ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

o0o

BÁO CÁO MÔN HỌC

HỌC PHẦN: KỸ THUẬT LẬP TRÌNH

Đề tài: Tìm hiểu những điều thú vị liên quan đến kỹ thuật lập trình

Các kỹ thuật kiểm thử - Testing

Sinh viên thực hiện : Lê Kiều Anh

MSSV : 20193979

Lớp : KHMT 01-K64

Mã lớp : 128698

Giảng viên hướng dẫn : TS. Vũ Đức Vượng

Hà Nội, tháng 01 năm 2022

Mục lục

I. Tổng quan về kiểm thử (testing)

1.1. Khái niệm kiểm thử

Kiểm thử phần mềm là một tiến trình hay một tập hợp các tiến trình được thiết kế để đảm bảo mã hóa máy tính thực hiện theo cái mà chúng đã được thiết kế để làm, và không thực hiện bất cứ thứ gì không mong muốn. Đây là một pha quan trọng trong quá trình phát triển hệ thống, giúp cho người xây dựng hệ thống và khách hàng thấy được hệ thống mới đã đáp ứng yêu cầu đặt ra hay chưa

1.2. Phân biệt testing và debugging

Testing và debugging thường đi cùng với nhau, testing tìm ra lỗi, debugging định vị lỗi và sửa chúng. Quá trình testing và debugging thường nối tiếp nhau và lặp lại, gọi là mô hình "testing/debugging cycle". Bất kỳ một debugging nào nên được tiếp theo là một sự áp dụng lại của hàng loạt các tests liên quan, đặc biệt là các bài tests hồi quy. Điều này giúp tránh nảy sinh các lỗi mới khi debugging.

Phân biệt testing và debugging:

Testing	Debugging
Thực hiện bởi testers	Được thực hiện bởi dev hoặc dev team
Có thể được thực hiện bằng tay hoặc tự động	Chỉ có thể được thực hiện bằng tay
Có thể được xác định trước khi bắt đầu testing. Kết quả kiểm tra có thể được dự đoán	Bắt đầu với các điều kiện chưa biết và thật khó để dự đoán kết quả
Tìm lỗi lập trình	Chứng minh rằng đó chỉ là một lỗi nhỏ không được giám sát
Có thể được thực hiện tự động bằng cách sử dụng các công cụ kiểm tra tự động	Tự động sửa lỗi phần mềm vẫn là mơ ước của các lập trình viên
Mục đích là để tìm lỗi	Mục đích là để tìm ra nguyên nhân của một lỗi

II. Các kỹ thuật kiểm thử

2.1. Internal testing

Là các kỹ thuật thiết kế dữ liệu để test chương trình.

2.1.1. Boundary testing

Đây là một trong những kỹ thuật kiểm thử phần mềm, trong đó các testcase được thiết kế bao gồm các giá trị tại các biên. Nếu dữ liệu đầu vào được sử dụng là trong giới hạn giá trị biên, nó được cho là Positive testing. Nếu dữ liệu đầu vào được sử dụng là ngoài giới hạn giá trị biên, nó được cho là Negative testing. Mục tiêu là lựa chọn các test case để thực thi giá trị biên.

Một số quy tắc chung khi thực hiện kỹ thuật này:

1. Nếu một trạng thái đầu vào định rõ giới hạn của các giá trị, hãy viết ra các ca kiểm thử cho các giá trị cuối của giới hạn, và các ca kiểm thử đầu vào không hợp lệ cho các trường hợp vượt ra ngoài phạm vi.
2. Nếu một trạng thái đầu vào định rõ số lượng giá trị, hãy viết các ca kiểm thử cho con số lớn nhất và nhỏ nhất của các giá trị và một giá trị trên, một giá trị dưới những giá trị này.
3. Sử dụng nguyên tắc 1 cho mỗi trạng thái đầu vào, nguyên tắc 2 cho mỗi trạng thái đầu ra.
4. Nếu đầu vào hay đầu ra của một chương trình là tập được sắp thứ tự, tập trung chú ý vào các phần tử đầu tiên và cuối cùng của tập hợp.

Các case chuẩn thường được lựa chọn dựa vào quy tắc sau:

Case 1: giá trị biên nhỏ nhất -1

Case 2: giá trị biên nhỏ nhất

Case 3: giá trị biên lớn nhất

Case 4: giá trị biên lớn nhất +1

2.1.2. Statement testing

Trong kỹ thuật statement testing, mọi câu lệnh trong cấu trúc code sẽ thực thi ít nhất một lần. Qua đó, tester có thể test được cách vận hành của toàn bộ source code (mã nguồn) phần mềm. Tuy nhiên, tester không thể kiểm thử điều kiện sai mà chỉ có thể thực thi các điều kiện đúng.

