Training fresher

Outline:

- Thực hành 3 bài code C trong 5 ngày, pass 3, trả lời những câu hỏi liên quan
- Basic:
  - biến và phạm vi của biến (toàn cục, cục bộ)
  - sử dụng kiểu dữ liệu phù hợp, biết cách ép kiểu
  - sử dụng toán tử đa dạng
  - các cấu trúc `if - else; switch - case; for; while - do; do - while`
  - nắm được các keyword hay dùng: `continue; break; goto; extern; volatile; const`
  - nắm được các chỉ thị tiền xử lý: `#define; #undef; #ifdef; #ifndef; #else; #endif`
  - hàm truyền tham chiếu, tham trị
    - tham trị: giá trị
    - tham chiếu: địa chỉ (con trỏ)
  - mảng và các thao tác với mảng
  - chuỗi và các thao tác với chuỗi (thư viện string.h)
  - con trỏ và các thao tác với con trỏ
    - phép toán trên con trỏ
  - các thao tác cấp phát địa chỉ
  - các thao tác với struct, enum, union
- Advanced:
  - quá trình biên dịch
  - toán tử logic
  - 3 thuật toán sắp xếp và tìm kiếm
  - vùng lưu trữ của các loại biến

    - các loại con trỏ: con trỏ mảng, con trỏ hàm, hàm con trỏ, con trỏ cấp 2, con trỏ hằng, hằng con trỏ, hằng con trỏ hằng
    - phân vùng bộ nhớ trên RAM
    - coding convention
    - data structure: stack, binary heap, queue
    - callback function
    - dynamic array
- Related
    - Git, SVN tortoise
    - SDLC
    - UML, flow chart, sequence diagram

==========BASIC==========

Exercise 1:

Viết 1 chương trình C quản lý và đánh giá sinh viên dựa trên điểm số, các môn thi.

Yêu cầu:

- khai báo biến toàn cục để lưu số tổng lượng sinh viên, số sv xuất xắc, giỏi, khá
- phân loại sinh viên dựa trên điểm số
    - xuất sắc: >8, không môn nào dưới 7
    - giỏi: ≥7 && ≤8, không môn nào dưới 5
    - khá: <7
- nhập thông tin sinh viên: id, điểm từng môn (3 môn văn, toán, anh)
- tính điểm trung bình
- tính số sinh viên xs, giỏi, khá
- in ra thông tin của tất cả sv: id, điểm từng môn, điểm tb, phân loại

Exercise 2:

Viết 1 chương trình C cho phép người dùng khởi tạo tài khoản mới.

Yêu cầu:

- có thể sử dụng linh hoạt `switch case` và `if - else`
- nhập tên đăng nhập và mật khẩu (dùng string)
- mật khẩu cần thỏa mãn điều kiện sau:
  - đủ chữ, số, in hoa, in thường, kí tự đặc biệt (5 tiêu chí)
    - dùng bảng mã ASCII để kiểm tra
  - đủ 8 ký tự
    - dùng hàm trong thư viện string.h
  - phân loại độ yếu - mạnh - rất mạnh của mật khẩu trước khi khởi tạo tài khoản, chỉ tài khoản có mật khẩu mạnh hoặc rất mạnh mới được khởi tạo (thử lại 5 lần)
    - yếu: dưới 8 ký tự hoặc đủ ≤ 3 tiêu chí
    - mạnh: đủ 8 ký tự và đủ > 3 tiêu chí
    - rất mạnh: đủ 8 ký tự và 5 tiêu chí
    - sử dụng enum để định nghĩa yếu - mạnh - rất mạnh
- lưu tên đăng nhập và mật khẩu
- yêu cầu người dùng đăng nhập lại, kiểm tra tk và mk
  - đúng: print login success (sử dụng hàm trong thư viện string.h)
  - sai: print login fail
    - thử lại 3 lần
    - vẫn sai thì in ra tên người dùng và mk đã khởi tạo

Exercise 3:

Sử dụng chỉ thị tiền sử lý để lựa chọn phương thức nhập thông tin nhân viên (tuổi, id < 100) cho hàm main() dưới đây

```
int main() {

    importStaffInfoTeamA();
```

```
    importStaffInfoTeamB();

    return 0;
}

void importStaffInfoTeamA() {
    // nhập tổng số nhân viên
    // cấp phát động struct
    // nhập thông tin từng nhân viên (id, salary)
    // in ra thông tin tất cả nhân viên
}

void importStaffInfoTeamA() {
    // nhập tổng số nhân viên
    // cấp phát động struct
    // nhập thông tin từng nhân viên (id, name, salary)
    // in ra thông tin tất cả nhân viên
}
```

