cân bằng giữa mức độ kiểm định và khả năng hiện thực của thiết kế. Phần sau là những cấp độ kiểm định dựa trên kỹ thuật kiểm thử hộp trắng.

7.2.1. Kiểm Thử Đường Diễn Tiến Của Chương Trình

Đây là khái niệm chỉ đến việc thiết kế các trượng hợp kiểm thử trên từng lệnh trong chương trình sẽ thực hiện ít nhất một lần. Kỹ thuật này không quan tâm đến ảnh hưởng lên các đường quyết định (decisions path).

Các bước để xây dựng tập hợp kiểm thử có thể theo những bước sau đây.

1. Dùng tài liệu thiết kế hay source code để vẽ ra một đồ thị mô tả flow chart của chương trình hay hàm.

2. Xác định đồ thị V(G).

3. Từ đồ thị xác định tập đường độc lập tuyến tính lẫn nhau.

4. Xây dựng những trường hợp kiểm thử dựa trên tâp đường xác định ở bước trên.

Ví Du

Tất cả các lệnh được thực thi sẽ nằm trên một đường thực thi của chương trình. Trong phần này chúng ta xét thủ tục **average**

1) Phân tích thủ tục **average**

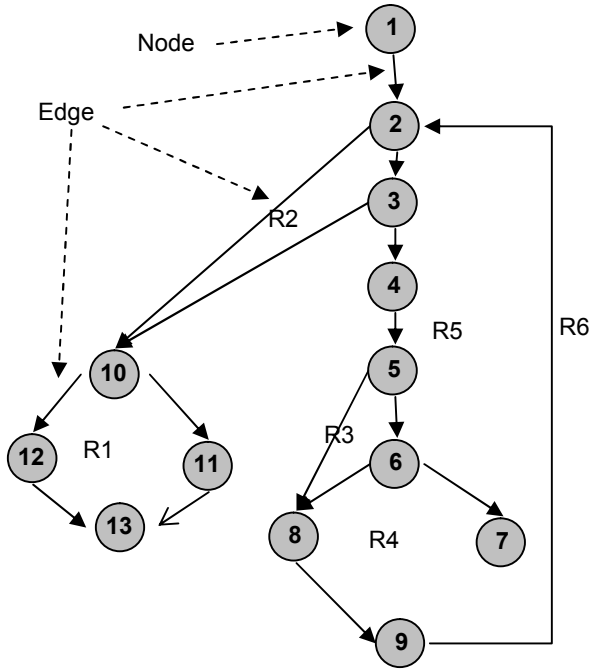
Hình 4 Mã Lệnh Của Thủ Tục average

```
Int average( )
{
        /* Hàm tính giá trị trung bình của 100 hoặc có thể ít hơn các số
*/
        int value[100];
        int average,totalinput, totalvalid,
        minimum, maximum, sum;
        int i;

1       i=0;
        totalinput = totalvalid =0;
        sum = 0;

4       while( value[i]<>-999 and totalinput <100
                totalinput ++;
5               if( value[i] >= minimum AND value[i] < maximum)
                {
7                   totalvalid  ++;
                    sum = sum + value[i]
                }
                i++;                                  8

9       )

10      if( totalvalid > 0 ){
                average = sum/totalvalid              11
        }
        else{
                average = -999                        12
        }
}
13
```

Mã lệnh của thủ tục được phân tích và biểu diễn thành dạng flowchart tương ứng như bên trong đoạn mã. Ở đay chương trình có 13 để điểm biểu diễn trong đồ thị.

2) Tạo một đồ thị flow graph biểu diễn tương ứng với flow chart của hàm **average**

Hình 5 Flow Graph Của Thủ Tục average



3) Trong trường hợp này ta có thể xác định có 6 trường hợp kiểm thử như sau :

Đường 1 : 1-2-10-11-13
Đường 2 : 1-2-10-12-13
Đường 3 : 1-2-3-10-11-13
Đường 4 : 1-2-3-4-5-8-9-2 ...
Đường 5 : 1-2-3-4-5-6-8-9-2 ...
Đường 6 : 1-2-3-4-5-6-7-8-9-2 ...

