Introduction to Programming

Lab 04
Arrays – String – Struct – File

Department of Software Engineering-FIT-VNU-HCMUS

1.

Content

In this lab, we will practice the following topics:

- Divide the source code into multiple files
- Write functions manipulating in 2D arrays, strings, structures, and file input/output operations.

2. Assignments

W07: YY: 04 => A: YY: 01; H: YY: 04

W08: YY: 04 => A: YY: 01; H: YY: 04

W10: YY: 04

The assignments of W07, W08, and W10 are not overlapped.

Besides the main() function, students are asked to write at least one more function in the following assignments. The source code should be located in 3 different files (or more).

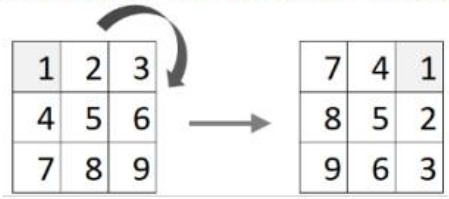
- A header file which containing all declarations.
- A source file which containing all function definitions.
- Another source file which containing the main() function.

2.1 Square Matrix

BÀI 5 Một ma trận vuông `A` cấp `N` được biểu diễn bằng một mảng hai chiều `N` dòng và `N` cột. Hãy cài đặt các hàm để thực hiện yêu cầu sau:

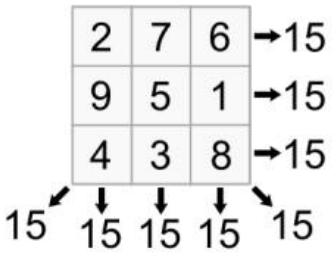
(a) *(10 điểm)* Xoay ma trận `A` một góc 90^o theo chiều kim đồng hồ.
Hình 1 minh họa việc xoay một ma trận vuông cấp 3.

Hình 1: Minh họa việc xoay ma trận vuông cấp 3



(b) *(10 điểm)* Kiểm tra xem ma trận vuông `A` có phải là *Magic square* cấp `N` hay không.
Biết rằng *Magic square* cấp `N` là một ma trận vuông cấp `N` chứa các số nguyên phân biệt từ `1` đến `N x N`, và tổng `N` phần tử trên mỗi hàng, cột, cũng như đường chéo đều bằng nhau. Hình 2 minh họa một *Magic square* cấp 3.

Hình 2: Một Magic square cấp 3



2.2 Matrix

BÀI 4 Cho ma trận 2 chiều có m dòng và n cột chứa các ký tự. Hãy viết chương trình thực hiện yêu cầu sau:

(a) *(10 điểm)* Trên biên của ma trận sẽ có 2 vị trí chứa 2 ký tự khác với các vị trí khác. Hãy tìm 2 vị trí này in ra toạ độ (dòng, cột).

(b) *(15 điểm)* Từ vị trí đầu tiên tìm được, có thể đi vào bên trong ma trận bằng cách đi theo các ký hiệu có hình *. Hãy tìm xem để đi từ vị trí đầu tiên đến vị trí thứ hai phải đi qua bao nhiêu ký hiệu hình *. Biết rằng chỉ có thể đi theo 4 hướng TRÊN, DƯỚI, TRÁI, PHẢI.

Giới hạn yêu cầu:

- Số dòng và số cột của ma trận sẽ không lớn hơn 100.
- Hai vị trí tìm được của yêu cầu (a) sẽ không bao giờ nằm ở 4 góc của ma trận.
- Chỉ có duy nhất một con đường đi từ vị trí đầu tiên đến vị trí thứ hai.

Ví dụ minh hoạ: Cho ma trận có 6 dòng và 7 cột như dưới đây

#	#	#	#	#	#	#
#	*			*	*	.
#				*		#
#	*		*	*		#
#		*	*		*	#
#	#	.	#	#	#	#

- Với yêu cầu của câu (a), Vị trí đầu tiên là (1, 6). Vị trí thứ hai là (5, 2).
- Với yêu cầu của câu (b), đi từ vị trí đầu tiên đến vị trí thứ hai cần đi qua 7 ký hiệu hình *.

