Báo cáo bài tập lớn 1

Sinh viên: Đỗ Thu Giang   
MSSV: 20200182  
Lớp: CTTN KTĐK – TĐH K65Mã lớp: 136089

*Tóm tắt*—Báo cáo trình bày lại các nội dung chính để thiết kế các chương trình phần mềm xử lý dữ liệu cảm biến. Các kết quả và đánh giá cũng được trình bày ở cuối báo cáo.

# GIỚI THIỆU CHUNG

Bài tập lớn đưa ra yêu cầu thiết kế chương trình phần mềm với các hàm và cấu trúc dữ liệu thích hợp để xử lý dữ liệu cảm biến, được chia thành 3 nhiệm vụ chính: (1) cho phép người dùng cung cấp thông tin về cảm biến và đưa ra các giá trị liên quan đến đo mô phỏng; (2) xử lý dữ liệu từ chương trình đầu; (3) chuyển các dữ liệu đã qua xử lý thành một bản tin truyền thông để gửi đi.

Để thực hiện các nhiệm vụ trên, ba chương trình con được tạo ra theo các yêu cầu cụ thể. Công cụ lập trình sử dụng bao gồm: Visual studio code (Editor) và gcc in MinGW-w64 (Compiler). Các chương trình được viết trong hệ điều hành Windows.

Các kiến thức được sử dụng trong bài tập lớn bao gồm các kiến thức được học trên lớp và các kiến thức tìm hiểu thêm như lập trình hướng đối tượng để giúp việc lập trình dễ dàng hơn.

Nội dung chính của báo cáo gồm các phần: (1) Giới thiệu chung; (2) Ý tưởng chính: mô tả ý tưởng thiết kế chương trình; (3) Thiết kế chi tiết: mô tả thiết kế các hàm quan trọng; (4) Kết quả và đánh giá: mô tả kết quả chạy và đánh giá chất lượng chương trình; (5) Kết luận.

# Ý TƯỞNG CHÍNH

## Mô tả chung ý tưởng thiết kế chương trình

### Chương trình 1:

#### Input: num\_sensors (số lượng cảm biến), sampling (thời gian trích mẫu) và interval (khoảng thời gian đo)

#### Output: tập dữ liệu theo cấu trúc “định danh cảm biến (id), thời điểm đo (time), giá trị cảm biến (values)”

#### Ý tưởng chung: cần (i) hàm để chuyển đổi kiểu dữ liệu nhập thành kiểu dữ liệu tính toán được, (ii) hàm tạo số ngẫu nhiên để tạo giá trị cảm biến, (iii) hàm lấy thời gian thực, (iv) các hàm để tạo, ghi file và (v) hàm để ghi file theo từng khoảng thời gian.

### Chương trình 2:

#### Input: file dữ liệu có định dạng như “dust\_sensor.csv”

#### Output: (i) **Task 2.1:** file dữ liệu lưu các điểm dữ liệu nằm ngoài giá trị đo theo định dạng “định danh cảm biến (id), thời điểm đo (time), giá trị cảm biến (values)”; (ii) **Task 2.2:** file dữ liệu bao gồm các dữ liệu đã qua tính toán theo định dạng “định danh cảm biến (id), thời gian (time), giá trị trung bình theo từng giờ (values), chỉ số AQI tương ứng (aqi), cấp độ ô nhiễm (pollution)”; (iii) **Task 2.3:** file dữ liệu bao gồm các dữ liệu đã qua tính toán theo định dạng “định danh cảm biến (id), thông số (parametters), thời gian các giá trị này xuất hiện trong file (time), giá trị (values)”, (iv) **Task 2.4:** file thống kê tổng số giờ ở mỗi cấp độ ô nhiễm đo được ở các node cảm biến.

