****

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

**KHOA AN TOÀN THÔNG TIN**

****

**BÁO CÁO THỰC HÀNH 2**

**Họ và tên: Trương Công Sơn**

**Mã sinh viên: B19DCAT156**

**Giảng viên giảng dạy: Đinh Trường Duy**

***Hà Nội – 2023***

Contents

[**BÁO CÁO THỰC HÀNH 2** 1](#_Toc131756285)

[I. Lý Thuyết 3](#_Toc131756286)

[1. Cấu trúc dữ liệu đơn giản: 3](#_Toc131756287)

[2. Kiến trúc máy tính 4](#_Toc131756288)

[3. Bộ nhớ máy tính 5](#_Toc131756289)

[4. Lỗi tràn bộ đệm 5](#_Toc131756290)

[5. Tìm hiểu về gdb: các câu lệnh, cách gỡ lỗi sử dụng gdb 6](#_Toc131756291)

I. Lý Thuyết

1. Cấu trúc dữ liệu đơn giản:

* Mảng (Array): là một cấu trúc dữ liệu lưu trữ một tập hợp các giá trị có cùng kiểu dữ liệu, được xác định bởi một chỉ mục (index) duy nhất. Mỗi phần tử của mảng được đánh số bắt đầu từ 0. Điều này có nghĩa là để truy xuất phần tử thứ i của mảng, ta cần chỉ ra chỉ mục i. Mảng có thể được sử dụng để lưu trữ các danh sách, ma trận, v.v. Các phép toán thường được sử dụng trên mảng bao gồm truy cập phần tử, thay đổi giá trị của phần tử, tìm kiếm và sắp xếp.
* Danh sách liên kết (Linked List): là một cấu trúc dữ liệu mà mỗi phần tử của nó bao gồm một giá trị và một con trỏ trỏ đến phần tử tiếp theo. Danh sách liên kết được sử dụng để lưu trữ các danh sách không có số lượng phần tử cố định và cần thêm hoặc xóa phần tử trong danh sách một cách linh hoạt. Các phép toán thường được sử dụng trên danh sách liên kết bao gồm thêm phần tử vào đầu hoặc cuối danh sách, thêm phần tử vào giữa danh sách, xóa phần tử và tìm kiếm phần tử.
* Ngăn xếp (Stack): là một cấu trúc dữ liệu lưu trữ một danh sách các phần tử, cho phép thêm hoặc xóa phần tử chỉ ở đầu danh sách. Ngăn xếp được sử dụng để lưu trữ các thao tác ngăn xếp (stack operations) như thực hiện theo thứ tự đảo ngược và sử dụng đệ quy trong lập trình. Các phép toán thường được sử dụng trên ngăn xếp bao gồm đẩy phần tử vào ngăn xếp, lấy phần tử ra khỏi ngăn xếp và kiểm tra xem ngăn xếp có trống hay không.
* Hàng đợi (Queue): là một cấu trúc dữ liệu lưu trữ một danh sách các phần tử, cho phép thêm phần tử vào cuối danh sách và xóa phần tử chỉ ở đầu danh sách. Hàng đợi được sử dụng để lưu trữ các thao tác hàng đợi (queue operations) như thực hiện theo thứ tự và xử lý các tác vụ đợi trong các ứng dụng như hệ thống phân phối và xử lý dữ liệu. Các phép toán thường được sử dụng trên hàng đợi bao gồm thêm phần tử vào hàng đợi, lấy phần tử ra khỏi hàng đợi và kiểm tra xem hàng đợi có trống hay không.
* Bảng băm (Hash table): là một cấu trúc dữ liệu lưu trữ một tập hợp các cặp khóa-giá trị, cho phép truy xuất giá trị dựa trên khóa. Bảng băm được sử dụng để lưu trữ và truy xuất dữ liệu một cách nhanh chóng trong các ứng dụng như cơ sở dữ liệu, bộ đệm và bộ lọc. Các phép toán thường được sử dụng trên bảng băm bao gồm thêm phần tử vào bảng băm, lấy giá trị từ khóa và xóa phần tử khỏi bảng băm.
* Cây (Tree): là một cấu trúc dữ liệu phân cấp, mỗi nút của nó bao gồm một giá trị và một danh sách các nút con. Cây được sử dụng để lưu trữ các dữ liệu phân cấp như cấu trúc của một tệp hoặc cây phân loại của một hệ thống tập tin. Các phép toán thường được sử dụng trên cây bao gồm thêm nút vào cây, xóa nút khỏi cây và tìm kiếm nút trong cây.