2.3 Secret

Câu 2 (25 *điểm*).

Nhằm trao đổi bí mật với nhau, hai bạn trong lớp đã thiết kế một quy tắc trao đổi mật mã thông qua một ma trận B công khai chung gồm nRow × nCol số nguyên cho trước. Cụ thể, người giải mật mã sẽ được cung cấp dãy số A gồm k số nguyên (k ≤ nCol) và được quan sát ma trận B gồm nRow hàng và nCol cột cho trước.

a. (15 điểm) Biết rằng trong nRow hàng của ma trận, có **duy nhất một hàng** chứa nội dung của dãy A. Mật mã cần tìm là tổng của các số nguyên còn lại trên hàng chứa A nhưng không bao gồm dãy A. Với những ràng buộc đã cho, hãy giúp người giải mã viết hàm xác định giá trị mật mã cần tìm.

Ví dụ: Với dãy A gồm 3 số nguyên {5, 6, 7} và ma trận B gồm nRow = 4 hàng, nCol = 5 cột như dưới đây:

1	12	33	3	5
2	44	51	94	83
4	**5**	**6**	**7**	**8**
4	5	5	5	8

Thì dòng thứ 2 chứa dãy A gồm các số {4, 5, 6, 7, 8}. Do đó, mật mã cần tìm là tổng của 4 và 8 là 12.

b. (10 điểm) Ở mức độ thử thách hơn, ma trận `B` lúc này có thể có nhiều hàng và nhiều cột chứa nội dung của dãy `A`, biết rằng khi so khớp `A` trong ma trận `B`, ta có thể đọc hàng/cột đang được xét theo cả 2 chiều xuôi (trái → phải, trên → dưới) hoặc ngược (phải → trái, dưới → trên). Viết hàm đếm xem có tổng cộng bao nhiêu lần dãy mật mã `A` xuất hiện trong ma trận `B` cho trước.

Ví dụ với dãy `A` gồm `3` số nguyên `{3, 2, 1}` và ma trận `B` như sau:

3	2	1	2	3	1
3	2	1	3	2	1
2	0	2	1	7	9
1	2	3	2	1	6

Ta có:

- hàng `0`: tại `B(0, 0)` → `B(0, 2)`; `B(0, 4)` → `B(0, 2)`
- hàng `1`: tại `B(1, 0)` → `B(1, 2)`; `B(1, 3)` → `B(1, 5)`
- hàng `3`: tại `B(3, 2)` → `B(3, 4)`; `B(3, 2)` → `B(3, 0)`
- cột `0`: tại `B(1, 0)` → `B(3, 0)`
- cột `2`: tại `B(3, 2)` → `B(1, 2)`

Vậy hàm sẽ trả về `8` nghĩa là mật mã `A` xuất hiện tổng cộng `8` lần trong ma trận `B` cho trước.

2.4 Average

BÀI 4 Cho tập tin văn bản `Input.txt` chứa danh sách các số nguyên với nội dung theo quy ước sau:

- Dòng đầu tiên chứa một số nguyên N ($N > 0$) cho biết số lượng phần tử trong danh sách.
- Dòng thứ hai chứa danh sách các số nguyên cách nhau bởi (ít nhất một) khoảng trắng.

Ví dụ, một tập tin `Input.txt` được cho phía dưới:

```
6
12 40 -11 15 15 -79
```

Hãy thực hiện các yêu cầu sau:

(a) *(10 điểm)* Cài đặt hàm đọc danh sách các số nguyên `A` gồm `n` phần tử từ tập tin `Input.txt` theo mô tả phía trên.

(b) *(10 điểm)* Cài đặt hàm tính trung bình cộng của các số nguyên dương (> 0) có trong danh sách các số nguyên `A` gồm `n` phần tử. Nếu không có số nguyên dương nào trong danh sách thì trả về giá trị `0`.

