Bài tập lớn

Mô phỏng cảm biến ánh sáng và phân tích dữ liệu

I. Mô tả:

(Nhóm) sinh viên cần viết chương trình sử dụng C hoặc C++ với các hàm và cấu trúc dữ liệu phù hợp để mô phỏng hoạt động của các cảm biến ánh sáng sử dụng để đo cường độ ánh sáng trong nhà. Thông số kỹ thuật chính của cảm biến như sau:

- Dải đo: $0.1 \div 40000\ Lux$
- Độ phân giải: $0.01\ Lux$

Các nhiệm vụ (task) cần thực hiện bao gồm:

1. Task 1:

Viết chương trình cho phép người dùng cung cấp số cảm biến, thời gian trích mẫu và khoảng thời gian đo sử dụng câu lệnh command-line để xuất ra dữ liệu mô phỏng. Khuôn mẫu của câu lệnh command-line như sau:

```
C:\\lux_sim -n [num_sensors] -st [sampling] -si [interval]
```

Trong đó:

- `lux_sim`: là tên chương trình sau khi biên dịch.
- `-n [num_sensors]` là cặp tham số đầu vào để cung cấp số lượng cảm biến, `[num_sensors]` cần được thay thế bởi một số nguyên dương. Nếu 1 trong 2 tham số này không có trong câu lệnh command-line, chương trình cần đưa ra thông báo lỗi. Nếu cả 2 tham số này không xuất hiện, chương trình sẽ sử dụng số lượng cảm biến mặc định là 1 (một).
- `-st [sampling]` là cặp tham số đầu vào cung cấp thời gian trích mẫu, `[sampling]` phải được thay thế bởi một số nguyên dương, đơn vị quy ước là giây. Thời gian trích mẫu nhỏ nhất có thể là 1 giây. Nếu 1 trong 2 tham số này không có trong câu lệnh command-line, chương trình cần đưa ra thông báo lỗi. Nếu cả 2 tham số này không xuất hiện, chương trình sẽ sử dụng thời gian trích mẫu mặc định là 60 giây.
- `-si [interval]` là cặp tham số đầu vào cung cấp khoảng thời gian mô phỏng/đo, `[interval]` phải được thay thế bởi một số nguyên dương, đơn vị quy ước là giờ. Khoảng thời gian nhỏ nhất là 1 giờ. Nếu 1 trong 2 tham số này không có trong câu lệnh command-line, chương trình cần đưa ra thông báo lỗi. Nếu cả 2 tham số này không xuất hiện, chương trình sẽ sử dụng khoảng thời gian mặc định là 24 giờ.

Dữ liệu mô phỏng xuất ra bao gồm số hiệu cảm biến (sensor id), thời điểm đo mô phỏng (timestamp) và giá trị đo mô phỏng (value). Thời điểm bắt đầu mô phỏng tức thời điểm có giá trị đo đầu tiên được lấy từ thời điểm hiện tại của hệ thống (giờ trong máy tính khi bắt đầu chạy chương trình) trừ đi khoảng thời gian đo. Múi giờ sử dụng là mặc định, thường sẽ là giờ địa phương; sinh viên không cần sử dụng bất kỳ hàm nào hoặc tham biến nào để thay đổi múi giờ.

- Số hiệu cảm biến (sensor id) được xuất ra trong phạm vi từ 1 đến `num_sensors`, trong đó `num_sensors` là số lượng cảm biến người dùng cung cấp trong câu lệnh command-line, ví dụ nếu `num_sensors = 10` chương trình sẽ tạo ra 10 cảm biến có ids là 1, 2, 3,… , 10.
- Thời điểm đo (mô phỏng) cần được in ra với định dạng YYYY:MM:DD hh:mm:ss, trong đó:
  - YYYY – năm, MM – tháng, DD – ngày.
  - hh – giờ, mm – phút, ss – giây.

  Ví dụ: 2023:11:01 08:30:02
- Giá trị đo (mô phỏng) được tạo ngẫu nhiên với độ chính xác 2 chữ số sau dấu phẩy.