#### Ý tưởng chung:

Task 2.1**:** (i) chuyển đổi và phân loại dữ liệu lưu trong file “dust\_sensor.csv” bằng cách định nghĩa một class mang thông tin về giá trị (values), định danh (indice) và thời gian (times), có hàm thành viên cho phép thêm, xử lý và ghi dữ liệu; (ii) hàm kiểm tra giá trị có nằm trong khoảng cho phép không và lưu các vị trí có giá trị dị biệt vào một vector; (iv) sử dụng vector để xác định vị trí ghi các class (i) vào file “dust\_ouliers.csv”

Task 2.2: (i) định nghĩa một class có các hàm thành viên cho phép tính toán AQI và độ ô nhiễm ; (ii) mảng vector với mỗi vector đại diện cho từng cảm biến có tác dụng để lưu các vector có kiểu class định nghĩa ở (1); (iii) ghi lần lượt vào file “dust\_aqi.csv”

Task 2.3: (i) định nghĩa một class có các hàm thành viên cho phép tính toán các thông số (giá trị lớn nhất, giá trị nhỏ nhất, giá trị trung bình) và xuất dữ liệu; (ii) mảng có kiểu dữ liệu định nghĩa từ class trên và ghi lần lượt vào file

Task 2.4: (i) định nghĩa một class có các hàm thành viên cho phép tính toán AQI và độ ô nhiễm ; (ii) mảng vector đại diện cho từng cảm biến có tác dụng để lưu các vector có kiểu class định nghĩa ở (1); (iii) ghi lần lượt vào file “dust\_aqi.csv”

### Chương trình 3:

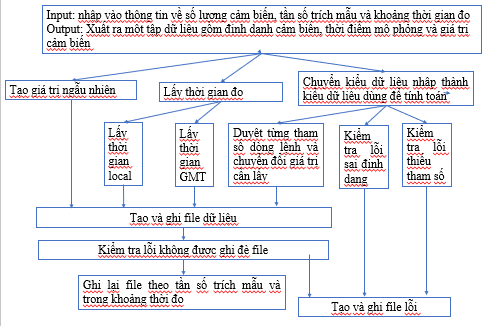
#### Input: file dữ liệu có định dạng như task 2.2 “dust\_aqi.csv”

#### Output: file hex gồm một chuỗi byte mang các thông tin cho trước

#### Ý tưởng chung: cần (i) hàm để chuyển đổi thời gian về dạng tính toán được, (ii) hàm để đọc và lưu từng dùng file, (iii) hàm chuyển đổi từ các dữ liệu đã lưu thành các mã hexa; (iv) hàm để lưu các mã hexa vào file “hex\_filename.dat”

## Sơ đồ top – down approach

### Chương trình 1:



1. Sơ đồ top – down approach chương trình 1

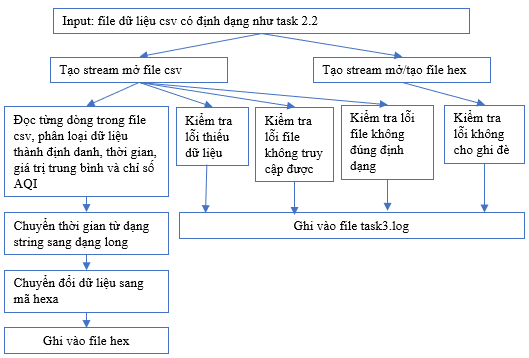
### Chương trình 2:

Ảnh có chứa bàn

Mô tả được tạo tự động

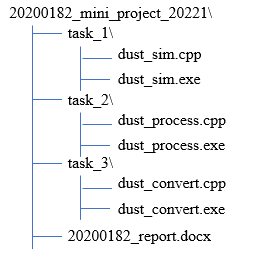
1. Sơ đồ top – down approach chương trình 2

