

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

**BÁO CÁO BÀI THỰC HÀNH**

**Kiểm thử xâm nhập**

**Quét mạng và dịch vụ nâng cao**

**Giảng viên: Đinh Trường Duy**

**Nhóm lớp: 01**

**Sinh viên: Hoàng Trung Kiên**

**Mã sinh viên: B20DCAT098**

**Hà Nội – 2024**

1. Mục đích.

Sử dụng tiện ích GDB để gỡ lỗi một chương trình C.

2. Yêu cầu đối với sinh viên.

Có kiến thức về ngôn ngữ lập trình C.

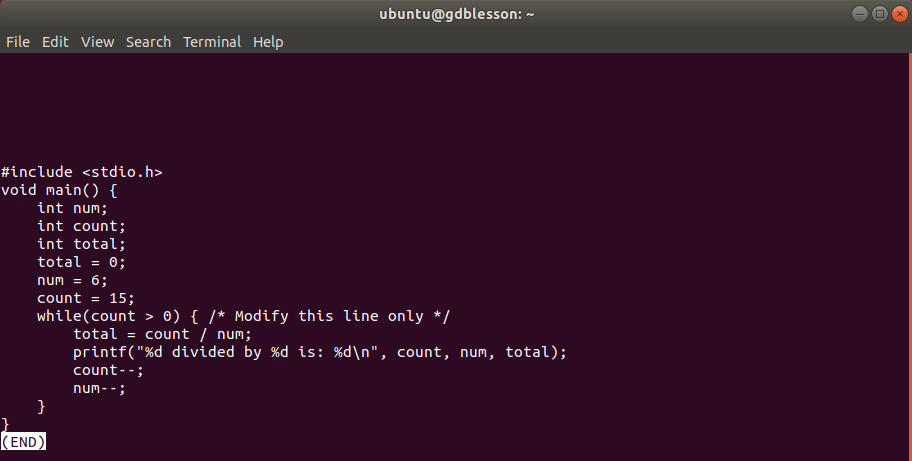
Biết cách biên dịch và thực thi một chương trình C bất kỳ.

3. Nội dung thực hành.

Khởi động lab:

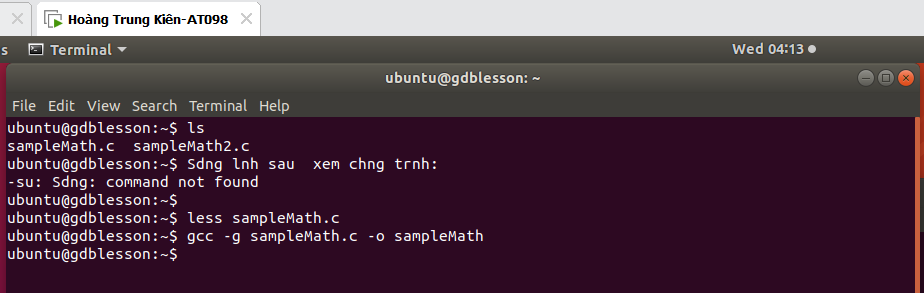
Chạy lệnh: labtainer -r gdblesson trong terminal của Labtainer

Sử dụng lệnh sau để xem chương trình: less sampleMath.c

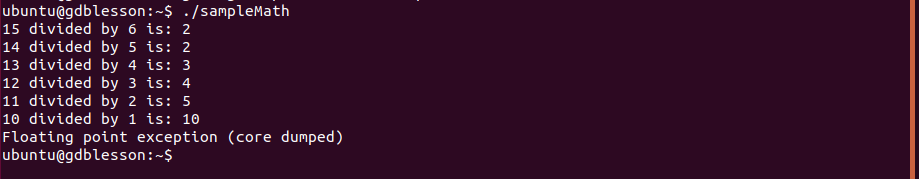


Sau đó, hãy chiêm nghiệm kết quả của chương trình sẽ như thế nào. Khi thực hiện xong, hãy gõ “q” để thoát khỏi màn hình. Tiếp theo, gõ lệnh gcc vào shell như sau:

gcc -g sampleMath.c -o sampleMath

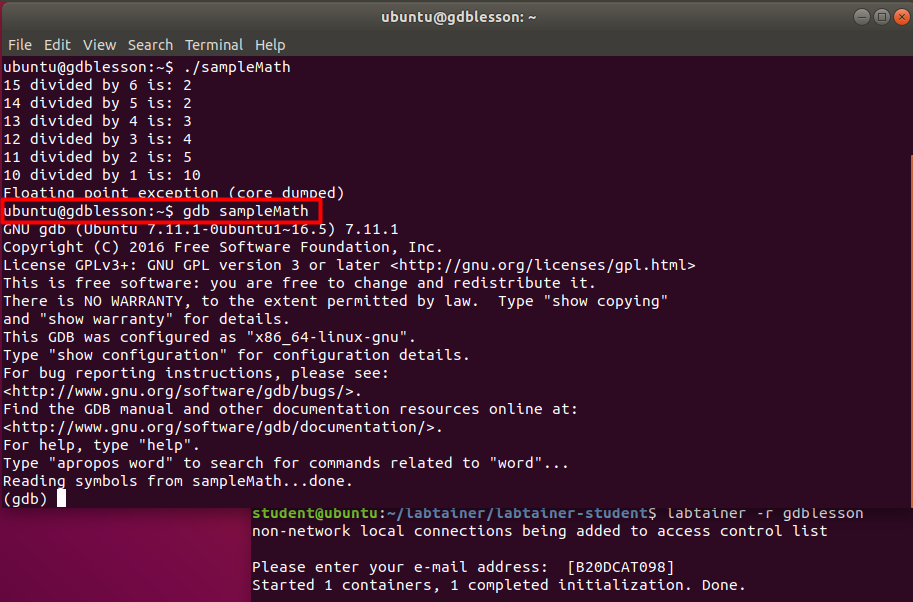


Lệnh trên sẽ biên dịch “*samplemath.c*” và tạo một phiên bản thực thi của chương trình được gọi là “*sampleMath*”. Để chạy chương trình mới này ta gõ: ./sampleMath

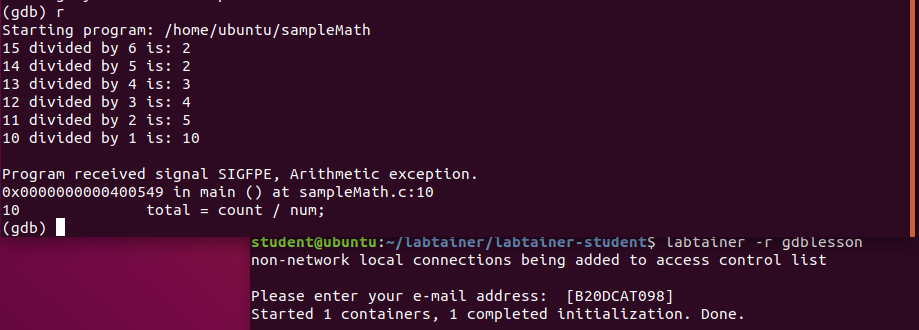


Lưu ý các lỗi xuất hiện khi chạy chương trình.

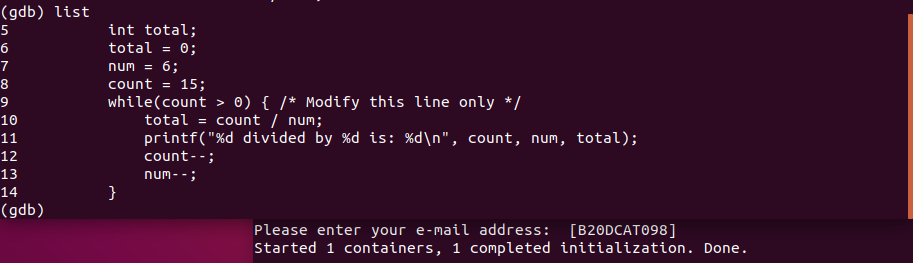
Thay vì chỉ xem xét chương trình C, chương trình GDB sẽ giúp chúng ta phân tích điều gì đang tạo ra một ngoại lệ. Gõ “gdb sampleMath” vào dòng lệnh.



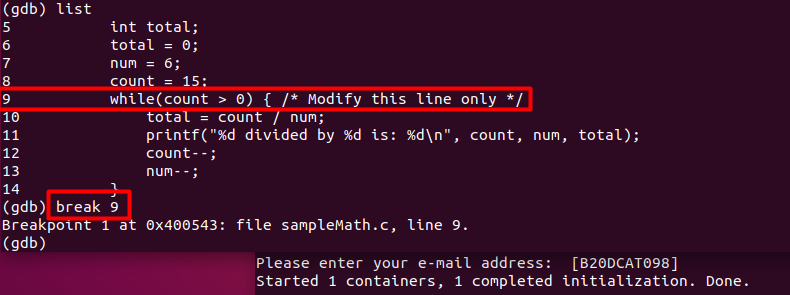
Khi chương trình bắt đầu, gõ “r” hoặc “run” để bắt đầu chạy chương trình.