Chú ý: thời gian mô phỏng (khoảng thời gian và thời điểm) không phải là thời gian thực mà chỉ được tính toán trong mô phỏng, do đó sinh viên không sử dụng các hàm tạo trễ ví dụ như hàm sleep() hoặc dùng vòng lặp để tạo trễ.

Dữ liệu xuất ra cần được lưu trong một file đặt tên là "lux_sensor.csv"; nếu file đã tồn tại, chương trình cần ghi đè lên file cũ. File này cần tuân theo định dạng CSV (comma-separated values), mỗi trường dữ liệu được phân tách bởi dấu phẩy ",". Sinh viên tham khảo định dạng file CSV ở trong link này: https://www.ietf.org/rfc/rfc4180.txt. Các file dữ liệu xuất ra cần nằm trong cùng thư mục với file chạy chương trình.

Ví dụ: Người dùng chạy câu lệnh sau:

```
C:\\lux_sim -n 3 -st 60 -si 10
```

- Giả sử câu lệnh này chạy ở thời điểm 2023:11:11 10:00:00, thời gian bắt đầu mô phỏng sẽ là 2023:11:11 00:00:00. Cả thời điểm đầu và thời điểm chạy chương trình đều phải được tính trong file dữ liệu.
- Dữ liệu trong file "`lux_sensor.csv`" được thể hiện như sau:

```
id,time,value
1,2023:05:08 00:00:00, 50.01
2,2022:05:08 00:00:00,24.02
3,2023:05:08 00:00:00, 200.05
1,2023:05:08 00:01:00,100.12
2,2023:05:08 00:01:00,55.34
3,2023:05:08 00:01:00,160.49
…
1,2023:05:08 10:00:00,120.52
2,2023:05:08 10:00:00,90.40
3,2023:05:08 10:00:00,351.00
```

Dòng đầu tiên "id,time,value" là dòng tiêu đề.

2. Task 2:

Sinh viên cần viết một chương trình xử lý file csv có định dạng như file csv trong task 1. Chương trình phải được chạy với câu lệnh command-line có định dạng như sau:

```
C:\\lux_process [data_filename.csv] [location.csv]
```

Trong đó:

- `lux_process`: là file chạy chương trình đã được biên dịch
- `[data_filename.csv]` là file csv chứ dữ liệu cảm biến ánh sáng.
- `[location.csv]` là file csv chứa mã vị trí cài đặt cảm biến với định dạng như ví dụ sau:

```
id,location
1,2
2,3
3,4
4,10
5,2
…
9,5
```

Trong đó cột id là số hiệu cảm biến của tất cả các cảm biến có trong file `[data_filename.csv]` và cột location là mã khu vực lắp cảm biến như mô tả trong bảng 1 (Table 1).

Nếu câu lệnh chỉ có 1 hoặc không có file đầu vào nào thì được coi là không hợp lệ.

Ví dụ: `C:\\lux_process lux_sensor_ee3490e.csv location.csv`

Table 1. Type of location with related activities and the required lux level

Type of area with related activity	Code	Min required lux	Max required lux
Public areas with dark surroundings	1	20	50
Simple orientation for short visits	2	50	100
Areas with traffic and corridors - stairways, escalators and travelators - lifts – storage spaces	3	100	200
Working areas where visual tasks are only occasionally performed	4	100	150
Warehouses, homes, theaters, archives, loading bays	5	150	250
Coffee break room, technical facilities, ball-mill areas, pulp plants, waiting rooms,	6	200	400
Easy office work (e.g. photocopy area)	7	250	350
Class rooms	8	300	500
Normal office work, PC work, study library, groceries, show rooms, laboratories, check-out areas, kitchens, auditoriums	9	500	700
Supermarkets, mechanical workshops, office landscapes	10	750	850
Detailed drawing work, very detailed mechanical works, electronic workshops, testing and adjustments	11	1500	2000
Performance of visual tasks of low contrast and very small size for prolonged periods of time	12	2000	5000
Performance of very prolonged and exacting visual tasks	13	5000	10000
Performance of very special visual tasks of extremely low contrast and small size	14	10000	20000

Chương trình cần phải xử lý được ít nhất 10000 điểm dữ liệu tức là file đầu vào `data_filename.csv` có thể chứa 10000 dòng. Dữ liệu cần được xử lý như trong các task con. Khi chạy chương trình, thì tất cả các task con của task 2 này cần được xử lý và cho ra kết quả đồng thời.