(c) *(10 điểm)* Sử dụng (các) hàm đã thực hiện ở trên để cài đặt hàm xuất danh sách các số nguyên `A` gồm `N` phần tử cùng trung bình các số nguyên dương của danh sách ra một tập tin văn bản `Output.json` như mô tả phía dưới. Lưu ý giá trị trung bình cộng được làm tròn 2 chữ số sau dấu chấm.

```
{
    "DanhSach": [danh_sach],
    "TrungBinhCong":gia_tri_tinh_duoc
}
```

Ví dụ với tập tin `Input.txt` ở trên thì tập tin `Output.json` sẽ như sau:

```
{
    "DanhSach": [12,40,-11,15,15,-79],
    "TrungBinhCong":20.50
}
```

2.5 Binary

BÀI 2 Chuỗi $x_n x_{n-1} ... x_2 x_1 x_0$ $(n \geq 0)$ được gọi là chuỗi nhị phân nếu các kí tự x_i là `0` hoặc `1`, tức là $x_i \in \{0, 1\}$ với mọi $i = 0, 1, ..., n$.

Giá trị của chuỗi nhị phân $x_n x_{n-1} ... x_2 x_1 x_0$ là số nguyên không âm được tính bằng công thức:

$\sum_{i=0}^{n} x_i 2^i = x_0 + 2x_1 + 4x_2 + ... + 2^{n-1} x_{n-1} + 2^n x_n$

Ví dụ, `10100` là chuỗi nhị phân có giá trị là `20` vì $2^2 + 2^4 = 4 + 16 = 20$.

(15 điểm) Hãy viết hàm `unsigned int convert(char bin[])` để tính giá trị của một chuỗi nhị phân được cho bằng chuỗi ký tự `bin`. Ví dụ, lời gọi hàm `convert("10100")` trả về giá trị `20`.

Lưu ý: sinh viên KHÔNG được phép dùng các hàm thư viện có sẵn (như hàm `pow`,...).

2.6 Ticket

BÀI 3 Chuỗi chiếu phim *NTHK* xây dựng phần mềm quản lý doanh thu. Phần mềm quản lý doanh thu cần nhiều thông tin và chức năng khác nhau liên quan đến khách hàng, phim, vé xem phim... Bạn được giao nhiệm vụ tổ chức thông tin và cài đặt các chức năng liên quan đến vé xem phim.

Biết rằng, thông tin chung của một vé xem phim bao gồm: Tên phim, thời gian bắt đầu (giờ, phút), tên rạp chiếu phim, giá thức ăn kèm theo, hệ số.
Có 2 loại vé xem phim với các thông tin thêm như sau:

- Vé thường có hệ số là `1.0`; giá vé được tính theo công thức là: [hệ số × `80000` + giá thức ăn kèm theo]
- Vé combo có hệ số là `1.5`; giá vé được tính theo công thức là: [hệ số × `80000` + giá thức ăn kèm theo] × `0.9`

Các chức năng liên quan đến vé xem phim bao gồm: nhập danh sách vé xem phim, tính tổng doanh thu thông qua giá vé tại một thời điểm nhất định.

Sinh viên thực hiện các yêu cầu sau:

(a) *(5 điểm)* Khai báo các cấu trúc (struct) cần thiết để thể hiện các thông tin liên quan đến vé xem phim đã cho.

(b) *(10 điểm)* Viết hàm nhập danh sách các vé xem phim từ bàn phím.

(c) *(10 điểm)* Viết hàm tính tổng doanh thu **dựa trên giá vé** cho tất cả vé xem phim có thời gian bắt đầu lúc *hh* giờ *mm* phút.

2.7 Encoding

Câu 3 (20 *điểm*).

Giá trị nguyên của một ký tự được quy ước là vị trí của nó trong bảng chữ cái alphabet, với vị trí bắt đầu từ 0, nghĩa là ký tự 'a' tương ứng với giá trị 0, tương tự với 'b' → 1, 'c' → 2, …, 'z' → 25. Giá trị nguyên của một chuỗi (viết thường) được quy ước là số được đổi từ chuỗi bằng cách ghép nối giá trị nguyên liên tục của các ký tự có trong chuỗi đó.

Ví dụ: "bac" → "102" → 102 (được ghép từ b – 1, a – 0 và c – 2), và "ab" → "01" → 1.