Ba chấm sau những đường 4, 5, 6 cho biết rằng một đường đi bất kì qua phần còn lại của câu trúc điều kiển đều được chấp nhận.

4) Tạo các trường hợp kiểm định dựa trên 6 đường trên.

Bảng 1 Các Trường hợp kiểm định

	Mô Tả
1	value[k] = là một giá trị hợp lệ, với k < i value[i] = -999 vơi một giá trị 2<=i <=100 Giá trị mong đợi Giá trị trung bình mong đợi là giá trị trung bình đúng của tất cả giá trị k.
2	value[1] = -999 Giá trị mong đợi : Giá trị trung bình mong đợi là giá trị trung bình -999, những biến chứa giá trị tổng cộng khác đều bằng 0.

	Mô Tả
3	Cố gắng thực hiện tiến trình cho 101 giá trị hoặc hơn, 100 giá trị đầu trong value là những giá trị hợp lệ Giá trị mong đợi : giống như trường hợp 1
4	value[i] = giá trị hợp lệ khi i< 100 value[k]< minimum khi k < i Giá trị mong đợi : Giá trị trung bình mong đợi là giá trị trung bình đúng trên k giá trị và tổng nhận giá trị đúng
5	value[i] = là một giá trị hợp lệ khi i <100 value[k] > maximum khi k<=i Giá trị mong đợi : Giá trị trung bình mong đợi là giá trị trung bình đúng trên n giá trị và tổng nhận giá trị đúng
6	value[i] = là một giá trị hợp lệ khi i <100 Giá trị mong đợi : Giá trị trung bình mong đợi là giá trị trung bình đúng trên n giá trị và tổng nhận giá trị đúng

Mỗi trường hợp được chạy và so sánh với kết quả mong đợi. Nếu tất cả các trường hợp kiểm định đều cho kết quả như mong muốn thì có thể khẳng định rằng tất cả các dòng lệnh trong thủ tục average đều được kiểm thử ít nhất một lần.

7.2.2. Kiểm Định Cấu Trúc Điều Kiển

a) Kiểm thử các biểu thức điều kiện

Kiểm thử biểu thức điều kiện là phương pháp kiểm thử trên những điều kiện logic của hàm hay module. Một điều kiện đơn giản là một biến boolean hoặc là một biểu thức quan hệ:

- X hay Not X một điều kiện logic đơn giản.
- Biểu thức quan hệ thường có dạng : E1 <phép toán quan hệ> E2

  E1, E2 là các biểu thức số học và phép toán quan hệ là một trong các phép toán sau : <, <=, ==, != , > hay >=. Một điều kiện kết hợp của 2 hay nhiều điều kiện đơn giản, các phép toán boolean : OR (| |, AND (&) and NOT (!)

(1) Các loại lỗi của điều kiện bao gồm

Lỗi trong các thao tác luận lý (lỗi tồn tại một biểu thức không đúng, thiếu hoặc thừa các thao tác luận lý

- Lỗi do giá trị của biến luận lý

- Lỗi do dấu ngoặc
- Lỗi do phép toán quan hệ
- Lỗi trong biểu thức toán học

Mục đích của kiểm thử cấu trúc điều kiển là phát hiện không chỉ lỗi trong điều kiện mà còn những lỗi khác trong chương trình. Nếu một tập kiểm thử cho một chương trình P là hiệu quả cho việc phát hiện lỗi trong điều kiện của P,thì bộ kiểm thử đò cũng có thể phát hiện các lỗi khác trong P.

E1 <phép toán quan hệ> E2

Ba trường hợp kiểm thử được yêu cầu để kiểm tra là giá trị E1 lớn hơn, nhỏ hơn và bằng giá trị của E2. Nếu <phép toán quan hệ> là không đúng và E1, E2 là đúng thì 3 loại kiểm thử trên có đảm bảo có thể xác định được lỗi trong phép toán quan hệ. Để phát hiện lỗi trong E1và E2 thì các trường hợp kiểm thử E1 lớn hơn, nhỏ hơn E2 có thể phát hiện ra được lỗi.

