

1

Lab

PHỤC VỤ MỤC ĐÍCH GIÁO DỤC
FOR EDUCATIONAL PURPOSE ONLY

Biểu diễn các kiểu dữ liệu và các phép tính toán bit

Thực hành môn Lập trình An toàn

Tháng 9/2025

Lưu hành nội bộ

<Nghiêm cấm đăng tải trên internet dưới mọi hình thức>

A. TỔNG QUAN

1. Mục tiêu

- Luyện tập với các phép tính toán bit và biểu diễn dữ liệu thông qua ngôn ngữ C.
- Làm quen với Linux và biên dịch chương trình C trên Linux.

2. Kiến thức nền tảng

B. CHUẨN BỊ MÔI TRƯỜNG

1. Môi trường Linux

- Ưu tiên Debian-based Distro: Ubuntu, Kali Linux, ...
- Cài đặt:
 - Máy ảo (VMWare Workstation/Fusion, VirtualBox, QEMU, ...).
 - Máy thật.
 - WSL (Windows).
 - Docker/DevContainer
 - ...

2. Trình biên dịch mã nguồn C

- **gcc:** `gcc -O2 -Wall -Wextra -o out_name input.c`.`
- **clang:** `clang -O2 -Wall -Wextra -o out_name input.c`.`

3. Trình soạn thảo

- Notepad/Notepad++.
- VSCode.
- Vi/Vim/Neovim/Helix.

C. THỰC HÀNH

1. Các phép biến đổi cơ bản

Trong ngôn ngữ lập trình C, **toán tử bitwise** (toán tử thao tác trên bit) được dùng để xử lý dữ liệu ở cấp độ nhị phân. Thay vì làm việc với giá trị số học thông thường, các toán tử này thao tác trực tiếp trên từng **bit** của toán hạng. Chúng thường được dùng trong các bài toán tối ưu bộ nhớ, xử lý cờ (flag), lập trình hệ thống, lập trình nhúng hoặc các tình huống cần thao tác nhanh trên dữ liệu nhị phân.

- **AND (&):** So sánh từng cặp bit của hai toán hạng. Bit kết quả bằng 1 khi cả hai bit đều bằng 1.
- **OR (|):** So sánh từng cặp bit, bit kết quả bằng 1 nếu ít nhất một trong hai bit bằng 1.
- **XOR (^):** Trả về 1 nếu hai bit khác nhau, 0 nếu giống nhau.
- **NOT (~):** Đảo bit (bit 0 thành 1, bit 1 thành 0). Trong C, toán tử này còn ảnh hưởng đến dấu do biểu diễn bù 2.
- **Dịch trái (<<):** Dịch toàn bộ bit sang trái n vị trí, thêm 0 vào bên phải. Tương đương với nhân cho 2^n .
- **Dịch phải (>>):** Dịch toàn bộ bit sang phải n vị trí. Tương đương với chia cho 2^n (lấy phần nguyên).

Bài thực hành 1 – 2đ: Thực hiện viết một chương trình C tính các phép toán sau đây và in ra kết quả đồng thời giải thích lý do:

- `0xC56A8FA37DD25 ^ 0x146C522CB8051`
- `~0x28378u`

Đảm bảo:

- In ra theo đúng kiểu dữ liệu.
- Không bị mất bit.
- Không bị sai lệch dấu dương/âm.