a. Task 2.1:

Giả sử dữ liệu ánh sáng thực tế thực tế trong môi trường quan trắc chỉ có thể nằm trong dải $1 \div 30000\ lux$. Do đó các dữ liệu cảm biến hợp lệ phải nằm trong dải này, và các giá trị đo nằm ngoài khoảng này được coi là không hợp lệ (outliers). Chương trìn cần kiểm tra các giá trị không hợp lệ, tức là các giá trị ngoại lệ chứa trong file `[data_filename.csv` và lưu vào một file csv đặt tên là "lux_outlier.csv". Nếu file "lux_outlier.csv" đã tồn tại thì ghi đè lên file cũ. Nội dung của file outlier giống như trong ví dụ dưới, trong đó dòng đầu tiên "number of outliers: X" với X là số lượng các giá trị ngoại lệ đã được lọc ra từ file dữ liệu, trong ví dụ này X=3.

```
number of outliers: 3
id,time,value
1,2023:11:11 00:00:00,0.08
3,2023:11:11 19:03:00,-1.01
3,2023:11:11 21:06:00,35690.22
```

Chỉ các giá trị cảm biến hợp lệ sẽ được sử dụng để thực hiện các nhiệm vụ từ task 2.2 trở đi.

b. Tasks 2.2:

Giá trị lux được sử dụng để xác định điều kiện ánh sáng trong khu vực quan trắc như sau:

- Nếu lux value < Min required lux: dark
- Nếu lux value > max required lux: bright

- Nếu min required lux ≤ lux value ≤ max required lux: good

Trong đó min required lux và max required lux tương ứng với mỗi loại khu vực được cho trong bảng 1.

Nếu location code **không được cung cấp** trong file location, điều kiện ánh sáng sẽ được viết là "NA" (not available).

Chương trình cần tính giá trị lux trung bình mỗi giờ, ví dụ giá trị lux trung bình tại 2023:11:11 02:00:00 là giá trị trung bình của tất cả các giá trị lux trong khoảng thời gian từ 2023:11:11 01:00:00 đến 2023:11:11 01:59:59. Chương trình cũng cần xác định điều kiện ánh sáng tương ứng với giá trị tính được. Kết quả thu được cần lưu vào một file có tên là "lux_condition.csv" với định dạng tương tự như ví dụ dưới đây:

```
id,time,location,value,condition
1,2023:11:11 01:00:00,1,30.51,good
2,2023:11:11 01:00:00,2,400.03,bright
3,2023:11:11 01:00:00,10,200.00,dark
4,2023:11:11 01:00:00,0,240.00,NA
1,2023:11:11 02:00:00,1,101.02,bright
2,2023:11:11 02:00:00,2,75.02,good
3,2023:11:11 02:00:00,10,812.05,good
4,2023:11:11 02:00:00,0,2.05,NA
…
```

c. Tasks 2.3:

Xác định giá trị lớn nhất (max), nhỏ nhất (min) và giá trị lux trung bình của toàn bộ thời gian mô phỏng cho mỗi cảm biến. Kết quả thu được cần lưu trong file có tên là "lux_summary.csv" với định dạng tương tự như ví dụ dưới đây:

```
id, parameter,time,value
1, max,2023:11:11 08:30:00,350.80
1, min,2023:11:11 09:31:03,5.61
1, mean,10:00:00 ,200.54
2, max,2023:11:11 08:35:00,300.83
2, min, 2023:11:11 09:32:03,15.61
2, mean,10:00:00, 110.55
3, max, 2023:11:11 09:05:02,120.67
3, min, 2023:11:11 09:21:03,20.81
3, mean,10:00:00,70.59
…
```

Thời gian tương ứng với giá trị max và min là thời điểm mà giá trị đó xuất hiện lần đầu trong file đầu vào. Thời gian tương ứng với giá trị trung bình là khoảng thời gian mô phỏng.

