1. **Giới thiệu chung**
2. **Sơ đồ top – down** **b. Cấu trúc thư mục**





1. **Các thư viện và mã chương trình**
2. Task1

* Task 1 sử dụng các hàm thuộc thư viện chuẩn C++ :<iostream>; <cstring>; <ctime>.

1. Task2

* Task 2 chia ra thành nhiều task nhỏ “(T2.1)check\_valid.cpp”; “(T2.2)check\_con.cpp”; “(T2.3)lux\_summary.cpp”.
* Hàm main của task 2 được đặt trong lux\_process.cpp.
* location\_min\_max.csv được sử dụng để lưu trữ các dữ liệu minmax ứng với từng location, nên phải đi cùng thư mục.
* Task 2 sử dụng các hàm thuộc thư viện chuẩn C++ :<iostream>; <cstring>; <ctime>; <map>.

1. Task3

* Task 3 sử dụng các hàm thuộc thư viện chuẩn C++ :<iostream>; <cstring>; <ctime>; <map>.
* Ngoài ra task3 có sử dụng “(T2.2)check\_con.cpp”, đi kèm theo đó là file dữ liệu minmax “location\_min\_max.csv”.

1. **Thiết kế chi tiết**
2. **Task 1**
3. Sơ đồ top – down



1. Một số cấu trúc và hàm quan trọng

Create\_Simulate\_Light\_Sensor : Tạo các cảm biến ánh sáng phù hợp yêu cầu đề bài

* Nhận đầu vào là các tham số (Đã được kiểm tra phù hợp): số lượng sensor [-n]; Thời gian trích mẫu [-st]; Thời gian giả lập [-si].
* Trả đầu ra là file lux\_sensor
* Hàm này lấy ý tưởng từ việc tạo một biến lưu trữ thời gian bằng các bội liên tiếp của thời gian giả lập [-si] qua mỗi vòng lặp :[Time\_Period]. Nếu [Time\_period] < [-si] thì tiếp tục vòng lặp với thời gian giả lập bằng thời điểm trích mẫu lần đầu [Start\_time] + [Time\_period]

1. **Task 2**
2. Sơ đồ top – down

Task2.1 Task2.2 Task2.3

  

1. Một số cấu trúc và hàm quan trọng
2. SensorData21; SensorData23: lưu trữ lần lượt id,time,value của cảm biến đọc từ file đầu vào.
3. SensorData: Lưu trữ các thông số cảm biến trong khi chạy task 2.2 lần lượt là id, time, location, value, condition.
4. SensorHourData: Lưu trữ tổng giá trị của cảm biến trong một khung thời gian và số lần mô phỏng trong khung thời gian đấy.
5. SummarySensorData: Lưu trữ thông số cảm biến trong khi chạy task 2.3 bao gồm giá trị min và max trong thời gian chạy và thời gian tương ứng với nó, giá trị trung bình trong thời gian chạy, tổng thời gian chạy tính bằng giờ và số lần mô phỏng trong toàn bộ thời gian.
6. Hàm determineCondition: Hàm nhận đầu vào là một tham chiếu của SensorData đã được xác định hết các giá trị khác trừ condition, và một map có key là idlocation value là min và max value của địa điểm đó

* Đây là hàm quan trọng vì sẽ được sử dụng trong cả task2.2 và task3
* Hàm này sẽ đi kèm với hàm đọc dữ liệu từ file location\_min\_max.csv để trả ra là một key như đầu vào của hàm
* Lưu ý hàm này chỉ đánh giá tình trạng cảm biến thông qua các giá trị đầu vào chứ không in ra thông báo lỗi thiếu location của sensor, nhiệm vụ này ở trong lux\_process.

1. Hàm Check\_Condition\_Sensor: Tính toán giá trị trung bình của cảm biến trong một giờ và trả về một file csv đã nói từ trước

* Đọc dữ liệu từ file đầu vào và tạo một map có :

+ key : một map con với key là khung thời gian (ví dụ 13:00:00 – 13:59:59 nằm trong khung thời gian 14:00:00) và giá trị là id của cảm biến.

+ value là một struct SensorHourData.

* Tìm vị trí của cảm biến đã được xác đinh trong một map từ trước sau đấy gán vào sensor.location.
* Đánh giá tình trạng cảm biến bằng hàm determineCondition đã khai báo từ trước.

1. Hàm Create\_Summary\_Sensor nhận đầu vào là file dữ liệu trước khi qua xử lý trong T2.1(lux\_sensor.csv) và sau khi qua xử lý trong task 2.1 (lux\_inliner.csv) trả về một file csv chứa dữ liệu tổng quát về một sensor trong toàn bộ thời gian chạy

* Với ý tưởng là trong toàn bộ thời gian chạy, một cảm biến sẽ chỉ có một giá trị min, max và trung bình, hàm này đã tạo ra một map với key là id cảm biến và value là struct SummarySensorData.
* Do sau khi chạy task 2.1, một số dữ liệu cảm biến đã bị mất nên hàm này nhận đầu vào là file dữ liệu tiền xử lý 2.1 để có thể tính chính xác toàn bộ thời gian mô phỏng cho từng sensor.

1. **Task 3**
2. Sơ đồ top – down



1. Một số cấu trúc và hàm quan trọng

Communicationpacket là một cấu trúc dung để lưu trữ trông tin sensor dưới dạng Hex, chi tiết đã được chú thích trong mã nguồn.

1. **Kết luận và đánh giá**
2. **Đánh giá**

Sau khi chạy một số file test đơn giản đã cho lại một kết quả đầu ra như mong muốn và tương đốn ổn định, tuy nhiên vẫn có một số lỗi khác chương trình chưa thể phát hiện một cách rõ rang.

1. **Kết luận**

Đây là kết quả từ bài làm cá nhân, vậy nên 100% đóng góp là của sinh viên Nguyễn Trung Kiên – MSSV : 20210500

1. **Tài liệu tham khảo**

<https://www.ietf.org/rfc/rfc4180.txt>

<https://www.gnu.org/prep/standards/html_node/Writing-C.html>

<https://www.ieee.org/>