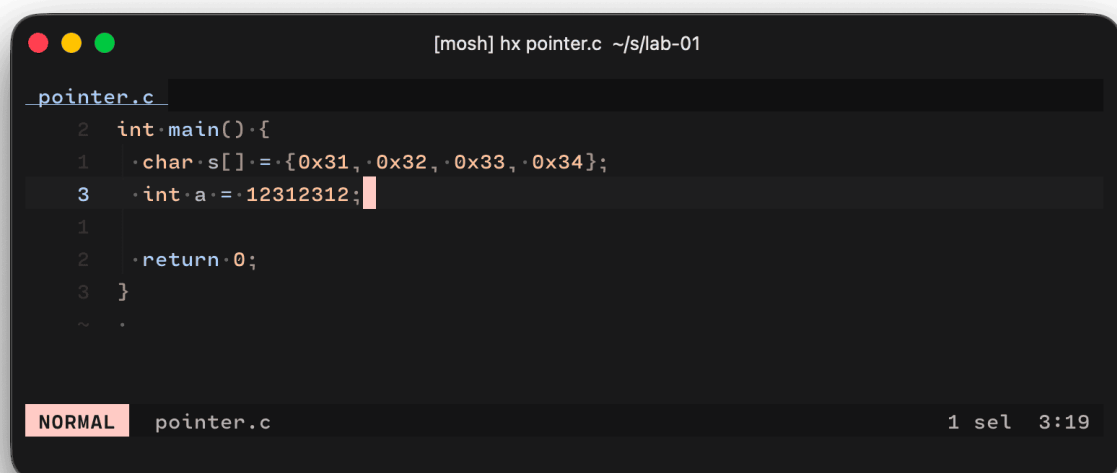
2. Biểu diễn các kiểu dữ liệu

Trong ngôn ngữ C, chuỗi (string) được hiểu là một mảng các ký tự kết thúc bởi ký tự đặc biệt '\0' (null-terminator). Mỗi phần tử trong chuỗi là một ký tự ASCII, và việc lưu trữ chuỗi thực chất là lưu trữ liên tiếp các giá trị ký tự trong bộ nhớ. Chuỗi có thể được khai báo theo nhiều cách, ví dụ `char s[] = "Hello";` hoặc `char *s = "Hello";`.

Con trỏ là một biến đặc biệt có chức năng lưu trữ địa chỉ của một biến khác trong bộ nhớ. Con trỏ có kiểu dữ liệu tương ứng với đối tượng mà nó trỏ tới, ví dụ `int *p` là con trỏ tới số nguyên, còn `char *p` thường dùng để trỏ tới chuỗi. Khi làm việc với con trỏ, toán tử `&` được sử dụng để lấy địa chỉ, còn toán tử `*` để truy cập giá trị tại địa chỉ đó. Con trỏ rất quan trọng trong C vì cho phép thao tác trực tiếp với bộ nhớ, quản lý mảng, truyền tham chiếu cho hàm, cấp phát động (malloc, calloc, free), và xây dựng các cấu trúc dữ liệu phức tạp như danh sách liên kết, cây hoặc đồ thị. Tuy nhiên, nếu không sử dụng cẩn thận, con trỏ có thể gây lỗi truy cập bộ nhớ hoặc rò rỉ bộ nhớ.

Bài thực hành 2 – 2đ: Thực hiện viết tiếp mã nguồn `pointer.c` đã được cung cấp theo các yêu cầu sau:

1. Tạo 1 con trỏ với địa chỉ là giá trị của `a`.
2. Sử dụng con trỏ tới `s`, thay đổi giá trị trong `s` sao cho `s` thành 1 chuỗi hoàn chỉnh.
3. Thay đổi 4 phần tử của `s` thành 4 bytes đầu của con trỏ của `s`.



```
[mosh] hx pointer.c ~/s/lab-01

pointer.c
2  int main() {
1  char s[] = {0x31, 0x32, 0x33, 0x34};
3  int a = 12312312;
1
2  return 0;
3  }
~

NORMAL pointer.c 1 sel 3:19
```

Bài thực hành 3 – 3đ: Biểu diễn 1 một chuỗi "Losers use LLM" chỉ bằng cách biến đổi số -2313L và các số từ 1 đến 9 và các bit operator (`&`, `|`, `<<`, `>>`, `~`, `^`). Sau đó in ra kết quả. Sử dụng mã nguồn `loser.c` được cung cấp.