Viết hàm **`checkEqual`** nhận vào 3 tham số `first`, `second`, `target` là các chuỗi đầu vào có tối đa 8 ký tự là chữ cái alphabet viết thường. Hàm sẽ kiểm tra xem liệu tổng giá trị nguyên của 2 chuỗi `first` và `second` có bằng chuỗi `target` hay không. Hàm trả về `true` nếu bằng, ngược lại trả về `false`.

Lưu ý:

- Sinh viên thiết kế hàm sao cho thực hiện đúng chức năng. **KHÔNG** nhập và in ra giá trị trong hàm.
- Sinh viên **KHÔNG** được sử dụng lớp `std::string` của `C++`, bắt buộc phải xử lý với mảng ký tự (`c-string`).

Ví dụ:

- Với chuỗi `first = "bac", second = "ba"` và `target = "bbc"` như sau, kết quả trả về từ hàm là `true`.

  Giải thích: `"bac" → 102, "ba" → 10, "bbc" → 112`.

  Ta có tổng `102 + 10 = 112`.

- Với chuỗi `first = "ad", second = "da", target = "dad"`, kết quả trả về là `false`.

  Giải thích: `"ad" → 3, "da" → 30, "dad" → 303`.

  Ta có tổng `3 + 30` khác `303`.

2.8 Order

Câu 4 (30 *điểm*).

Cho thông tin của một đơn hàng được giao trong địa bàn TP. Hồ Chí Minh bao gồm:

- Mã đơn hàng: chuỗi ký tự (tối đa 17 ký tự).
- Tên người nhận: chuỗi ký tự (tối đa 15 ký tự).
- Địa chỉ: chuỗi ký tự (tối đa 50 ký tự).
- Số điện thoại: chuỗi ký tự (tối đa 10 ký tự).
- Khối lượng: số thực (đơn vị kilogram).

a. (5 điểm) Khai báo cấu trúc **Package** dựa theo mô tả phía trên nhằm lưu thông tin của một đơn hàng.

b. (10 điểm) Cho tập tin đầu vào `input.txt` chứa thông tin của các đơn hàng với:

- Dòng đầu tiên: chứa thông tin các thông số của đơn hàng
- Mỗi dòng tiếp theo chứa thông tin của một đơn hàng cụ thể

Viết hàm đọc và lưu trữ danh sách đơn hàng của các đơn hàng. Biết rằng, cấu trúc tập tin `input.txt` như sau:

```
Ma don hang|Ten nguoi nhan|Dia chi|So dien thoai|Khoi luong
59392159392159392|Nguyen Teo|28, Le Duan, Ben Nghe, Q1|0909001001|0.3
58423158423158423|Tran Ty|34, Vo Van Tan, P5, Q3|0908123123|1.2
58522258522258522|Le Meo|220, Nguyen Xi, P26, Binh Thanh|0978123456|0.9
59283159283159283|Ly Suu|40, Bach Dang, P24, Binh Thanh|0962456567|0.5
```

c. (15 điểm) Biết rằng thông tin về địa chỉ nhận hàng của một đơn hàng bao gồm 4 thông tin lần lượt là số nhà, tên Đường, tên Phường, tên Quận được ngăn cách với nhau bởi dấu phẩy như cấu trúc trong file đầu vào `input.txt`.

Viết hàm lọc ra các đơn hàng được giao tới Quận `deliveryDist` (là tham số được truyền vào hàm), sau đó ghi ra tập tin `output.txt`. Ví dụ với tập tin `input.txt` minh họa trên và đối số được truyền vào hàm `deliveryDist="Binh Thanh"` thì nội dung tập tin `output.txt` là:

```
Ma don hang|Ten nguoi nhan|Dia chi|So dien thoai|Khoi luong
58522258522258522|Le Meo|220, Nguyen Xi, P26, Binh Thanh|0978123456|0.9
59283159283159283|Ly Suu|40, Bach Dang, P24, Binh Thanh|0962456567|0.5
```

2.9 Directory

Câu 2

Khoảng cách thư mục giữa hai tập tin được định nghĩa là số bước chuyển thư mục ít nhất để chuyển từ thư mục chứa tập tin này sang thư mục chứa tập tin kia. Một bước chuyển thư mục từ thư mục A được hiểu là chuyển sang thư mục cha trực tiếp hoặc con trực tiếp của A.