Để hiểu rõ hơn nguồn gốc của vấn đề, hãy liệt kê nội dung của chương trình bằng lệnh list

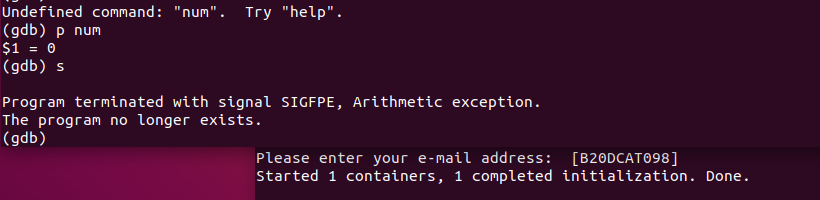


Để lặp lại một lệnh mà không cần phải nhập lại bất kỳ thứ gì, chỉ cần nhấn phím Return (Enter); làm điều này cho lệnh list sẽ hiển thị mười dòng mã tiếp theo. Nhìn vào mã và tìm ra dòng nào mà “while-loop” bắt đầu. Nhập “break #” trong đó “#” là số dòng tương ứng.



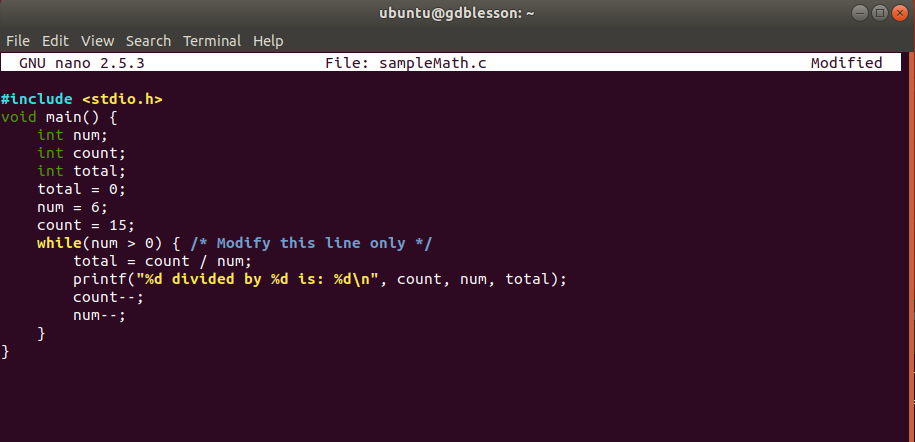
Sử dụng lại lệnh “run”.

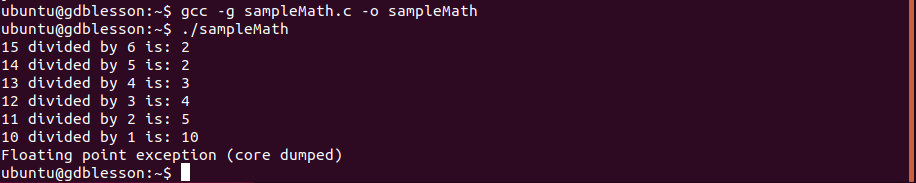
Sử dụng lệnh “p” hoặc “print” để hiển thị giá trị của bất kỳ biến nào, chỉ cần nhập tên của nó sau lệnh “p” hoặc “print”. Nhập “s” hoặc “step” để tiếp tục chạy một dòng trong chương trình.



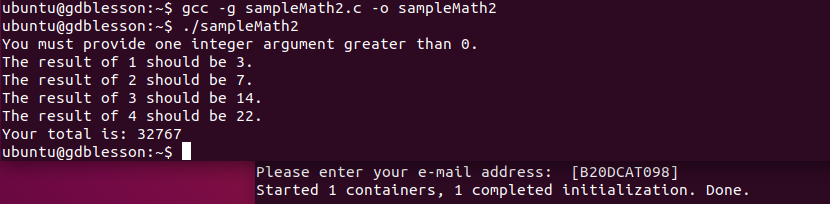
=> Ta nhận thấy lỗi do thực hiện phép chia cho 0