Ví dụ: Nhập vào 2 số a, b. Tìm tổng của 2 số, nếu tổng này lớn hơn 0 thì in ta màn hình “Tổng dương”, nếu tổng này nhỏ hơn hoặc bằng 0 thì in ra màn hình “Tổng âm”.

```
printSum (int a, int b){
        int sum = a + b;
        if (sum > 0) printf (“Tổng dương”);
        else printf (“Tổng âm”);
}
```

Statement testing: phải chắc chắn rằng lệnh if và các lệnh bên trong khối lệnh if đều được thực hiện.

Xây dựng testcases:

Testcase 1: a = -1; b = 0: sum = -1 < 0, thực hiện lệnh in ra màn hình “Tổng âm”.

Testcase 2: a = 1; b = 0: sum = 1 > 0, thực hiện lệnh in ra màn hình “Tổng dương”.

2.1.3. Path testing

Kiểm tra để đáp ứng các tiêu chuẩn đảm bảo rằng mỗi đường dẫn logic xuyên suốt chương trình được kiểm tra. Thường thì đường dẫn xuyên suốt chương trình này được nhóm thành một tập hữu hạn các lớp . Một đường dẫn từ mỗi lớp sau đó được kiểm tra.

Nguyên tắc:

1. Với các chương trình đơn giản, có thể liệt kê các nhánh đường dẫn xuyên suốt code.
2. Ngược lại, bằng các đầu vào ngẫu nhiên tạo các đường dẫn theo chương trình.

Ví dụ: Nhập vào 2 số a, b. Tìm tổng của 2 số, nếu tổng này lớn hơn 0 thì in ta màn hình “Tổng dương”, nếu tổng này nhỏ hơn hoặc bằng 0 thì in ra màn hình “Tổng âm”. Tìm tích của 2 số a, b. Nếu tích lớn hơn 0, in ra màn hình “Tích dương”, ngược lại in “Tích âm”.

```
int main(){
        int a, int b;
        int sum = a + b, product = a*b;
```

```
    if (sum > 0) printf ("Tổng dương"); //(1)
    else printf ("Tổng âm");
    if (product > 0) printf ("Tích dương"); //(2)
    else printf ("Tích âm");
    return 0;
}
```

Xây dựng testcases sao cho:
- Điều kiện (1) đúng, điều kiện (2) đúng: a = 1, b = 2
- Điều kiện (1) đúng, điều kiện (2) sai: a = -1, b = 2
- Điều kiện (1) sai, điều kiện (2) đúng: a = -1, b = -2
- Điều kiện (1) sai, điều kiện (2) sai: a = -2, b = 1

2.1.4. Stress testing

Tiến hành thử nghiệm để đánh giá một hệ thống hay thành phần tại hoặc vượt quá các giới hạn của các yêu cầu cụ thể của nó. Đây là một loại kiểm thử xác định sự ổn định và tính mạnh mẽ của hệ thống.

Khi tiến hành stress test, một môi trường bất lợi được cố tình tạo ra và duy trì. Các hành động liên quan có thể bao gồm:

- Chạy một số ứng dụng sử dụng nhiều tài nguyên trong máy tính cùng một lúc.
- Cố gắng hack vào máy tính và sử dụng nó để phát tán thư rác.
- Làm tràn ngập một máy chủ bằng các e-mail vô dụng.
- Thực hiện nhiều nỗ lực đồng thời để truy cập vào một trang web.
- Cố gắng lây nhiễm virus, Trojan, phần mềm gián điệp hoặc phần mềm độc hại khác vào hệ thống.

Tình trạng bất lợi được làm cho xấu dần đi cho đến khi mức hiệu suất giảm xuống dưới một ngưỡng tối thiểu nhất định hoặc hệ thống hoàn toàn ngưng hoạt động. Để có được kết quả hữu ích nhất, các yếu tố riêng lẻ sẽ được lần lượt thay đổi từng cái một. Điều này giúp có thể dễ dàng xác định các điểm yếu và lỗ hổng cụ thể.

2.2. External testing

Là các kỹ thuật xây dựng chương trình để nó tự test chính nó.

2.2.1. Kiểm tra bất biến

- Thử nghiệm các điều kiện trước và sau.
- Các khía cạnh của cấu trúc dữ liệu không được thay đổi. Một hàm khi tác động vào cấu trúc dữ liệu phải kiểm tra tính bất biến ở đầu và cuối của nó.