3. Task 3:

Giả thiết người dùng cần gửi và nhận dữ liệu đầu ra của task 2.2 qua một giao thức truyền thông. Gói tin truyền thông là một chuỗi byte có cấu trúc như sau:

Table 2. Data packet frame

Start byte	Packet Length	ID	Time	Location	Lux	Checksum	Stop byte
0xA0 (1 byte)	1 byte	1 byte	4 bytes	1 byte	4 bytes	1 byte	0xA9 (1 byte)

Trong đó :

- Start byte (1 byte) là byte khởi đầu của gói tin và luôn có giá trị là 0xA0.
- Stop byte (1 byte) là byte kết thúc của gói tin và luôn có giá trị là 0xA9.
- Packet length là độ dài gói tin bao gồm cả byte khởi đầu và byte kết thúc.

- Id là số hiệu cảm biến (sensor ID) và luôn có giá trị dương (>0)
- Time là thời điểm đo đã chuyển đổi sang định dạng Unix timestamp format với đơn vị là giây.
- Lux là giá trị lux, là một số thực 4-byte biểu diễn theo chuẩn IEEE 754 single precision floating-point.
- Location là mã khu vực lắp cảm biến được định nghĩa trong bảng 1 và là một số nguyên 1 byte.
- Checksum là byte kiểm tra gói tin và được tính bằng cách sử dụng thuật toán mã bù 2 của nhóm byte [packet length, id, time, location, lux].

Tất cả trường dữ liệu số nguyên và số thực trong gói tin được biểu diễn theo big-endian.
Người dùng chạy chương trình task 3 với câu lệnh command-line như sau:

```
C:\\ lux_comm [input_file] [output_file]
```

Trong đó

- `[input_file]` là tên file đầu vào.
- `[output_file]` là tên file đầu ra.

Tùy thuộc vào định dạng của file đầu vào và file đầu ra, chương trình sẽ quyết định chuyển đổi dữ liệu từ file đầu vào sang định dạng mới trong file đầu ra.
Người dùng phải cung cấp cả 2 tên file, nếu không đó sẽ là câu lệnh không hợp lệ.

3.1 Trường hợp 1:

Nếu file đầu vào là file dữ liệu có định dạng giống như file đã mô tả trong task 2.2, file đầu ra sẽ là file text có phần mở rộng ".dat" chứa các gói tin có cấu trúc như bảng 2 (Table 2).
Ví dụ nếu người dùng gõ câu lệnh:

```
C:\\lux_comm lux_condition.csv hex_packet_ee3491.dat
```

Chương trình sẽ:

- Đọc từng dòng của file đầu vào, tức là file lux_condition.csv
- Chuyển đổi từng dòng dữ liệu thành gói tin như mô tả trong bảng 2, mỗi byte được phân tách bởi dấu cách và biểu diễn dưới dạng số hex, ví dụ:

  *Dòng dữ liệu "**2,2023:11:11 01:00:00,2,400.03,bright**" trong file "lux_condition.csv" được chuyển đổi thành*

  A0 0E 02 65 4E 6F A0 02 43 C8 03 D7 47 A9
- Viết mỗi gói tin đó trên một dòng của file đầu ra, tức là file hex_packet_ee3491.dat.
- Ghi đè lên file hex_packet_ee3491.dat nếu file đó đã tồn tại.
- Có thể xử lý được tối thiểu 10000 điểm dữ liệu, tức là file đầu vào lux_condition.csv có thể chứa 10000 dòng dữ liệu.

3.2 Trường hợp 2:

Nếu file đầu vào là một file text với phần mở rộng ".dat" chứa các gói tin như mô tả trong bảng 2 và mỗi gói tin nằm trên một dòng khác nhau, file đầu ra sẽ là một file csv có định dạng như đã mô tả trong task 2.2.
Ví dụ nếu người dùng chạy câu lệnh:

```
C:\\lux_comm hex_packet_ee3491.dat lux_condition.csv
```

Chương trình sẽ:

- Đọc từng dòng trong file đầu vào, tức là file: hex_packet_ee3491.dat
- Chuyển đổi từng gói tin thành các trường dữ liệu phù hợp và mỗi trường dữ liệu sẽ được phân tách bởi dấu phẩy như file đã mô tả trong task 2.2, ví dụ:

  *Dòng "**A0 0E 02 65 4E 6F A0 02 43 C8 03 D7 47 A9**" trong file "hex_packet_ee3491.dat" được chuyển đổi thành*

  2,2023:11:11 01:00:00,2,400.03,bright
- Viết mỗi dữ liệu đã chuyển đổi trên một dòng trong file đầu ra là file lux_condition.csv
- Ghi đè lên file đầu ra lux_condition.csv nếu file đã tồn tại.
- Có thể xử lý được tối thiểu 10000 điểm dữ liệu, tức là file đầu vào hex_packet_ee3491.dat có thể chứa 10000 dòng dữ liệu.