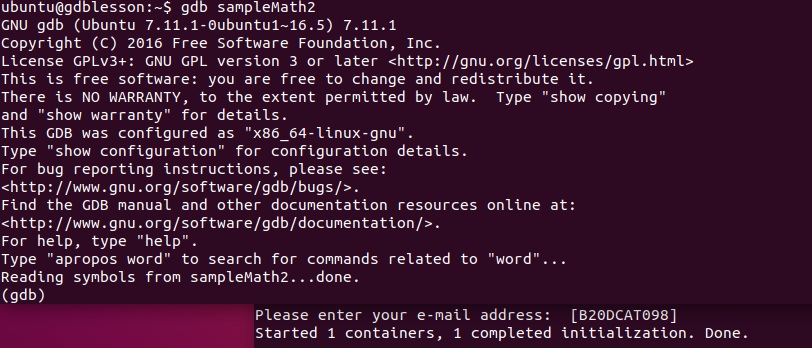
Sửa lại code:



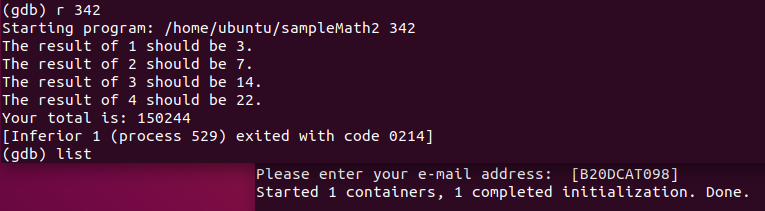
Cuối cùng, thực hiện lại chương trình. 

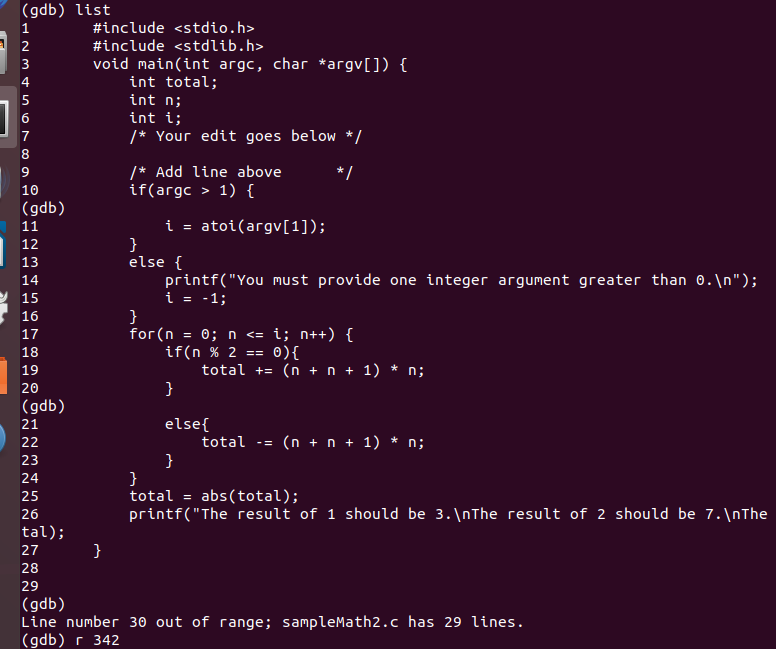
Biên dịch “sampleMath2.c” thành “sampleMath2” bằng gcc. Thực thi chương trình bằng cách gõ “./sampleMath2”





Chạy chương trình một lần bằng cách sử dụng “r #” trong đó ký hiệu “#” là giá trị của tham số đang được đưa vào chương trình để kiểm tra. Đặt một break point ở đâu đó trong mã và chạy tiếp. In các biến và cố gắng tìm ra giá trị nào gây ra kết quả không chính xác. Khi phát hiện ra điều gì cần thay đổi, hãy sửa đổi mã cho phù hợp và nhập 342 vào đó.





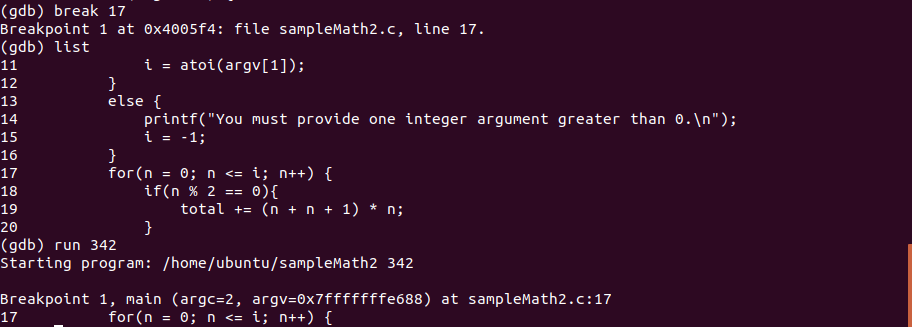
argv (Argument Vector) là một mảng các con trỏ liệt kê tất cả các đối số được truyền vào.