2. Kiến trúc máy tính

* CPU (Central Processing Unit): là trái tim của máy tính, nó làm nhiệm vụ thực hiện các phép tính và điều khiển các hoạt động khác của hệ thống. CPU bao gồm các thành phần chính như bộ điều khiển (Control Unit) và bộ xử lý (ALU - Arithmetic Logic Unit).
* Bộ nhớ (Memory): là nơi lưu trữ dữ liệu và chương trình mà CPU sử dụng. Bộ nhớ được chia thành nhiều loại như bộ nhớ ROM (Read-Only Memory) và bộ nhớ RAM (Random Access Memory). Bộ nhớ ROM chứa các chương trình được lưu trữ sẵn trong hệ thống, còn bộ nhớ RAM là nơi lưu trữ dữ liệu và chương trình đang được thực thi.
* Bộ điều khiển (Control Unit): là thành phần quản lý hoạt động của hệ thống, nó điều khiển các hoạt động của CPU và các thiết bị khác như bộ nhớ và thiết bị ngoại vi. Bộ điều khiển giải mã các lệnh được gửi đến CPU và điều khiển các hoạt động của CPU để thực hiện các lệnh đó.
* Bộ phân phối (Bus): là thành phần kết nối các thành phần của hệ thống với nhau, bao gồm các bus dữ liệu, bus địa chỉ và bus điều khiển. Bus dữ liệu chịu trách nhiệm truyền dữ liệu giữa CPU, bộ nhớ và các thiết bị ngoại vi, trong khi bus địa chỉ quản lý việc truy xuất đến các vị trí bộ nhớ và bus điều khiển điều khiển hoạt động của các thành phần khác trong hệ thống.
* Thiết bị nhập xuất (Input/Output devices): là các thiết bị cho phép nhập dữ liệu từ bên ngoài vào máy tính hoặc xuất dữ liệu từ máy tính ra bên ngoài.
* Thiết bị lưu trữ (Storage devices): là các thiết bị cho phép lưu trữ dữ liệu lâu dài như ổ cứng, ổ đĩa mềm, ổ đĩa CD/DVD, thẻ nhớ, USB, …
* Thiết bị ngoại vi (Peripheral devices): là các thiết bị giao tiếp với máy tính bên ngoài như bàn phím, chuột, màn hình, máy in, loa, microphone, webcam, …

## 3. Bộ nhớ máy tính

Bộ nhớ máy tính (Memory) là một thành phần quan trọng trong kiến trúc máy tính. Bộ nhớ là nơi lưu trữ các chương trình và dữ liệu mà máy tính đang xử lý trong khi nó đang chạy. Bộ nhớ được chia thành hai loại: bộ nhớ chính (Main memory) và bộ nhớ thứ cấp (Secondary memory).

* Bộ nhớ chính (Main memory): Bộ nhớ chính hay còn gọi là bộ nhớ trực tiếp (RAM - Random Access Memory), là nơi lưu trữ tạm thời các chương trình và dữ liệu mà máy tính đang xử lý trong khi nó đang chạy. Bộ nhớ chính có thể được đọc và ghi dữ liệu nhanh chóng và dễ dàng, và nó rất quan trọng đối với hiệu suất của máy tính. Khi nguồn điện được cung cấp cho máy tính, bộ nhớ chính sẽ được khởi động và các chương trình được sao chép vào bộ nhớ để được xử lý.
* Bộ nhớ thứ cấp (Secondary memory): Bộ nhớ thứ cấp hay còn gọi là bộ nhớ ngoài (Secondary storage), là nơi lưu trữ dữ liệu lâu dài của máy tính như các tệp tin, hệ điều hành, phần mềm ứng dụng, … Bộ nhớ thứ cấp thường có dung lượng lớn hơn so với bộ nhớ chính và có thể lưu trữ dữ liệu trong thời gian dài mà không cần thiết phải hoạt động liên tục. Ví dụ như ổ cứng, ổ đĩa mềm, ổ đĩa CD/DVD, thẻ nhớ, USB, …

