BÀI 6 KIỂM THỬ VÀ SỬA LỖI CHƯƠNG TRÌNH

Học xong bài này, em sẽ:

- Biết được việc kiểm thử giúp lập trình viên phát hiện lỗi, làm tăng độ tin cậy của chương trình nhưng chưa chứng minh được chương trình đã hết lỗi.
- Biết được một số kinh nghiệm gỡ lỗi và các thói quen lập trình tốt để dễ gỡ lỗi.

Theo em, tại sao rất ít khi viết chương trình xong có thể chạy được ngay?

1 Nguyên nhân gây lỗi và truy vết lỗi

a) Các loại lỗi và nguyên nhân

Lỗi cú pháp là lỗi hay xảy ra trong quá trình soạn thảo chương trình. Người lập trình chỉ cần hiểu rõ ngôn ngữ lập trình mình sử dụng là có thể dễ dàng sửa lỗi cú pháp. Hiện nay các môi trường tích hợp phát triển phần mềm (IDE – Integrated Development Environment) có công cụ soạn thảo chương trình nhằm hạn chế những sai sót có thể sinh ra lỗi cú pháp.

Chương trình đã biên dịch, chạy thử thành công một vài lần vẫn có thể đột ngột dừng giữa chừng hoặc chạy mãi không dừng. Đó là lỗi thời gian chạy (*runtime errors*). Nguyên nhân thường do có giá trị không hợp lệ khi thực hiện một tính toán nào đó. Ví dụ: quên gán trị khởi tạo một biến, bỏ qua việc kiểm tra mẫu số khác 0 trước khi chia; chỉ số phần tử của danh sách ngoài phạm vi cho phép; quên tăng giá trị biến đếm để kiểm tra vòng lặp,...

b) Truy vết lỗi và thông báo lỗi

Vùng soạn thảo các câu lệnh trong môi trường lập trình IDE thường có hiển thị số thứ tự các dòng lệnh, đánh số tăng dần từ 1. Khi phát sinh một lỗi, chức năng gỡ lỗi sẽ truy ngược lùi về phía trên, tìm đến tận gốc, tới dòng lệnh có câu lệnh gây lỗi. Thông báo lỗi in ra danh sách các dòng lệnh truy vết được, ghi kèm số thứ tự dòng lệnh trong văn bản chương trình. Người lập trình dễ dàng tìm ra chuỗi dòng lệnh gây lỗi.

2 Chạy thử chương trình

Chạy thử là để phát hiện lỗi trong mã nguồn của chương trình. Gỡ lỗi là xác định vị trí có lỗi, nguyên nhân gây lỗi và sửa lỗi. Phát hiện lỗi và sửa lỗi là hai việc đan xen trong một quá trình. Mục đích cuối cùng là đảm bảo rằng chương trình hoạt động đúng, đáp ứng yêu cầu bài toán đặt ra.

Đọc bản mới nhất trên hoc10.vn Bản sách mẫu

Thuật toán sai thì chương trình thực hiện đúng thuật toán đó sẽ cho kết quả sai. Việc phát hiện chương trình còn lỗi và sửa lỗi sẽ không phân biệt đó là lỗi chương trình thực hiện thuật toán hay lỗi của bản thân thuật toán.

Tập hợp toàn bộ các trường hợp đầu vào có thể xảy ra của một chương trình thường là vô hạn. Đầu vào cho chương trình thực hiện một thuật toán sắp xếp là bất cứ dãy số nào. Không thể chạy thử chương trình với tất cả các đầu vào có thể có.

Chạy thử cho phép người lập trình dễ phát hiện lỗi hơn, qua đó kịp thời đưa ra các biện pháp xử lí lỗi. Mặc dù điều này không đảm bảo tuyệt đối rằng chương trình không còn lỗi nhưng nó cũng hạn chế được rất nhiều rủi ro phát sinh lỗi trong quá trình vận hành.

3 Một số kinh nghiệm thực hành gỡ lỗi chương trình

Các ca kiểm thử để phát hiện lỗi chương trình

Một ca kiểm thử là một trường hợp đã cho các đầu vào cụ thể và dự đoán trước kết quả đầu ra đúng yêu cầu của bài toán.

Các ca kiểm thử nhằm phát hiện các lỗi tiềm ẩn. Dưới đây là một số gợi ý các ca kiểm thử:

– Kiểm tra các câu lệnh rẽ nhánh với đầu vào tương ứng cho đủ các trường hợp.

– Kiểm tra các câu lệnh lặp với đầu vào khiến số lần lặp là 0 lần, 1 lần, nhiều lần.

– Kiểm tra với các giá trị ở các đầu mút trái, phải của một biểu thức điều kiện. Ví dụ, với điều kiện $a \le x \le b$, hãy thử với các giá trị a và b. Các hạn chế "không là số dương", "không là số âm",... hãy thử với đầu vào bằng 0.

– Cẩn thận trọng với điều kiện "bằng nhau" khi so sánh hai biến kiểu số thực vì kết quả tính toán có thể bị làm tròn. Ví dụ, sau khi tính tỉ lệ phần trăm, cộng các tỉ lệ phần trăm không chắc sẽ đúng bằng 100.

– Kiểm tra với các đầu vào "không mong đợi" nếu muốn biết chương trình sẽ hoạt động như thế nào khi người khác chạy "khám phá". Các giá trị không mong đợi có thể là giá trị rất lớn hoặc rất gần số không, giá trị không hợp lệ,...

Hãy cho một số ví dụ ca kiểm thử:

1) Chương trình giải phương trình bậc hai.

2) Chương trình tính đếm (tính tổng, tính trung bình cộng,...) các số dương trong một mảng số thực.