2.2.2. Kiểm tra các thuộc tính lưu trữ

- Là trường hợp khái quát hóa của kiểm tra bất biến.
- Một hàm cần kiểm tra các cấu trúc dữ liệu bị tác động tại các điểm đầu và cuối.

2.2.3. Kiểm tra các giá trị trả về

- Kiểm tra giá trị trả về của hàm.
- Trong C++, chúng ta có thể sử dụng kỹ thuật bắt ngoại lệ “try-catch”.

2.2.4. Tạm thay đổi code

- Tạm thời thay đổi đoạn code để tạo ra các giới hạn hoặc stress test.

III. Phương pháp kiểm thử

3.1. Black-box testing

3.1.1. Định nghĩa

Kiểm tra hộp đen (Black box testing) là một phương pháp kiểm thử phần mềm mà việc kiểm tra các chức năng của một ứng dụng không cần quan tâm vào cấu trúc nội bộ hoặc hoạt động của nó.

3.1.2. Phương pháp thử nghiệm

Dựa vào chức năng Kiểm thử hộp đen (Black box test) có thể được áp dụng hầu như đến mọi cấp độ của kiểm thử phần mềm:

- Kiểm thử đơn vị (Unit test)
- Kiểm thử tích hợp (Intergration test)
- Kiểm thử hệ thống (System test)
- Kiểm thử chấp nhận (Acceptance test).

Tuy nhiên, Black box test được sử dụng thích hợp nhất trong kiểm thử hệ thống (System test) và Kiểm thử chấp nhận (Acceptance test).

3.1.3. Đặc điểm

- Là chiến lược kiểm thử thành phần phần mềm dựa vào thông tin duy nhất là các đặc tả về yêu cầu chức năng của thành phần phần mềm tương ứng.
- Người kiểm thử không cần thiết phải có kiến thức về việc mã hoá, cấu trúc bên trong của thành phần phần mềm, cũng như không yêu cầu phải biết lập trình phần mềm.
- Việc kiểm thử được tiến hành dựa vào việc kiểm thử thành phần phần mềm làm được gì, có phù hợp với yêu cầu của người dùng hay không. Các tester nhập số liệu vào phần mềm và chỉ cần xem kết quả của phần mềm và các mục tiêu kiểm tra.
- Mức test này thường yêu cầu các tester phải viết test case đầy đủ trước khi test; khi test, đơn giản chỉ cần thực hiện theo các bước mô tả trong test case thao tác và nhập data vào, sau đó xem kết quả trả về hoặc hành vi của phần mềm, rồi so sánh với kết quả mong đợi được viết trong testcase.

3.1.4. Tạo testcase và thực hiện testcase

- Tạo và thực hiện testcase trên giao diện bên ngoài của chương trình, không can thiệp vào code.

3.2. White-box testing

3.2.1. Định nghĩa

Kiểm thử hộp trắng (While box test) là phương pháp thử nghiệm phần mềm, trong đó các thiết kế, cấu trúc giải thuật bên trong, và việc thực hiện các công việc đều được biết đến.

3.2.2. Phương pháp thử nghiệm

Dựa vào thuật giải kiểm thử hộp trắng dựa vào thuật giải cụ thể, vào cấu trúc dữ liệu bên trong của đơn vị phần mềm cần kiểm thử để xác định đơn vị phần mềm đó có thực hiện đúng không.

- Với những thanh phần phần mềm quá lớn sẽ tốn rất nhiều thời gian và công sức để kiểm thử nếu như dùng kiểm thử tích hợp (Integration test) hay kiểm thử chức năng (Functional test)).
- Kỹ thuật white box test thích hợp dùng để kiểm thử đơn vị (Unit test)

3.2.3. Đặc điểm

- Là chiến lược kiểm thử TPPM dựa vào giải thuật, cấu trúc bên trong chức năng của TPPM tương ứng.
- Người kiểm thử phải có kiến thức nhất định về việc mã hoá, cấu trúc bên trong của chức năng, biết lập trình phần mềm.
- Việc kiểm thử được tiến hành dựa vào việc kiểm xem giải thuật, mã lệnh đã làm có đúng không.
- Mức test này thường yêu cầu các tester phải viết test case đầy đủ các nhánh trong code; khi test, sẽ set điều kiện và data để chạy vào đủ tất cả các nhánh trong giải thuật, đảm bảo thực hiện đầy đủ.

3.2.4. Tạo testcase và thực hiện testcase

- Khi viết test case: Dựa vào yêu cầu và nội dung Source Code (can thiệp vào bên trong Code của chương trình)
- Khi thực hiện test: Thực thi test trong code (không cần thực thi chương trình, vì thực hiện test white box sẽ sử dụng framework nào đó hỗ trợ (Ví dụ như test kiểu debug)