## 4. Lỗi tràn bộ đệm

Lỗi tràn bộ đệm (Buffer Overflow) là một lỗi phổ biến trong lập trình, khi một chương trình cố gắng ghi vào bộ nhớ ngoài kích thước đã được xác định trước hoặc truy cập vào một phần của bộ nhớ mà không được phép truy cập. Điều này có thể xảy ra khi một chương trình cố gắng ghi dữ liệu vào bộ đệm một cách vượt quá giới hạn đã được cấu hình, và ghi vào những vùng bộ nhớ được xác định cho mục đích khác nhau.

Khi một lỗi tràn bộ đệm xảy ra, dữ liệu có thể bị ghi đè lên dữ liệu khác trong bộ nhớ hoặc dữ liệu bên ngoài có thể bị tấn công bởi hacker. Nếu tấn công được thực hiện thành công, hacker có thể thực hiện các hành động như thực thi mã độc, tạo tài khoản truy cập hoặc lấy thông tin nhạy cảm từ máy tính của người dùng.

Lỗi tràn bộ đệm có thể xảy ra do nhiều nguyên nhân khác nhau, chẳng hạn như không kiểm tra đầu vào từ người dùng, cấu trúc dữ liệu không được thiết kế tốt hoặc lỗi lập trình.

Để ngăn chặn lỗi tràn bộ đệm, người lập trình có thể sử dụng các phương pháp như kiểm tra đầu vào, kiểm tra giới hạn kích thước của bộ đệm và thiết kế cấu trúc dữ liệu tốt hơn để tránh tình trạng tràn bộ đệm. Ngoài ra, người dùng cũng nên cập nhật các bản vá và bảo mật của hệ thống để giảm thiểu nguy cơ bị tấn công bởi hacker sử dụng lỗi tràn bộ đệm.

## 5. Tìm hiểu về gdb: các câu lệnh, cách gỡ lỗi sử dụng gdb

GDB (GNU Debugger) là một công cụ gỡ lỗi mã nguồn mở phổ biến trong lập trình, được sử dụng để phân tích lỗi trong mã nguồn, tìm ra nguyên nhân gây lỗi và sửa chữa chúng. GDB cung cấp một loạt các câu lệnh để giúp phân tích lỗi và tìm kiếm các vấn đề trong mã nguồn.

Các câu lệnh cơ bản trong GDB:

* *run* hoặc *r*: Chạy chương trình.
* *break* hoặc *b*: Thiết lập điểm ngắt tại một vị trí cụ thể trong mã nguồn.
* *step* hoặc *s*: Thực hiện từng lệnh trong chương trình.
* *next* hoặc *n*: Thực hiện từng lệnh và không đi vào các hàm con.
* *continue* hoặc *c*: Tiếp tục thực thi chương trình cho đến khi gặp điểm ngắt.
* *print* hoặc *p*: In giá trị của một biến hoặc biểu thức.
* *backtrace* hoặc *bt*: In ra thông tin về vị trí của các hàm và trạng thái của các biến trong stack.