II. Các yêu cầu kỹ thuật khác:

Chương trình cần phải lưu các lỗi khi chạy chương trình (run-time errors) vào các file log, và phải có 3 file log khác nhau cho 3 tasks trong phần I, các file được đặt tên là task1.log, task2.log và task3.log tương ứng với task 1, task 2 và task 3. Mỗi lỗi xuất hiện cần được viết trên một dòng tương ứng với các file log với cấu trúc như sau:

Error AB: DESCRIPTION

Trong đó:

- AB là mã lỗi chứa 2 chữ số (nếu mã lỗi < 10 thì cần thêm số 0 phía trước, ví dụ: 01, 02, …)
- DESCRIPTION là phần mô tả cụ thể lỗi.

1. Các lỗi có thể xuất hiện trong task 1:
   - Câu lệnh command-line sai, ví dụ: câu lệnh thiếu 1 hoặc một vài tham biến. Thông báo lỗi có thể là "Error 01: invalid command".
   - Giá trị của các tham biến trong câu lệnh command-line không hợp lệ, ví dụ: số lượng cảm biến < 0. Thông báo lỗi có thể là "Error 02: invalid argument".
   - "lux_sensor.csv" đã tồn tại và là file chỉ đọc (read-only file), không cho phép ghi đè. Thông báo lỗi có thể là "Error 03: file access denied"

2. Các lỗi có thể xuất hiện trong task 2:
   - File đầu vào data_filename.csv hoặc location.csv không tồn tại hoặc không cho phép truy cập. Thông báo lỗi có thể là "Error 01: input file not found or not accessible"
   - File đầu vào data_filename.csv hoặc location.csv không có định dạng phù hợp, ví dụ không có dòng tiêu đề, số lượng dấu phẩy không đúng, hoặc thậm chí có định dạng hoàn toàn khác. Thông báo lỗi có thể là "Error 02: invalid input file format"
   - Câu lệnh command-line sai, ví dụ: không gõ tên một hoặc cả 2 file đầu vào. Thông báo lỗi có thể là "Error 03: invalid command"
   - File dữ liệu đầu vào chứa dữ liệu sai:
     - Tất cả các trường dữ liệu trên một dòng nào đó bị thiếu trong 1 hoặc cả 2 file, ví dụ: ", , " trong file data_filename.csv
     - Id thiếu hoặc không hợp lệ, ví dụ: "-1, 2023:11:11 00:00:00, 50.1"
     - Thời gian thiếu hoặc không hợp lệ, ví dụ: "1,2023:11:11 00:00:, 50.1"
     - Giá trị Lux thiếu, e.g.: "1,2023:11:11 00:00:00, "

     Thông báo lỗi cần phải kèm theo số dòng trong file đầu vào mà lỗi xuất hiện, ví dụ: "Error 04: invalid data at line X" trong đó X là số dòng của lỗi xuất hiện trong file dữ liệu. Dòng đầu tiên của file đầu vào là dòng tiêu đề được đánh số là 0, ví dụ trong file data_filename.csv thì dòng tiêu đề "id,time,value" là dòng thứ 0 (không), các dòng tiếp theo là dòng chứa dữ liệu được đánh số từ 1 (một).

   Trong quá trình xử lý dữ liệu khi thực hiện task 2 nếu gặp phải một dòng chứa dữ liệu lỗi thì chương trình sẽ bỏ qua dòng đó và xử lý dòng dữ liệu tiếp theo.
   - Nếu cảm biến có trong file dữ liệu không xuất hiện trong file location, ví dụ: file dữ liệu có 9 cảm biến với id từ 1 đến 9 nhưng trong file location chỉ có 7 cảm biến với id là 1, 2, 5, 6, 7, 8, 9 hoặc mã khu vực trong file `location.csv` thiếu ví dụ: "3,". Thông báo lỗi có thể là: "Error 05: unknown location of sensor ID" trong ID là số hiệu của cảm biến có trong file dữ liệu nhưng không có trong file location.