Một biểu thức có n biến, thì có 2^n khả năng kiểm thử xãy ra khi (n>0)

b) Kiểm Thử luồng Dữ liệu (DFT)

Phương pháp kiểm thử luồng dữ liệu chọn lựa một số đường diễn tiến của chương trình dựa vào việc cấp phát, định nghĩa, và sư dụng những biến trong chương trình.

Để hình dung ra cách tiếp cận này ta giả sử rằng mỗi câu lệnh của chương trình được gán một số duy nhất và rằng mỗi hàm khong được thay đổi thông số của nó và biến toàn cục.

DEF(S) = { X | lệnh S chứa định nghĩa X }
USE(S) = { X | lệnh S chứa một lệnh/biểu thức sủ dụng X }

Nếu S là câu lệnh if hay loop, thì tập DEF của S là rỗng và USE là tập dựa trên điều kiện của câu lệnh S.

Định nghĩa 1 biến X tại câu lênh S được cho là vẫn còn sống tại câu lênh S' nếu như tồn tại một đường từ câu lệnh S đến câu lệnh S' không chứa bất kỳ định nghĩa nào của X.

Định nghĩa 2 Một chuỗi dùng của biến X (gọi là DU của X) ký hiệu [X, S, S'] là định nghĩa của X trong câu lệnh S vẫn sống trong câu lênh S'.

Phương pháp kiểm thử luồng dữ liệu yêu cầu rằng tất cả các chuỗi DU đều được kiểm thử ít nhất một lần. Có thể thấy rằng bộ kiểm thử cho luồng dữ liệu có thể không bao trùm tất cả các nhánh của chương trình. Tuy nhiên nêu môt nhánh đảm bảo được sẽ được phát hiện bỏi phương pháp kiểm thử này. Trong một số hiếm

trường hợp như là cấu trúc lệnh if-then trong phần then không có định nghĩa thêm một biến nào và phần else không tồn tại. Trong tình huống này thì nhánh else của câu lênh ì trên không cần thiết phải bảo hộ bởi phương pháp này.

DFT rất hữu ích cho các loài kiểm thửmột chương trình có nhiều lệnh if và lệnh lặp lồng nhau nhiều cấp.

Ví Dụ

Hình 6 Một Thủ Tục Với Lệnh Điều Kiện Và Lệnh Lặp Phức Tạp

```
proc x
        B1;
        do while C1
                if C2
                then
                        if C4
                        then B4;
                        else B5
                        endif;
                else
                        if C3
                        then B2
                        else B3
                        endif;
                endif
        enddo

        B6

End proc
```