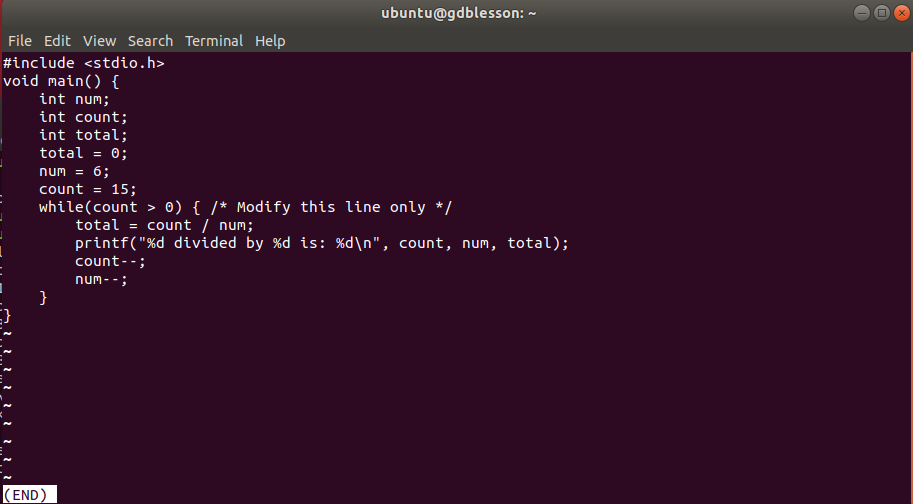
Cách sử dụng GDB để gỡ lỗi:

* Compile chương trình với tùy chọn *-g* để tạo ra thông tin gỡ lỗi.
* Khởi động GDB bằng cách gõ lệnh *gdb <tên chương trình>.*
* Thiết lập các điểm ngắt để theo dõi mã nguồn và xem giá trị của các biến trong quá trình thực thi.
* Thực thi chương trình bằng lệnh *run*.
* Sử dụng các câu lệnh *step, next, print* để theo dõi mã nguồn và giá trị của các biến.
* Nếu gặp lỗi, sử dụng lệnh *backtrace* để xem thông tin về các hàm và biến trong stack.
* Sửa lỗi và chạy lại chương trình để kiểm tra.
* Khi hoàn tất, sử dụng lệnh *quit* để thoát khỏi GDB.