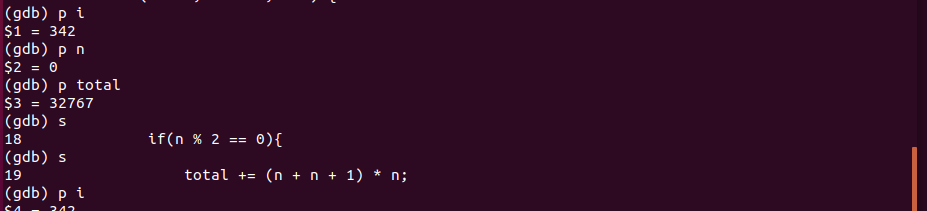
Các đối số có thể được ngăn cách bởi dấu cách. Nếu bản thân đối số có khoảng trắng bên trong, thì có thể dùng dấu ngoặc đơn hoặc ngoặc kép cho đối số đó.

argc (Argument Count) là số nguyên biểu thị số lượng các đối số ở trên.

Đặt break point

Break

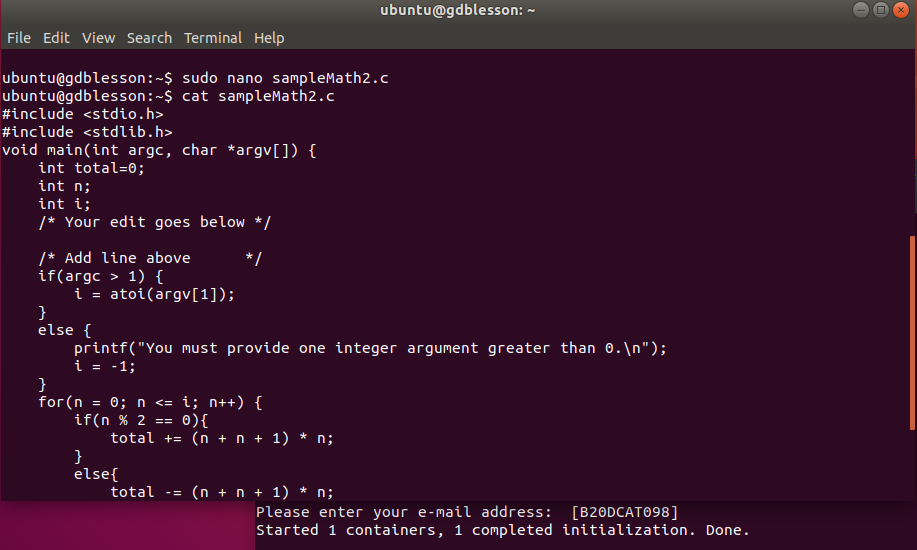




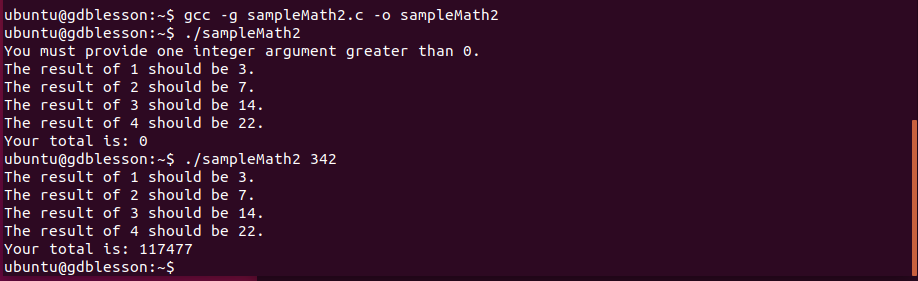
⇒ có thể nhận thấy khi chạy và in ra giá trị các biến thì total = 32767 với n = 0 , với n = 2 total = 32764

Có sự vô lý => giá trị total ban đầu chưa được gán giá trị nào, dẫn đến kết quả sai.

Sửa lại code, gán total = 0



Biên dịch lại và run:



Checkwork