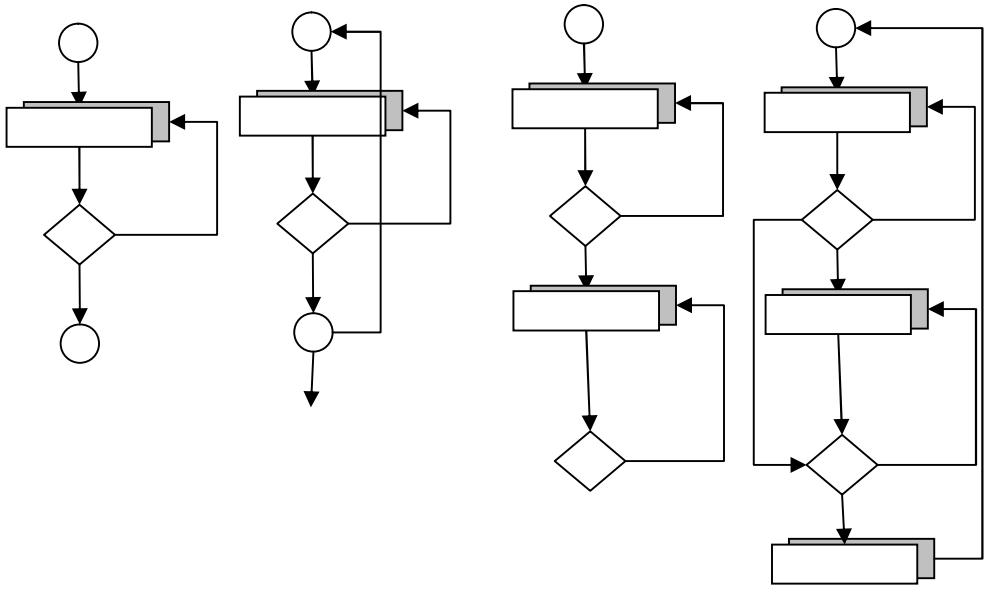
Để xây dựng các trường hợp kiểmthử DFT cho thủ tục trên, chúng ta cần phải biết định nghĩa và sử dụng biến ở mỗi điều kiện hoặc một khối trong thủ tục này. Giả sử biến X được định nghĩa trong câu lệnh cuối của khối lệnh B2, B3, B4 và B5. và biến X được sử dụng ở đầu của các khối B2, B3, B4, B5 và B6. Kiểm thử DU yêu cầu đường thực thi ngắn nhất từ Bi, 0< i <= 5 đến Bj 1<j<=6.(thật sự thì trong trường hợp này các trường hợp kiểm thử cũng có khả năng phát hiện bất kỳ việc dùng biến X trong các điều kiện C1, C2, C3 và C4) mặc dù có đến 25 chuỗi DU nhưng chỉ cần 5 là đủ để bao hàm các trường hợp khác.

c) Kiểm Thử Vòng Lặp

Vòng lặp là một trong những nền tảng cho rất nhiều các thuật toán được cài đặc trong các phần mềm. tuy nhiên cho đến lúc này chúng ta vẫn còn ít chú ý đến việc xây dựng các trương hợp để kiểm thử.

Kiểm thử vòng lặp tập trung vào tính chất của cấu trúc vòng lặp. Có 4 cầu trúc vòng lặp như sau: vòng lặp đơn giản, vòng lặp móc nối, vòng lặp tạo thành tổ, và vòng lặp không cầu trúc

Hình 7 Các Cấu Trúc Lặp



vòng lặp đơn giản | vòng lặp tạo thành tổ | vòng lặp móc nối | vòng lặp không cầu trúc

(1) Vòng Lặp Đơn

Tập hợp tiếp theo là các trường hợp kiểm thử cho vòng lặp đơn, với n là maximum số lần lặp.

- Bỏ tình toàn vẹn của vòng lặp
- Chỉ cần một lần duyệt xuyên qua cả vòng lặp
- Hai lần duyệt xuyên qua cả vòng lặp
- m lần duyệt xuyên qua cả vòng lặp
- n-1, n, n+1 lần duyệt xuyên qua cả vòng lặp

(2) Vòng Lặp Tạo Tổ

Nếu như chúng ta mở rộng phương pháp kiểm thử cho vòng lặp đơn thì số lượng trường hợp kiểm thử sẽ tăng rất nhiều. Sau đây là một cách là giảm sồ lượng trường hợp kiểm thử :

- Bắt đầu tại vòng lặp con trong cùng. Thiết lập tất cả các vòng lặp khác là giá trị minimum.

- Kiểm soát vòng lặp ở trong cùng trong khi giữ các vòng lặp bên ngoài lặp lại với giá trị là minimum thông số ảnh hưởng nhau (thông số đó có thể là biến lặp). Thêm môt số trường hợp ngoài phạm vi của biến lặp và một số giá trị đặc biệt.

- Thực hiên như bước trên và tiến ra ngoài dần

- Thực hiện tiếp cho đến khi tất cả các vòng lặp được kiểm thử hết.

(3) Vòng Lặp Móc Nối

Đồi vói kiểu này có thể kiểm thử bằng cách như với vòng lặp đơn ở trên nếu các biền lặp độc lập với nhau. Tuy nhiên nếu 2 vòng lặp là móc nối và biến lặp của vòng lặp thứ nhất được sử dụng như là biến khởi tạo cho vòng lặp 2 thì 2 vòng lặp này không còn độc lặp nữa, Phương pháp dùng cho vòng lặp tạo tổ sẽ được sử dụng ở đây.