Minh họa:

```
/------- dirA
        |------- file1
        dirB
        |------- file2
        |------- dirC
                |------- file3
                |------- file4
```

- Khoảng cách file4 và file3 = 0
- Khoảng cách file4 và file2 = 1
- Khoảng cách file4 và file1 = 3
- Khoảng cách file2 và file1 = 2

Hãy viết hàm `int calcDirDistance( const char f1[], const char f2[] )` trả về khoảng cách thư mục giữa hai tập tin có đường dẫn f1 và f2.

Lưu ý: đường dẫn tập tin luôn bắt đầu bằng thư mục gốc "/".

Ví dụ: `f1 = "/dirB/dirC/file4", f2 = "/dirB/file2"  => Trả về 1`

2.10Wildcard

Câu 3

Ký tự đại diện '*' (wildcard) được dùng để đại diện cho không hoặc nhiều ký tự bất kỳ.

Một chuỗi ký tự có chứa ký tự đại diện '*' sẽ so khớp với một tập các chuỗi.

Minh họa:

Chuỗi đại diện	So khớp	Ví dụ
"app*"	Tập các chuỗi bắt đầu bằng "app"	"app", "apple", "appstore", ...
"*ion"	Tập các chuỗi kết thúc bằng "ion"	"ion", "lion", "function", ...
"the*quick*fox"	Tập các chuỗi bắt đầu bằng "the", kết thúc bằng "fox", và chứa "quick" ở giữa	"the very quick brown fox", "the-quickfox", ...

a) Hãy nêu ý tưởng giải thuật để kiểm tra xem chuỗi s chứa ký tự đại diện '*' có so khớp với chuỗi t không chứa ký tự đại diện hay không.

b) Hãy viết hàm `int strmatch(const char s[], const char t[])` cài đặt ý tưởng giải thuật ở câu a, hàm trả về 1 nếu s (chứa ký tự đại diện '*') so khớp với t (không chứa ký tự đại diện), trả về 0 nếu không so khớp.

2.11 Shape

Câu 4

Tập tin văn bản "SHAPES.TXT" lưu danh sách N hình trên mặt phẳng, vừa hình tam giác vừa hình tròn. Thông tin mỗi hình được lưu trên một dòng trong tập tin văn bản.

Định dạng lưu hình tam giác: [loại hình-0, (tọa độ đỉnh 1), (tọa độ đỉnh 2), (tọa độ đỉnh 3)]

Định dạng lưu hình tròn: [loại hình-1, (tọa độ tâm), bán kính]

Ví dụ: tập tin văn bản lưu danh sách 3 hình, theo thứ tự tam giác, tròn, tam giác

```
[0, (1.1, 2.5), (3.6, 5.0), (7.2, 2.5)]
[1, (2.2, 3.3), 8]
[0, (1.9, 2.4), (5.6, 8.2), (9.3, 2.4)]
```

Hãy thực hiện:

a) Khai báo 3 kiểu cấu trúc `Point`, `Triangle`, `Circle` biểu diễn các hình trên.

b) Viết hàm đọc tập tin "SHAPES.TXT" danh sách các hình theo cú pháp:

```
void read(struct Triangle tri[], int &n1, struct Circle cir[], int &n2)
```

Trong đó: - tri: danh sách hình tam giác đọc được - n1: số hình tam giác đọc được.

- cir: danh sách hình tròn đọc được - n2: số hình tròn đọc được.

2.12 Pangram

Question 4. A Pangram is a sentence containing every letter in the English Alphabet. (Lowercase and Uppercase character are considered same.)

Example: "*The quick brown fox jumps over the lazy dog*" is a Pangram because it contains all the characters from 'a' to 'z'.

a. Write a function to check if it a string is Pangram or not.

b. Write a function to print out all characters that are missing from the string, i.e., the characters that can make string a Pangram.

Example: Input: The quick brown fox jumps

Output: adglvyz