Quy tắc:

1. Số base phải luôn là toán hạng bên trái của các toán tử. Ví dụ:
 - Sai: `1 << ~-2313`
 - Đúng: `~-2313 << 1`
2. Các số từ 1 đến 9 luôn là toán hạng bên phải. Ví dụ:
 - Sai: `1 | 9`

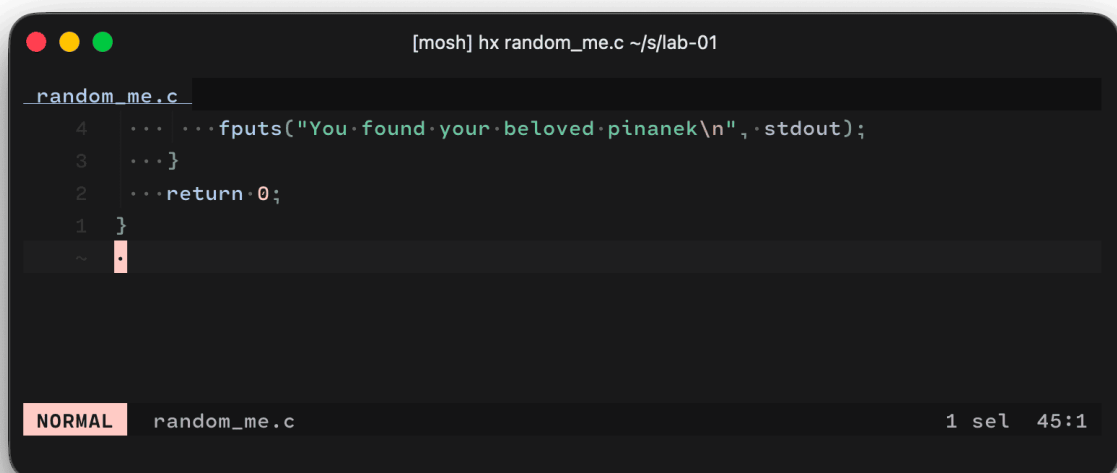
- Đúng: -2313 | 9



```
[mosh] hx loser....  
loser.c  
1 #include <stdio.h>  
1 #include <stdint.h>  
2  
3 int main() {  
4     ...int32_t base = -2313;  
5  
6     ...return 0;  
7 }  
~ .  
NORMAL loser.c 1 sel 1:1  
Loaded 1 file.
```

3. Thử thách

Thử thách 1 – 3đ: Tìm ra chuỗi số để chương trình có exit code 0 và in ra thông điệp. Sử dụng mã nguồn random_me.c được cung cấp.



```
[mosh] hx random_me.c ~/s/lab-01  
random_me.c  
4     ...fputs("You found your beloved pinaneek\n", stdout);  
3     ...}  
2     ...return 0;  
1 }  
~ .  
NORMAL random_me.c 1 sel 45:1
```

D. YÊU CẦU & ĐÁNH GIÁ

- Sinh viên tìm hiểu và thực hành theo hướng dẫn, thực hiện **theo từng cá nhân**.
- Thực hiện báo cáo theo 2 hình thức:
 - Báo cáo trực tiếp với GVTH, chấm điểm trên lớp.
 - Nộp báo cáo kết quả gồm **4 file code** tương ứng với **4 bài tập** và chi tiết những việc mà bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có). Thời hạn nộp trong vòng 3 ngày và nộp trên trang courses.uit.edu.vn

Nội dung báo cáo bằng file:

- File **.PDF**. Tập trung vào nội dung, không mô tả lý thuyết.
- Đặt tên theo định dạng: **[Mã lớp]-LabX_MSSV1**.
Ví dụ: [NT230.Q1X.ANTT]-Lab1_2452xxxx.
- Nếu báo cáo có nhiều file, nén tất cả file vào file **.ZIP** với cùng tên file báo cáo.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Bài sao chép tương đương với 0 điểm.

Bài nộp trễ sẽ bị trừ 50% số điểm.

HẾT

Chúc các bạn hoàn thành tốt!