==========ADVANCED==========

Exercise 4:

Viết 1 chương trình C để tìm chỉ số của 1 phần tử bất kì trong mảng

Yêu cầu:

- Nhập vào số phần tử của mảng
- Tạo mảng (cấp phát động) và nhập mảng
- Nhập vào số cần tìm index
- Sắp xếp mảng
  - sử dụng 3 thuật toán sắp xếp: bubble soft, insertion soft, selection soft
  - sử dụng chỉ thị tiền xử lý để lựa chọn thuật toán sắp xếp
- Tìm kiếm index và in ra màn hình

Exercise 5:

Tạo 1 linked list có các function sau

- chèn phần tử vào vị trí bất kì
  - kiểm tra xem phần từ trước đó có available hay không
- xóa phần từ tại vị trí bất kì
- kiểm tra list có trống hay không
- tìm key và index
- sắp xếp linked list
- nhập, suất linked list

Exercise 6:

Revision

```c
// con trỏ cấp 2
int main() {
        int x = 0;
    int *ptr = &x;
    int **pptr = &ptr;

    printf("x = %d\n", x);
    printf("*ptr = %d\n", *ptr);
    printf("*ptr = %d\n", &(*ptr));
    *ptr = 5; // giá trị tại vùng nhớ con trỏ ptr trỏ đến được
    printf("x = %d\n", x);
    printf("*ptr = %d\n", *ptr);
    printf("*ptr = %d\n", &(*ptr));
    /*
    pptr là con có giá trị là &ptr
    *pptr trả về giá trị của con trỏ pptr là ptr (là &x)
    **pptr trả về giá trị của con trỏ ptr là x
    */
    **pptr = 10;
    printf("x = %d\n", x);
    printf("*ptr = %d\n", *ptr);
    printf("*ptr = %d\n", &(*ptr));
```

```
    return 0;
}

// con trỏ hàm fuction pointer là con trỏ trỏ đến hàm
void main() {
        int (*funcPtr)(int);

        int func(int a) {
        printf("param is %d\n", a);
        return 0;
    }

    funcPtr = func;
    funcPtr(6);

    return 0;
}

// hàm con trỏ là hàm trả về con trỏ
void main() {
        int a[] = {0; 1; 2}

        int* getAddress(int value) {
        int *ret;
        for(int i = 0; i < sizeof(a); i++) {
            if (a[i] == value) {
                ret = &a[i];
            }
        }
        return ret;
    }

        printf("address of value 1 is %d", getAddress(1));
}

// callback function là hàm được gọi như 1 param của 1 hàm kh
// Define a function that will be used as a callback
void my_callback_function(int num) {
```

```
    printf("Callback function called with value: %d\n", num);
}

// Define a function that accepts a function pointer as a par
void execute_callback(void (*callback)(int), int value) {
    // Call the callback function
    callback(value);
}

int main() {
    // Declare a function pointer and assign the address of t
    void (*callback_ptr)(int) = my_callback_function;

    // Call the function that takes a callback, passing the f
    execute_callback(callback_ptr, 42);

    return 0;
}
```

Exercise:

Sử dụng con trỏ cấp 2 để tạo mảng 2 chiều

Viết hàm tính tổng các members của mảng có cùng chỉ số hàng và cột (eg: a[0,0]) ⇒ function pointer

Viết hàm in ra tất cả phần tử của cột (hoặc hàng) bất kỳ trong mảng ⇒ callback

Exercise 7:

Tạo repo tên là Andre_Exercise7 dưới dạng public repo

https://github.com/Andrea3052/Exercise_7/commits/main/user_2.txt

- add ssh key
- clone repo to local machine
- tạo 2 file mới có tên là user_1.txt và user_2.txt và thêm đoạn text sau vào mỗi file

```
// user_1.txt
```

```
This is my code
```

```
// user_2.txt
This is my code
```

- commit và push 2 file lên github
- trên local
  - sửa file user_2.txt, thêm đoạn text sau và commit (không push)

```
// user_2.txt
This is my code
I'm developing feature 1
    Task 3
    Task 4
```

- trên repo github, sửa file user_2.txt, thêm đoạn text sau và commit

```
// user_2.txt
This is my code
I'm developing feature 1
    Task 1
    Task 2
```

- pull source code về, resolve confilic, sửa file user_2.txt, thêm đoạn text sau

```
....
    Task 5
    Task 6
```

- commit change mà dùng git rebase để tạo 1 commit duy nhất và push
- remember some useful command in git

```
git commit --amend
git cherry-pick
git rebase
```

```
git checkout
git diff
git stash
```