Ghi chú: khiếm khuyết = defect (trong bài hay dùng defect, thôi thống nhất cứ ghi là lỗi hết đi nhan, cho dễ)

Các bước tìm kiếm lỗi:

Gỡ rối bao gồm việc tìm kiếm ra lỗi và sửa chữa nó. Việc tìm kiếm và hiểu biết về nó chiếm khoảng 90% phần việc của gỡ rối.

* Ổn định các lỗi. Chắc chắn vị trí đoạn code gây ra lỗi, dự đoán được lỗi sẽ xảy ra trong mọi trường hợp, lỗi xảy ra không theo dự đoán thường là lỗi khởi tạo, lỗi con trỏ… (sử dụng con trỏ chưa được khởi tạo, hay con trỏ đang trỏ tới giá trị rác…). Việc này không hề đơn giản, nó bao gồm thu hẹp các trường hợp thử nghiệm mà vẫn phát sinh ra lỗi. (Phần này có thể gọi là tái tạo lại vấn đề gây nên lỗi)
* Tìm ra vị trí nguồn của các lỗi. Thử thay đổi các tham số mà bạn nghi ngờ gây nên lỗi, một ở dưới biên, một ở biên, một ở trên biên để xác định giả thuyết của mình là đúng. Hay với đầu vào một chức năng quá phức tạp, cố gắng đơn giản đầu vào tới mức tối thiểu để dễ dàng nhận dạng lỗi.
* Đôi khi lỗi xảy ra khi bạn thay đổi hay thêm bớt code gần đây. Nếu bạn gặp một lỗi mới mà không đoán được, hãy thử chạy lại phiên bản trước đó để xem có lỗi không, nếu không, bạn biết chắc rằng lỗi xảy ra ở đoạn code bạn vừa chỉnh sửa.
* Mở rộng phạm vi tìm kiếm. Nếu bạn không tìm được lỗi trong phạm vi mà bạn vạch ra, đừng ngần ngại mở rộng tìm kiếm ra xung quanh.
* Đừng ngại bàn bạc vấn đề với người khác
* Sử dụng công cụ hỗ trợ đề tìm lỗi. Các trình biên dịch hiện nay hỗ trợ khá tốt việc này, nó có khả năng chỉ ra chính xác vị trí của lỗi (với những lỗi đơn giản), còn với lỗi runtime, bạn phải dùng kỹ thuật khác)

***Gỡ lỗi:***

Phần khó của gỡ lỗi là việc tìm kiếm các khiếm khuyết (defect). Sửa chữa các khiếm khuyết (defect) là phần dễ dàng. Nhưng  
như nhiều nhiệm vụ dễ dàng khác, thực tế là nó dễ dàng làm cho nó đặc biệt dễ gây ra lỗi. Ít nhất  
một nghiên cứu thấy rằng sửa lỗi có nhiều hơn 50 phần trăm cơ hội bị  
sai ngay lần đầu tiên (Yourdon 1986b). Dưới đây là một vài hướng dẫn để giảm  
cơ hội báo lỗi:

* **Hiểu được vấn đề trước khi bạn sửa chữa nó**. Chất lượng chương trình sẽ bị giảm sút nếu bạn sửa vấn đề mà không thực sự hiểu biết chúng. Trước khi bạn sửa chữa mộtvấn đề, hãy chắc chắn bạn hiểu nó tới phần lõi.
* **Hiểu được các chương trình**. KHông chỉ có vấn đề, nếu bạn hiểu được ngữ cảnh mà vấn đề xảy ra, bạn có nhiều khả năng giải quyết vấn đề hoàn toàn thay vì chỉ một khía cạnh của nó. Một nghiên cứu được thực hiện với các chương trình ngắn cho thấy lập trình viên đạt được một sự hiểu biết toàn diện về hành vi của chương trình có một cơ hội tốt hơn để sửa nó thành công hơn so với lập trình viên tập trung duy nhất vào 1 vùng nhỏ (et al Littman năm 1986.). Bởi vì chương trình trong nghiên cứu này  
  là nhỏ (280 dòng), nó không chứng minh được rằng bạn nên cố gắng hiểu hoàn toàn một chương trình có 50.000 dòng trước khi sửa chữa một khiếm khuyết. Nó đề nghị bạn nên hiểu ít nhất là các mã trong vùng lân cận của khiếm khuyết đó- "vùng lân cận" được hiểu là vài trăm chứ chứ không phải 1 vài dòng.
* **Xác nhận các chuẩn đoán về vấn đề**. Trước khi bạn vội vã để sửa chữa một khiếm khuyết, hãy chắc chắn rằng bạn đã chẩn đoán các vấn đề chính xác.
* **Lưu mã nguồn gốc** .Trước khi bạn bắt đầu sửa chữa các khiếm khuyết, hãy chắc chắn rằng mình đã lưu trữ một phiên bản mã mà bạn có thể quay trở lại sau đó. Rất dễ để quên 1 thay đổi trong một nhóm các thay đổi. Nếu bạn có mã nguồn gốc, ít nhất bạn có thể so sánh tập tin cũ và các tập tin mới và nhìn thấy vị trí các thay đổi.
* **Sửa chữa các vấn đề, không phải là sửa chữa triệu chứng** Bạn cũng nên sửa chữa các triệu chứng, nhưng nên tập trung vào việc sửa các vấn đề cơ bản. Nếu bạn không hoàn toàn hiểu được vấn đề, bạn không nên sửa mã nguồn. Nếu bạn sửa thì bạn đang sửa chữa các triệu chứng và làm cho code tồi tệ hơn.
* **Chỉ thực hiện một thay đổi tại một thời điểm.** Chỉ 1 thay đổi cũng đủ phức tạp rồi. Nếu bạn thực hiện nhiêu hơn 1 thay đổi trong 1 thời điểm thì có thể bạn sẽ vô tình tạo nên những lỗi mới.
* **Kiểm tra sửa chữa của bạn**. Hãy tự mình kiểm tra, để người khác kiểm tra hoặc cùng kiểm tra với họ. Nếu bạn mới chỉ sửa chữa đc 1 phần của vấn đề, bạn sẽ nhận ra để tiếp tục công việc của mình.
* **Thêm các unit test vào chương trình của mình.**
* **Truy tìm các lỗi tương tự.** Các lỗi có xu hướng xảy ra trong nhóm, vì vậy khi bạn tìm đc 1 lỗi, hãy tìm thêm các lỗi tương tự. Để làm đc diều này đòi hỏi bạn phải hiểu chương trình 1 cách thấu đáo. Nếu bạn không thể tìm các lỗi tương tự thì có lẽ bạn vẫn chưa thực sự hiểu chương trình vận hành ntn.
* **Thư giãn.** Giả sử bạn đang sẵn sàng cho 1 chuyến du lịch, và chương trình của bạn đã sẵn sàng để phát hành. Bạn chỉ còn 1 lỗi nữa để sửa thôi. Bạn vội vàng thay đổi mã nguồn và check in nó lên trình quản lý mã nguồn mà k biên dịch lại. Điều gì có thể xảy ra? Cái lỗi đó đã không đc sửa 1 cách đúng đắn. Và quản lý dự án sẽ “hỏi thăm sức khỏe” của bạn. :D Điều tốt nhất bạn nên làm là: sửa cái lỗi đó, biên dịch , chạy thử và kiểm tra trên máy của bạn. Nhưng đừng check in lên trình quản lý mã nguồn vội, hãy thư giãn (đi chơi 1 trận bóng chẳng hạn), bạn cần thời gian để chắc chắn rằng giải pháp của bạn là chính xác. Điều này có thể mất thêm 1 ít thời gian nhưng bạn sẽ không bị quan rlys dự án gọi giật về khi đang trong chuyến đi.