(4) Vòng Lặp Không Có Cấu Trúc

Khi nào gặp các cầu trúc lặp như vầy thì nên thiết kế lại. Việc kiểm thử rất phức tạp.

7.3. Kỹ Thuật Kiểm Thử Hộp Đen (Black Box)

Là phương pháp tập trung vào yêu cầu về mặt chức năng của phần mềm. Có thể tạo ra một bộ các điều kiện các input để kiểm thử tất cả các chức năng của một chương trình. Kiểm thử hộp đen về bản chất không phải là môt phương pháp trái ngược với kiểm thử hộp trắng. Đúng hơn đây là phương pháp bổ xung cho phương pháp kiểm thử hộp trắng để phát hiện tất cả các loại lỗi khác nhau nhiều hơn là phương pháp kiểm thử hộp trắng đã biết.

Kiểm thử hộp đen cố gắng phát hiện các loại lỗi như sau:

- Không đúng hay mất môt số hàm/module
- Giao diện không phù hợp/ lỗi về interface.
- Lỗi về cấu trúc dữ liệu hay thao tác lên data bên ngoài.
- Lỗi thực thi.
- Lỗi về khởi động và huỹ dư liệu, biến.

Không giống như phương pháp kiểm thử hộp trắng có thể được thực hiện ỏ những giai đoạn đầu của quá trình kiểm thử phần mềm, Phương pháp này tập trung vào phần sau của quá trình kiểm thử. Mục đích của quá trình kiểm thử là tập trung trên vùng thông tin chư không phải trên vùng mã chương trình. Các trường hợp kiểm thử để trả lời các câu hỏi sau:

- Như thế nào là hàm/chức năng hợp lệ?
- Lớp gì của thông tin đầu vào sẽ tạo ra những trương hợp kiểm thử tốt ?
- Hệ thống có khả năng bị thương tổn với một giá trị nhập vào nào đó không?
- Ranh giới của các vùng dữ liệu có đôc lập với nhau hay không ?
- Tỷ lệ và kích thước dữ liệu mà hệ thống có thể hứng chiệu là bao nhiêu?

7.3.1. Phân Vùng Tương Đương

Đây là kỹ thuật chia vùng thông tin nhập vào của chương trình thành các lớp thông tin/dữ liệu. Lớp tương đương biểu diễn thành một tập các giá trị hợp lệ và không hợp lệ. Nhưng lớp dữ liệu tương đương này có thể được xác định theo những cách sau:

- Nếu thông tin đầu vào chỉ định một vùng các giá trị, thì ta có một lớp dữ liệu hợp lệ và hai không hợp lệ được định nghĩa.

- Nếu thông tin đầu vào chỉ định một giá trị, thì ta có một lớp dữ liệu hợp lệ và hai không hợp lệ được định nghĩa.

- Nếu thông tin đầu vào chỉ định một giá trị của một tập, thì ta có một lớp dữ liệu hợp lệ và hai không hợp lệ được định nghĩa.

- Nếu thông tin đầu vào chỉ định một giá trị boolean, thì ta có một lớp dữ liệu hợp lệ và một không hợp lệ được định nghĩa.

Ví Dụ
Một khách hàng có thể liên lạc với ngân hàng bằng máy tính cá nhân, họ gởi một mật khẩu gồm 6 chử số và các thao tác khởi động một số chức năng của ngân hàng. Phần mềm hổ trợ cho các ứng dụng của ngân hàng chấp nhận dữ liệu theo dạng sau:

Mã vùng - rỗng hay 3 chử số
Tiền tố - 3 chử số không bắt đầu bằng 0 hay 1.
Hậu tố - 4 chử số
Mật khẩu – 6 ký tự alphanumberic
Thao tác/nghiệp vụ ngân hàng – "xemtàikhoản", "gởitàikhoản" , "rúttàikhoàn" …