3. Các lỗi có thể xuất hiện trong task 3:
   - File đầu vào không tồn tại hoặc không cho phép truy cập. Thông báo lỗi có thể là "Error 01: input file not found or not accessible"

- File đầu vào không có định dạng phù hợp, ví dụ không có dòng tiêu đề, số lượng dấu phẩy không đúng, hoặc thậm chí có định dạng hoàn toàn khác. Thông báo lỗi có thể là "Error 02: invalid input file format"
- Câu lệnh command-line sai, ví dụ: không gõ tên một hoặc cả 2 file. Thông báo lỗi có thể là "Error 03: invalid command"
- File đầu ra đã tồn tại và không thể ghi đè. Thông báo lỗi có thể là: "Error 07: cannot override output file"
- Nếu file đầu vào là một file csv và chứa dữ liệu sai:
    - Tất cả các trường dữ liệu trên một dòng nào đó bị thiếu trong 1 hoặc cả 2 file, ví dụ: ", , " trong file data_filename.csv
    - Id thiếu hoặc không hợp lệ, ví dụ: "-1, 2023:11:11 00:00:00, 50.1"
    - Thời gian thiếu hoặc không hợp lệ, ví dụ: "1,2023:11:11 00:00:, 50.1"
    - Giá trị Lux thiếu, e.g.: "1,2023:11:11 00:00:00, "

  Thông báo lỗi cần phải kèm theo số dòng trong file đầu vào mà lỗi xuất hiện, ví dụ: "Error 04: invalid data at line X" trong đó X là số dòng của lỗi xuất hiện trong file dữ liệu. Dòng đầu tiên của file đầu vào là dòng tiêu đề được đánh số là 0, ví dụ trong file data_filename.csv thì dòng tiêu đề "id,time,value" là dòng thứ 0 (không), các dòng tiếp theo là dòng chứa dữ liệu được đánh số từ 1 (một).

  Trong quá trình xử lý dữ liệu khi thực hiện task 3 nếu gặp phải một dòng chứa dữ liệu lỗi thì chương trình sẽ bỏ qua dòng đó và xử lý dòng dữ liệu tiếp theo.
- Nếu file đầu vào là một file text (.data) và chứa gói tin sai định dạng hoặc sai dữ liệu:
    - Sai hoặc thiếu byte khởi đầu và byte kết thúc
    - Byte độ dài và độ dài thực của gói tin không thống nhất
    - Sai byte checksum (checksum error)
    - Thời gian là giá trị ở tương lai của thời điểm chạy chương trình và xử lý dữ liệu.

  Thông báo lỗi cần phải kèm theo số dòng trong file đầu vào mà lỗi xuất hiện, ví dụ: "Error 05: data packet error at line X" trong đó X là số dòng của lỗi xuất hiện trong file dữ liệu. Dòng đầu tiên của file ".dat" đầu vào là dòng số 1.

  Trong quá trình xử lý dữ liệu khi thực hiện task 3 nếu gặp phải một dòng chứa dữ liệu lỗi thì chương trình sẽ bỏ qua dòng đó và xử lý dòng dữ liệu tiếp theo.
- Nếu file đầu vào chứa dữ liệu lặp lại, ví dụ: 2 hoặc nhiều dòng trong file chứa dữ liệu giống hệt nhau, thông báo lỗi cần phải chứa các số dòng trong file đầu vào mà lỗi xuất hiện, ví dụ: "Error 06: data at line X and Y are duplicated" trong đó X là số dòng mà dữ liệu xuất hiện lần đầu tiên và Y là số dòng chứa dữ liệu lặp lại như dòng X.

  Trong quá trình xử lý dữ liệu khi thực hiện task 3 nếu gặp phải một dòng chứa dữ liệu lặp lại thì chương trình sẽ bỏ qua dòng đó và xử lý dòng dữ liệu tiếp theo.