Đọc bản mới nhất trên hoc10.vn Bản sách mẫu

Chia để trị

Kiểm thử và sửa lỗi một đoạn mã lệnh ngắn dễ hơn nhiều so với cả một văn bản chương trình dài. Hãy kiểm thử và sửa lỗi từng đoạn mã lệnh, từng hàm riêng biệt, chắc chắn rằng nó làm đúng việc cần làm trước khi chuyển sang phần khác. Văn bản chương trình có thể được gỡ lỗi, chỉnh sửa hoàn thiện dần từng phần nếu ta biết cách tổ chức tách biệt các phần công việc của chương trình để dễ sửa lỗi. Việc tổ chức tách biệt các phần công việc của chương trình cũng là một khía cạnh của *phương pháp lập trình theo mô đun* (sẽ được trình bày chi tiết hơn ở các bài học sau).

Hãy in ra

Có những lỗi logic rất khó phát hiện. Một kinh nghiệm là kiểm soát các giá trị biến, biểu thức trong quá trình chạy kiểm thử chương trình. Điều này có thể thực hiện bằng cách in ra các giá trị biến, biểu thức; hoặc theo dõi các giá trị biến, biểu thức bằng trình gỡ rối nếu nó được trang bị sẵn trong IDE.

4 Tập thói quen tốt khi lập trình để dễ gỡ lỗi

Các kĩ năng lập trình và gỡ lỗi chỉ có được qua kinh nghiệm thực hành. Hãy học từ những sai lầm của bản thân, ghi nhớ những lỗi đã mắc, cải tiến phong cách lập trình. Nên tập một số thói quen tốt sau đây để chương trình ít lỗi và việc gỡ lỗi dễ dàng hơn:

– Không lập tức bắt đầu viết các câu lệnh ngay sau khi đọc xong bài toán lập trình và nảy ra ý tưởng (thuật toán) giải bài toán. Nên bắt đầu với việc tách biệt các phần công việc cần làm và thiết kế tổng thể chương trình.

– Mô tả thuật toán bằng liệt kê các bước, chọn những phần việc chuyển thành chương trình con (hàm tự định nghĩa), xác định rõ đầu vào đầu ra của mỗi hàm.

– Chọn đặt *tên gợi nhớ* cho các hàm và các biến quan trọng, sao cho dễ nhận biết nó làm gì, đó là cái gì. Điều này giúp tránh nhầm lẫn khi viết câu lệnh và dễ phát hiện lỗi hơn.

– Viết chú thích đầy đủ, ngay trước hay ngay sau các khai báo tên hàm, tên biến quan trọng, các đoạn mã lệnh cần chú ý. Sau này, chính người lập trình có thể không còn nhớ mình có ý tưởng gì lúc viết các dòng lệnh đó.

5 Tổ chức tách biệt các phần của một chương trình

Định nghĩa hàm để thực hiện thuật toán

Người lập trình tự định nghĩa một (hay một số) hàm: chọn tên hàm, tên các biến đầu vào và cách trả về kết quả. Trong mô tả thuật toán đã có sẵn những thông tin này. Phần thân hàm là kết quả chuyển từ mô tả thuật toán thành câu lệnh của ngôn ngữ lập trình đã chọn.

Đọc bản mới nhất trên hoc10.vn Bản sách mẫu

Các câu lệnh để chạy thử phát hiện lỗi

Trong chương trình cần có thêm các câu lệnh làm những việc sau:

– Gán dữ liệu đầu vào: Một số câu lệnh gán giá trị cho các biến đầu vào. Dữ liệu đầu vào cũng có thể đọc từ tệp cho trước.

– Xuất kết quả đầu ra: Một số câu lệnh in ra màn hình. Để tiện kiểm tra, đỡ nhầm lẫn, nên in kèm lời mô tả đầu ra là gì; có thể in kèm cả dữ liệu đầu vào tương ứng.

Lợi ích của việc tổ chức tách biệt các phần công việc

– Dễ chạy thử: Các câu lệnh để chạy thử kiểm tra ở các chỗ cần theo dõi giá trị của các biến, việc thực hiện các đoạn chương trình. Dùng dấu chú thích "#" có thể liệt kê một danh sách các ca kiểm thử khác nhau và chạy thử từng ca.

– Dễ sửa lỗi: Bố cục chương trình có logic rõ ràng, dễ thấy lỗi xảy ra ở việc nào.

Câu 1. Có các loại lỗi chương trình nào? Nguyên nhân gây ra loại lỗi đó có thể là gì?

Câu 2. Hãy nêu một vài thói quen lập trình tốt để chương trình ít lỗi và dễ gỡ lỗi.

Em hãy liệt kê một số ca kiểm thử cho chương trình:

a) Tìm số x trong một dãy số (đã cho cụ thể).

b) Sắp xếp một dãy số.

Câu 1. Tại sao nói kiểm thử chương trình làm tăng độ tin cậy của chương trình nhưng chưa chứng minh được chương trình đã hết lỗi?

Câu 2. Nên làm gì mỗi khi nghi ngờ một chức năng nào đó của chương trình chưa chắc đúng như ta mong muốn?

Tóm tắt bài học

- Chương trình đã chạy ra kết quả, có thể vẫn còn lỗi tiềm ẩn; kiểm thử để phát hiện lỗi và sửa lỗi nhằm đảm bảo rằng chương trình đáp ứng yêu cầu bài toán đặt ra.
- Cần kiểm thử: đủ các trường hợp của cấu trúc rẽ nhánh, các trường hợp ở đầu mút của một biểu thức điều kiện và các trường hợp của cấu trúc lặp có số lần lặp là 0 lần, 1 lần, nhiều lần.
- Cần tập các thói quen lập trình tốt để chương trình ít lỗi và dễ gỡ lỗi.

Đọc bản mới nhất trên hoc10.vn Bản sách mẫu