## **Các định nghĩa cơ bản về testcase**

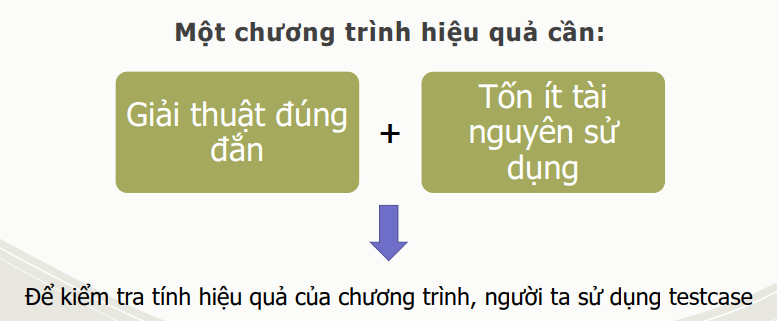
### **1. Testcase là gì**

Test case là “một tập hợp các thông số đầu vào kiểm thử, điều kiện thực thi, và kết quả mong đợi được phát triển cho một mục tiêu cụ thể, như thực hiện một chương trình cụ thể hay kiểm tra sự tuân thủ với một yêu cầu cụ thể.”

Test case hiểu đơn giản là tài liệu dùng để mô tả: Dữ liệu đầu vào (Input) – Hành động (Active) – Kết quả mong đợi (Expected response) để xác định một chức năng của ứng dụng phần mềm hoạt động đúng hay không.

### **2. Tại sao phải tạo các testcase**

* Xác định xem chương trình có hoạt động như mong đợi hay không (kiểm tra độ chính xác của chương trình)
* Xác định chương trình đáp ứng tất cả các tiêu chuẩn, hướng dẫn và yêu cầu được đưa ra.
* Cải thiện chất lượng chương trình, phần mềm



* Tạo ra một sản phẩm hoàn thiện
* Giảm chi phí bảo trì và hỗ trợ phần mềm

### **3. Thế nào là một testcase tốt**

* Chính xác: Test case bao phủ được hết các yêu cầu và giúp phát hiện nhiều lỗi
* Dễ đọc, dễ hiểu: Testcase cần đơn giản và rõ ràng, bất kỳ thử nghiệm nên có thể hiểu được nó bằng cách đọc một lần.
* Có khả năng tái sử dụng: Testcase có khả năng tái sử dụng để giảm đi sự trùng lặp và tiết kiệm thời gian bảo trì khi có sự thay đổi hệ thống, cũng như có thể áp dụng trong tương lai
* Độc lập: có thể thực hiện mà không phụ thuộc vào các test case khác, dễ dàng chia cho nhiều người cùng kiểm thử

### **4. Cấu trúc cơ bản của một test case**

* ID test case: Giá trị cần để xác định số lượng trường hợp cần để kiểm thử. ID có thể bao gồm chữ và số được đánh dấu theo thứ tự tăng dần.
* Test Data: Dữ liệu cần chuẩn bị để thực hiện việc kiểm thử. Có thể ở dạng tên data hoặc đường dẫn tới file.
* Test Steps: Mô tả chi tiết những bước thực hiện test. Tuy nhiên, nên mô tả một cách ngắn gọn và thật rõ ràng.
* Expected Results: Hiển thị kết quả mong đợi từ những bước kiểm thử. Kết quả mong muốn thường dựa trên các yêu cầu hoặc đánh giá theo tài liệu chuyên môn.
* Actual result: Kết quả cuối cùng sau khi test, có hai giá trị tham chiếu: pass, fail.
* Thời gian chạy chương trình, bộ nhớ sử dụng

### **5. Các trường hợp của test case**

Để testcase có độ bao phủ cao, cần xác định trường hợp kiểm thử để không bỏ sót những trường hợp quan trọng. Với 1 giá trị cần kiểm tra luôn luôn có 3 trường hợp lớn cần kiểm tra có thể xảy ra:

* Normal case: Các trường hợp kiểm thử thông thường. Kiểm tra chương trình có đáp ứng đúng các yêu cầu không.
* Abnormal case: Các trường hợp kiểm thử bất bình thường. Kiểm tra các trường hợp không có trong danh sách yêu cầu
* Boundary case: Các trường hợp kiểm tra boundary (Phân tích giá trị biên)