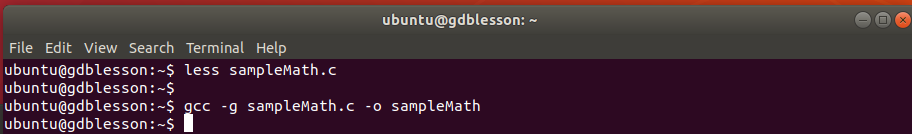
II. Thực Hành

1. GDB LESSON

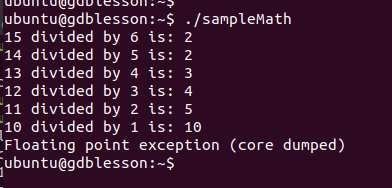
*less sampleMath.c*

**

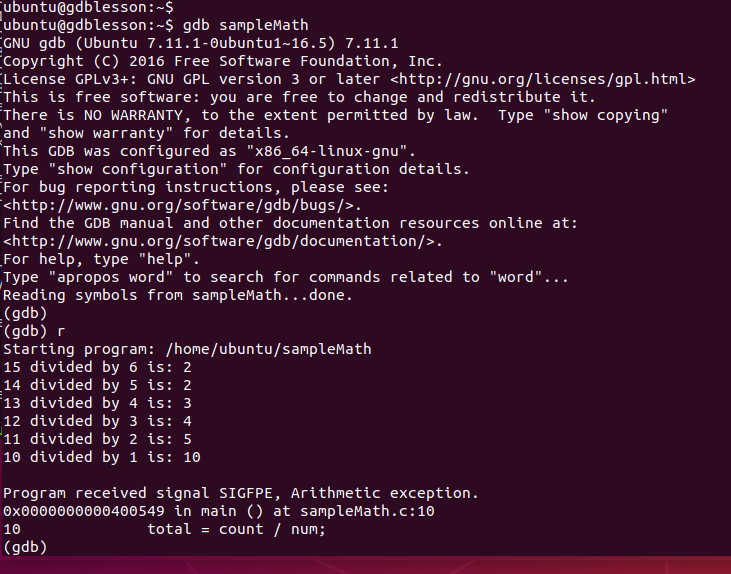
*gcc -g sampleMath.c -o sampleMath*

**

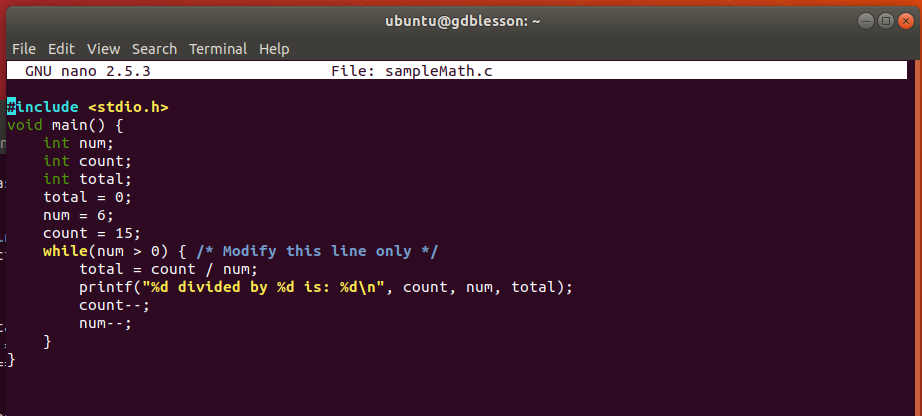
*./sampleMath*

**

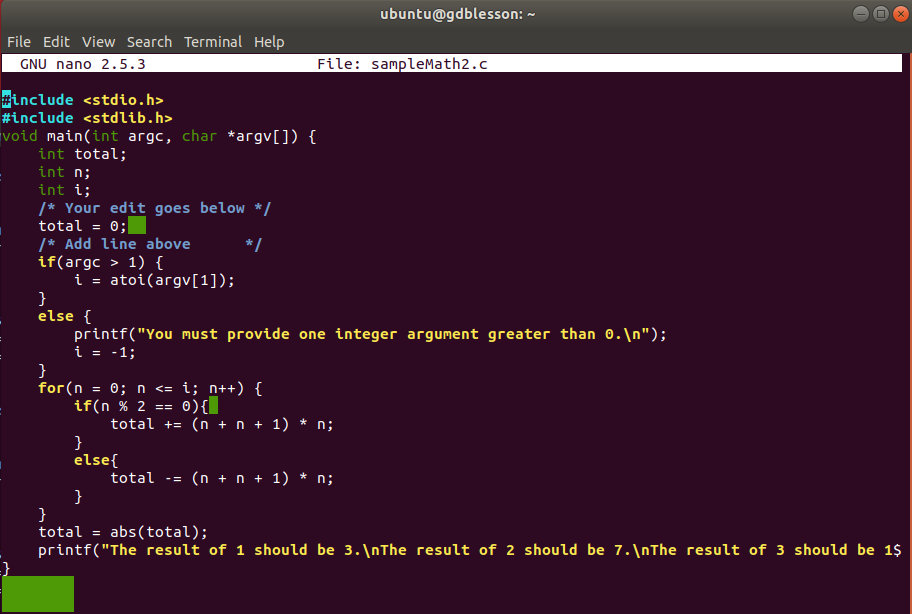
*gdb sampleMath*

**

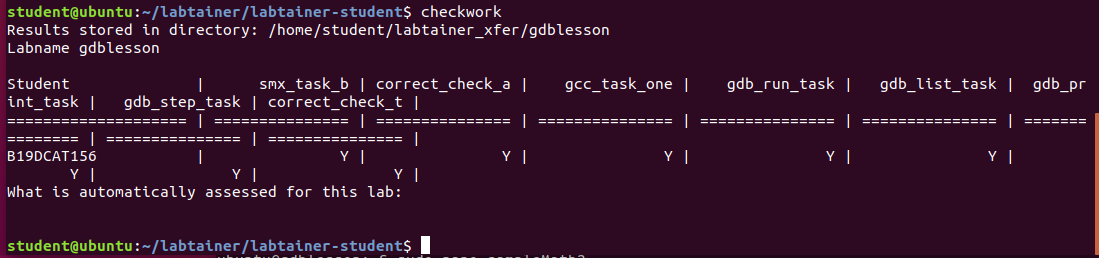
*sudo nano sampleMath.c*

**

*sudo nano sampleMath2.c*

**

*Checkwork*

**