Áp dụng kỹ thuật phân vùng thông tin để tạo các bộ kiểm thử như sau :

Bảng 2 Các Trường Hợp Kiểm Định

Dữ Liêu vào	Mô Tả
Mã vùng	Boolean – giá trị của mã vùng có thể được nhập hoặc không Range – giá trị của mã vùng có thể được định nghĩa từ 200 đến 999.
Tiền tố	Range – Xác định các giá trị >200
Hậu tố	Value - một chuỗi 4 số
Mật khẩu	Boolean – giá trị của mật khẩu có thể được nhập hoặc không Value – giá trị nhận là một chuỗi 6 ký tự
Thao tác/nghiệp vụ ngân hàng	Set - một tập các thao tác được ghi bên trên.

7.3.2. Phân Tích Giá Trị Biên

Thực tế thì phần lớn các lỗi có khuynh hướng xuất hiện tại biên của vùng thông tin đầu vào hơn là ở tại những vị trí ở giữa vùng. Do đó kỹ thuật phân tích giá trị biên (BVA) được phát triển. kỹ thuật này sẽ lựa chọn một số trường hợp kiểm thử tại các giá trị biên và là kỹ thuật bổ xung cho kỹ thuật phân vùng tương đương ở trên, hơn là việc chọn một giá trị bất kỳ trong vùng này.

Nguyên tắc của BVA có phần tương tự với phương pháp phân vùng thông tin:

1. Nếu điều kiện đầu vào xác định một phạm vi được chỉ định bởi 2 giá trị a và b, những trường hợp kiểm thử sẽ được thiết kế tại các giá trị biên a và b, và trên a và dưới b.

2. Nếu điều kiện đầu vào xác định một phạm vi được chỉ định bởi tập hợp nhiều giá , những trường hợp kiểm thử sẽ được thiết kế tại các giá trị biên min và max của tập hợp đó, các giá trị lớn hơn max và nhỏ hơn min cũng được kiểm thử.

3. Áp dụng 1,2 cho giá trị trả về

7.3.3. Kỹ Thuật Cause-Effect Graphing

Ta thấy rằng 2 kỹ thuật trên dữ liệu đầu vào đã được phân loại để phân tích. Tuy nhiên kỹ thuật sắp trình bày dưới đây cho phép xác định ra các trường hợp kiểm thử hiểu quả nhất ngay cả trong lúc dữ liệu đầu vào là khó phân loài thành các lớp như trong 2 kỹ thuật trên.

Kỹ thuật này gồm có 4 bước như sau :

1. Xác định Cause (điều kiện nhập vào) và effect (là hành động) cho mỗi một module cần kiểm định.

2. Xây dựng đồ thị cause-effect:

3. Đồ thị được chuyển thành bảng quyết định

4. Những phần/luật trong bảng quyết định được chuyển thành các trường hợp kiểm thử.

Ví Dụ

Mô tả của một module:

Số lượng vần đề cần kiểm tra là 5.

Kết quả

“Pass” - nếu kết quả của 4 hay nhiều hơn 4 chủ để là thành công.

“Temporary pass” - nếu kết quả của 2 hay 3 chủ để là thành công

“Failure” - nếu kết quả có duy nhất 1 chủ để là thành công

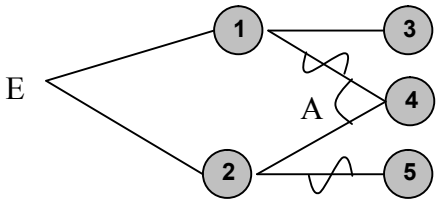
Bước 1: Xác định quan hệ giữa input & output của module trên