### **6. Các phương pháp thực hiện test**

|  | **Black box test** | **White box test** |
| --- | --- | --- |
| 1. Định nghĩa | - Kiểm tra hộp đen là phương pháp kiểm thử được sử dụng để kiểm tra chương trình, phần mềm mà không quan tâm đến cấu trúc bên trong của chương trình. | - Kiểm tra hộp trắng là phương pháp kiểm thử được sử dụng để kiểm tra chương trình, phần mềm mà yêu cầu phải biết mã nguồn, thuật toán của chương trình. |
| 2. Biết lập trình | - Không yêu cầu hiểu biết về Lập trình | - Yêu cầu hiểu biết nhất định về lập trình. |
| 3. Biết việc thực hiện chương trình | - Không yêu cầu hiểu về cấu trúc bên trong chức năng, và không cẩn hiểu làm thế nào để có được chức năng đó | - Yêu cầu hiểu cấu trúc bên trong chức năng được thực hiện như nào. |
| 4. Cơ sở tạo Test Cases | - Kiểm tra hộp đen được bắt đầu dựa trên tài liệu yêu cầu kỹ thuật | - Kiểm tra hộp trắng được bắt đầu dựa trên các tài liệu thiết kế chi tiết |

### **7. Các lưu ý khi tạo bộ test**

* Các test case phải rõ ràng, ngắn gọn, dễ hiểu
* Các test case không trùng lặp
* Mọi test case đều phải đi kèm với dữ liệu test.
* Testcase nên đảm bảo đủ các trường hợp kiểm thử: Trường hợp thông thường, các trường hợp đặc biệt, các điều kiện biên
* Thực hiện các kĩ thuật kiểm thử
* Mỗi test case thực hiện độc lập và thứ tự thực hiện của các test case không ảnh hưởng đến kết quả thực thi
* Ước lượng và nới rộng kích cỡ bộ dữ liệu/ để biết được ngưỡng dữ liệu của chương trình là bao nhiêu, mỗi cách giải sẽ cho mức ngưỡng để chương trình thực thi
* Nếu tạo test tự động thì khi viết chương trình sinh dữ liệu tự động phải /đảm bảo input theo mô tả, output tương ứng và công thức đảm bảo output đúng

### **8. Ví dụ**

**8.1. Chia hai phân số**

**Yêu cầu:** Thực hiện phép tính chia 2 phân số

**Input:** Nhập vào phân số thứ nhất, phân số thứ hai

**Output:** In ra kết quả thực hiện phép tính đó

**Input Format:** Nhập vào số nguyên a,b,c,d với a,b là tử, mẫu của phân số thứ nhất và c,d là tử và mẫu của phân số thứ hai. a, b,c,d nằm trên cùng một dòng (−105≤ 𝑎, 𝑏, 𝑐, 𝑑 ≤ 105)

**Output Format:**

Kết quả theo định dạng a/b

Nếu kết quả không tồn tại thì xuất 'Error'

**Các trường hợp có thể xảy ra:**

* Một trong 3 đầu vào b, c, d = 0: Xuất ra màn hình Error
* b, c, d khác 0: Rút gọn phân số:

+ Phân số ở dạng tối giản x/y

+ Nếu x.y < 0 thì phải đặt dấu ” - “ ở trên tử số

+ Kết quả ở dạng số nguyên thì chỉ xuất ra số nguyên

+ Kết quả ở dạng số vô tỉ thì làm tròn đến chữ số thập phân thứ hai

| **Input** | **output** |
| --- | --- |
| 2 1 0 5 | Error |
| 1 -2 4 8 | -1 |
| 0 1 3 5 | 0 |
| 1 0 3 8 | Error |
| 3 4 5 -6 | -9/10 |
| -2 1 1 -2 | 4 |
| 2 3 5 0 | Error |
| 100000 -90000 90000 100000 | -100/81 |

**8.2. Phương trình bậc 2**

**Yêu cầu:** Hãy viết chương trình tìm các nghiệm của một phương trình: ax2+bx+c=0 rồi in ra màn hình các nghiệm đó.

**Input:** Nhập vào trên cùng một dòng 3 số nguyên a, b, c tương ứng với các hệ số của phương trình.

**Output:** In ra kết quả nghiệm của phương trình

**Các trường hợp có thể xảy ra:**

* a=0:
* b=0, c0: Phương trình vô nghiệm. Xuất ra màn hình “VN”
* b=0, c=0: Phương trình có vô số nghiệm. Xuất ra màn hình “VSN”
* b0: Phương trình có 1 nghiệm. Xuất ra màn hình
* a0:
* : Phương trình vô nghiệm. Xuất ra màn hình “VN”
* : Phương trình có 1 nghiệm kép. Xuất ra màn hình
* : Phương trình có 2 nghiệm phân biệt. Xuất ra màn hình