4. Sinh viên có thể đề xuất các lỗi có thể xuất hiện khác với các lỗi ở trên. Các lỗi đó cũng cần phải lưu trong file log và mô tả trong báo cáo.

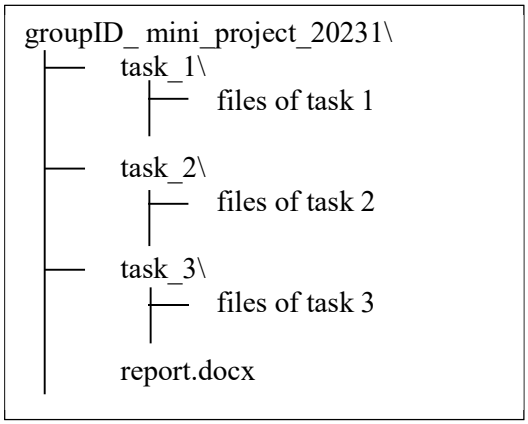
III. Thiết kế chương trình:

Sinh viên sử dụng các tiếp cận từ trên xuống (top-down approach) để thiết kế chương trình. Sinh viên cần vẽ được các sơ đồ top-down thể hiện mối quan hệ giữa các hàm trong chương trình cho mỗi task và mô tả ngắn gọn trong báo cáo.

Sinh viên cũng cần phải vẽ ít nhất 2 lưu đồ thuật toán:

- Một lưu đồ của toàn bộ chương trình (của 1 trong 3 tasks hoặc vẽ 3 lưu đồ cho cả 3 tasks)
- Và một lưu cho một hàm có vai trò quan trọng trong chương trình. Sinh viên có thể vẽ nhiều hơn 1 lưu đồ cho các hàm khác nhau.

Sinh viên cũng phải cung cấp cấu trúc thư mục và cấu trúc files như sau kèm theo mô tả ngắn gọn:



Trong đó thư mục gốc cần được đặt tên là "groupID _ mini_project_20231" và 3 thư mục con "task_1", "task_2" và "task_3" chứa các file liên quan. "report.docx" là file báo cáo và phải được đặt trong thư mục gốc của project. *groupID* trong tên thư mục cần được thay bằng ID của nhóm.

IV. Phong cách lập trình (Coding styles):

Coding style cần nhất quán trong toàn bộ chương trình và tuân theo GNU style được mô tả trong link sau: https://www.gnu.org/prep/standards/html_node/Writing-C.html
Một cách ngắn gọn:

- Cấu trúc mã chương trình phải sạch sẽ và dễ đọc bằng cách sử dụng lùi đầu dòng, ngoặc, chia đoạn, xuống dòng và cách dòng hợp lý.
- Phải có chú thích để giải thích chương trình rõ ràng hơn nhưng chứ không nên chỉ chú thích để diễn xuôi lại câu lệnh.
- Tên hàm và tên biến nên đặt bằng tiếng Anh, súc tích và có tính tự mô tả.
- Tránh hard-coding.

Chú ý: Sinh viên KHÔNG sử dụng các thư viện bên ngoài mà không phải là thư viện C/C++ chuẩn.

V. Công cụ:

Soạn thảo: Visual studio code (https://code.visualstudio.com/download)
Biên dịch: gcc or g++ in MinGW-w64 (https://sourceforge.net/projects/mingw-w64/) version 8.1.0
Sinh viên cũng cần phải mô tả trong báo cáo chương trình được viết và chạy thử trên hệ điều hành nào (Windows, Linux, MacOS).

VI. Hướng dẫn báo cáo và nộp bài:

- Sinh viên làm bài tập lớn (project) theo nhóm.
- Toàn bộ project cần được tổ chức thành thư mục như đã mô tả trong phần III.
- Sinh viên viết báo cáo bằng tiếng Việt hoặc tiếng Anh dùng file Word và đặt tên là "report.docx", báo cáo không quá 4 trang A4 và không nên chứa mã chương trình. Báo cáo cần phải viết theo định dạng IEEE với mẫu báo cáo được đính kèm trong Team Assignment.
- Nội dung của báo cáo bao gồm:
  - Giới thiệu: Mô tả ngắn gọn ý tưởng thiết kế chương trình bao gồm
    - Sơ đồ top-down,
    - Cấu trúc thư mục,
    - Các file mã chương trình (nếu có nhiều file),
    - Và các thư viện chuẩn được sử dụng.