Bảng 3 Mối Quan Hệ Giữa input & output

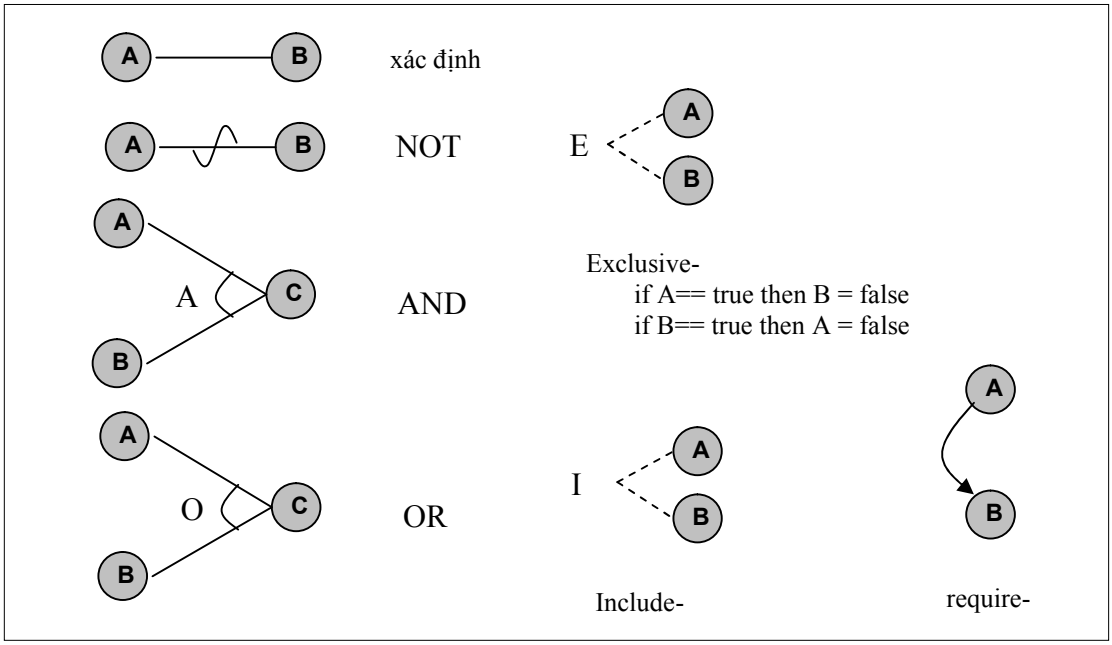
Cause(Dữ Liệu Nhập)	Result (Dữ Liệu Xuất)
1. Nếu kết quả của 4 hay nhiều hơn 4 chủ đề là thành công 2. Nếu kết quả của 2 hay 3 chủ đề là thành công	4. Pass 5. Temporary Pass 6. Failure

Bước 2 Biểu diễn quan hệ giữa cause và result trên đồ thị

Hình 8 Một Đồ Thị cause - effect



Hình 9 Một Số Ký Hiệu Sử Dụng Trong Đồ Thị cause - effect



Bước 3 Tạo bảng quyết định

Bảng 4 Bảng Quyết Định

Cause & Result		T1	T2	T3
Cause	1. Pass 4 hay nhiều hơn 4 chủ đề	Y	N	N
	2. “Pass” 2 hay nhiều hơn 2 chủ đề	-	Y	N

Result	3. “Pass” là kết quả xuất ra	X	-	-
	4. “Temporary Pass” là kết quả xuất ra	-	X	-
	5. “Failure” là kết quả xuất ra	-	-	X

Chú thích cho bảng quyết định :
Dòng chỉ định điều kiện: mỗi dòng bao gồm các điều kiện để quyết định cho chương trình: (đây là các dòng có màu xâm hơn trên bảng)

Y : true, N false, | không có quyết định nào

Dòng chỉ định hành động: mỗi dòng chỉ định tiến trình có được thực thi hay không:(đây là các dòng có màu sáng hơn trên bảng)

X : tiến trình hoạt động, | không có tiến trình nào hoạt động cả

8. Chương Trình Minh Hoạ

Dựa vào các kỹ thuật đã trình bày ở phần trên, phần này xây dưng chương trình. Tập trung vào thiết kế xây dựng một mô hình để kiểm định chương trình C/C++ và một chương trình minh hoạ có những chưc năng chính như sau.