**Bộ test mẫu:**

| **Input** | **output** |
| --- | --- |
| 0 0 0 | VSN |
| 0 0 2 | VN |
| 0 2 0 | 0 |
| 0 2 1 | -0.5 |
| 1 0 1 | VN |
| 1 0 0 | 0 |
| 2 -4 2 | 1 |
| 1 -3 2 | 2 1 |
| 1 -7 5 | 0.65 -7.65 |
| 100000 -452110 431220 | 3.15 1.36 |

**8.3. Kiểm tra ngày tháng năm hợp lệ**

**Yêu cầu:** Kiểm tra ngày tháng năm nhập vào có phải là một ngày tháng năm hợp lệ

**Input:** Nhập vào trên cùng một dòng 3 số nguyên dd,mm,yy tương ứng với các giá trị ngày, tháng, năm cần kiểm tra.

**Output:** In ra màn hình ‘True’ nếu ngày tháng năm hợp lệ, ‘False’ nếu ngày tháng năm không hợp lệ

**Các trường hợp có thể xảy ra:**

* yy < 1: Không hợp lệ. In ra màn hình ‘False’
* yy >= 1:
* mm < 1 hoặc mm > 12: Không hợp lệ. In ra màn hình ‘False’
* mm là một trong những giá trị: 1,3,5,7,8,10,12
* dd < 1 hoặc dd > 31: Không hợp lệ. In ra màn hình ‘False’
* 1<= dd <= 31: Hợp lệ. In ra màn hình ‘True’
* mm là một trong những giá trị: 4,6,9,11
* dd < 1 hoặc dd > 30: Không hợp lệ. In ra màn hình ‘False’
* 1<= dd <= 30: Hợp lệ. In ra màn hình ‘True’
* mm = 2: Kiểm tra năm nhuận
* Nếu năm nhuận: 1<= dd <=29: Hợp lệ. In ra màn hình ‘True’
* Nếu năm không nhuận: 1<= dd <=28: Hợp lệ. In ra màn hình ‘True’
* Các trường hợp dd còn lại không hợp lệ. In ra màn hình ‘False’

**Bộ test mẫu:**

| **Input** | **output** |
| --- | --- |
| 25 2 0 | False |
| 7 0 2016 | False |
| 6 13 2022 | False |
| 31 8 2018 | True |
| -1 6 20 | False |
| 31 9 2018 | False |
| 28 3 22 | True |
| 29 2 2000 | True |
| 29 2 2004 | True |
| 29 2 2200 | False |

## **Kiểm thử tự động chương trình**

### **1. Kiểm thử tự động chương trình**

Khái niệm

Kiểm thử tự động là việc sử dụng các công cụ để thực hiện các test case. Kiểm thử tự động cũng có thể nhập dữ liệu thử nghiệm vào hệ thống kiểm thử, so sánh kết quả mong đợi với kết quả thực tế và tạo ra các báo cáo kiểm thử chi tiết.

Tại sao phải thực hiện kiểm thử tự động

* Giúp tiết kiệm tiền bạc và thời gian
* Chính xác hơn: Nhờ độ ổn định cao, kiểm thử tự động có thể thực thi các test case với độ chính xác cao hơn.
* Độ bao phủ cao: Sử dụng kiểm thử tự động, chúng ta có thể thực thi số lượng lớn test case trong một thời gian ngắn. Nên độ bao phủ của nó rất cao.
* Tăng năng suất của công việc
* Không cần sự giám sát của con người

### **2. Unit test**

Unit test là một cấp độ kiểm thử phần mềm trong đó các đơn vị (unit) / thành phần riêng lẻ của phần mềm được kiểm thử. Mục tiêu của Unit test là cô lập một phần code và xác minh tính chính xác của đơn vị đó.

Kiểm thử đơn vị tự động bằng Pyunit của Python

Python ngay từ đầu không phải là một công cụ kiểm thử tự động mà là một [ngôn ngữ lập trình](http://vntesters.com/gioi-thieu-nnlt-python/). Vậy làm thế nào để có thể tạo ra được các kịch bản kiểm thử tự động trên ngôn ngữ Python. Câu trả lời cho vấn đề này là thư viện *unittest* của Python.

## **Quiz; bài tập về nhà**

## 