    Lưu ý: KHÔNG chép lại đề bài trong báo cáo.
  - Thiết kế chi tiết:

        - Giới thiệu các cấu trúc dữ liệu quan trọng mà nhóm sinh viên tự định nghĩa để xử lý dữ liệu (nếu có),
        - Mô tả thiết kế của một vài hàm quan trọng (tự lựa chọn) bao gồm lời gọi hàm (tên hàm, danh sách tham biến, và kiểu trả về) và đầu vào/ ra, điều kiện đầu, điều kiện sau.;
        - Ít nhất 2 lưu đồ thuật toán như đã mô tả trong phần III.
    - Kết quả và đánh giá: tổng hợp kết quả chạy chương trình và đánh giá chất lượng.
    - Kết luận:
        - Kết luận ngắn gọn những điểm đã làm được và chưa làm được trong bài tập lớn;
        - Cung cấp bảng chứa thông tin đóng góp của mỗi thành viên nhóm như dưới đây::

Họ và tên sinh viên	Tasks	Phần trăm đóng góp
Nguyen Van A	1, 2.1, 2.2	50%
Nguyen Van B	2.3, 2.4, 3	50%

      Nếu chỉ một trong 2 sinh viên của nhóm thực hiện toàn bộ bài tập lớn và người còn lại không làm gì, thì người thực hiện có thể viết trong báo cáo là đóng góp 100% và người còn lại 0%.
    - Tài liệu tham khảo (Nếu có).
- Sinh viên cần nén toàn bộ thư mục bài tập lớn trong 1 file **zip** và đặt tên là "groupId_mini_project_20231.zip" để nộp. Chú ý phải là file **ZIP** chứ không được sử dụng định dạng file nén nào khác như ".rar".
    - Lưu toàn bộ các file mã nguồn, file chạy chương trình và file Word báo cáo.
    - Xóa toàn bộ các file nháp không liên quan đến phần thực hiện bài tập lớn trước khi nộp.
    - "groupID" trong tên file cần được thay thế bởi ID của nhóm (group ID), ví dụ: "20221234_20222345_mini_project_20231.zip"
- Sinh viên nộp bài tập lớn trên Team Assignment trước thời hạn đã quy định trên Teams. KHÔNG nộp bài qua email, Teams chat hoặc bất kỳ kênh nào khác. Mỗi nhóm chỉ cần nộp một bài.
- Nếu có bất kỳ thắc mắc gì về đề bài, sinh viên có thể liên hệ với giảng viên để làm rõ. Tuy nhiên sinh viên nên đặt câu hỏi trong Teams lớp chứ không nên liên hệ riêng để tất cả sinh viên khác trong lớp cũng có thể theo dõi.

VII. Đánh giá:

- Phần thực hiện bài tập lớn được đánh giá như sau:
    - Tất cả các tasks được thực hiện hoàn chỉnh, không lỗi khi chạy chương trình, xử lý lỗi và thực hiện một cách sáng tạo (60%)
    - Thiết kế tốt và dễ sử dụng lại (20%)
    - Coding style tốt (20%)
    - Báo cáo có bố cục rõ ràng, sạch đẹp theo đúng dịnh dạng của file mẫu và nhất quán với phần mã nguồn (20%)
    - Nếu đặt tên file, thư mục và cấu trúc thư mục không như yêu cầu trong phần III (-5%)
- Nhóm sinh viên phải tự thực hiện bài tập lớn. KHÔNG sao chép của người/nhóm khác. Mỗi nhóm cần giữ bí mật bài làm của mình. Nếu các nhóm có phần mã nguồn và/hoặc báo cáo giống nhau, thì phần bài làm của tất cả các nhóm đó sẽ không được chấp nhận và coi như không nộp bài bất kể ai sao chép của ai.
- Không được phép nộp muộn.
- Bài làm tốt sẽ được cộng vào điểm quá trình.
- Nếu không nộp bài thì sinh viên sẽ bị trừ 3 điểm (trên 10 điểm) trong điểm quá trình.

---------- HẾT ----------