Những Chức Năng Chính

- Xây dựng một chương trình có khả năng tạo tự động mốt số trường hợp kiểm thử một chương trình C/C++
- Kết quả của chương trình là một chương trình C/C++ chứa các testcase dùng để kiểm thử một số hàm của chương trinh bằng cách dùng phương pháp kiểm thử hộp đen trên các dư liệu nhập vào ở dạng chuẩn của C/C++

Ví Dụ Của file kết quả sau khi chạy để tạo testcase tự động

```
//  (C) Unit Test Generator 2002-2004.
//   The file is automatically generated by the program Unit Test Generator 1.0

// Boost.Test
#include <boost/test/unit_test.hpp>
using boost::unit_test::test_suite;

void free_test_function()
{
    BOOST_CHECK(2 == 1);
    int* p = (int*)0;
    *p = 0;
}

test_suite* init_unit_test_suite( int, char* [] ) {
    test_suite* test= BOOST_TEST_SUITE( "Unit test example 1" );

    test->add( BOOST_TEST_CASE( &free_test_function ), 1 /* expected one error */
);

    return test;
}

// EOF
```

Phụ Lục A Bảng Chú Giải

Testing SoftWare	Kiểm thử phần mềm
Test case	Một trường hợp kiểm thử.
Test case design	Quá trình thiết kế các trường hợp kiểm thử.

Phụ Lục B Yêu Cầu Hệ Thống

Những thông tin trong phụ lục này xác định yêu cầu tối thiểu để chạy được chương trình minh hoạ.

Cấu Hình PC

CPU Intel Pentium 100MHz.or faster with the following:

- 32MB memory
- 3.5 inch, 1.44 diskette drive
- CD ROM drive
- 500MB hard disk (50 MB free space recommended; for more information, refer to “Hard Disk Requirement” on page xx)
- SVGA graphics monitor with interface card (the resolution is 800x600)
- Windows 2000 software

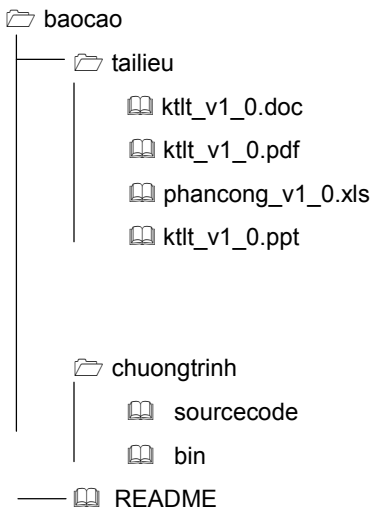
Bảng 5 Dung Lượng Của Chương Trình Minh Hoạ

Thành Phần	Kích Thước
Program	
Source code for unit test framework	200k
unittestgen_v1.0	

Phụ Lục C Cấu Trúc Thư Mục

Cấu Trúc Thư Mục

Hình 10 Cấu Trúc Thư Mục của báo cáo



- **baocao:** Đây là thư mục gốc của báo cáo.
- **tailieu:** Thư mục chứa tài liệu này ở dạng PDF và Ms Word
- **phancong_v1.0.xls** :phân công là xls chứa phân công các công việc trong nhóm.
- **ktlt_v1.0.ppt** báo cáo tóm lược dưới dạng MS -PowerPoint
- **chuongtrinh:** Thư mục này chứa source code của chương trình minh hoạ và chương trình sau khi đã biên dịch thành dạng nhị phân.
  - **source:**
  - **bin:** .

Tài Liệu Tham Khảo

[1] [BEI90] Beizer, B.,Software Testing Techniques, 2d ed., Van Nostrand Reinhold, 1990,

[2] [DEU79] Detsch, M., "Verification and Validation" in software Engineering,(R. Jensen and C. Tonies, eds.) Prentice-Hall, 1979, pp 329-408.

[3] Software Engineering A Practitioner's Approach, Roger S. Pressman.

[4] GT.TSKH. Hoàng Kiếm, Bài Giảng Nguyên Lý và Phương Pháp Lập Trình, 2004

[Blank PAGE]