### Chương trình 3:



1. Sơ đồ top – down approach chương trình 3

## Cấu trúc thư mục



1. Cấu trúc thư mục

## Các thư viện được sử dụng:

1. CÁC THƯ VIỆN ĐƯỢC SỬ DỤNG

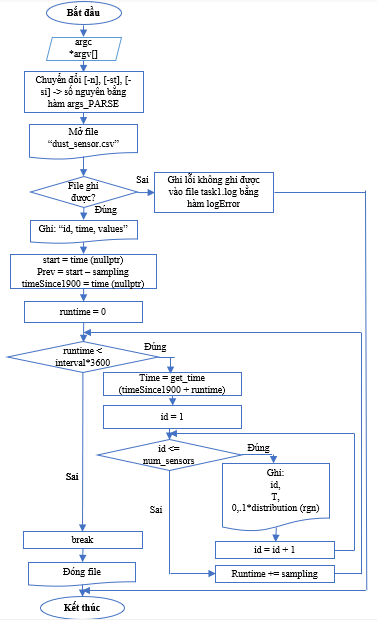
|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Tên thư viện** | **Chương trình sử dụng** | **Mục đích** |
| 1 | fstream | dust\_sim.cpp  dust\_process.cpp  dust\_convert.cpp | Tạo, đọc, ghi dữ liệu cho file |
| 2 | iomanip | dust\_sim.cpp  dust\_process.cpp  dust\_convert.cpp | Định dạng dữ liệu nhập xuất cho file văn bản |
| 3 | cstring | dust\_sim.cpp  dust\_process.cpp  dust\_convert.cpp | Thao tác với chuỗi |
| 4 | random | dust\_sim.cpp | Tạo hàm random |
| 6 | algorithm | dust\_process.cpp | Thao tác với |
| 7 | vector | dust\_process.cpp | Thao tác với vector |
| 8 | map | dust\_process.cpp | Thao tác với map |
| 9 | sstream | dust\_convert.cpp | thao tác với string như một stream |
| 10 | numeric | dust\_convert.cpp |  |

# Thiết kế chi tiết

## Chương trình 1

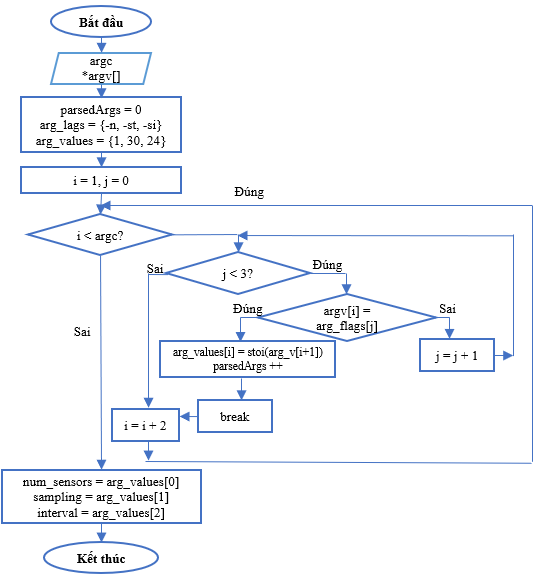
### Lưu đồ thuật toán:

#### Luồng chương trình chính:



1. Lưu đồ thuật toán hàm main chương trình 1

#### Hàm args\_parse:



1. Lưu đồ thuật toán hàm args\_parse chương trình 1

### Mô tả thiết kế hàm:

#### Hàm logError:

* Mô tả: tạo, mở và ghi file task1.log; dừng chương trình khi gặp lỗi.
* Kiểu trả về: void
* Tham biến: *msg* (string): lỗi được ghi trong file
* Đầu vào: *msg* (string): lỗi được ghi trong file
* Đầu ra:

#### Hàm args\_parse

* Mô tả: chuyển đổi kiểu định dạng của các tham số [num\_sensors], [sampling] và [interval] từ kiểu string về kiểu int, check lỗi định dạng và lỗi thiếu tham số.
* Kiểu trả về: void
* Tham biến: *arg\_flags* (string): lưu các tham số "-n", "-st", "-si"; *arg\_values* (int): lưu các giá trị của [num\_sensors], [sampling] và [interval]; *parsedArgs* (int): đếm số tham số đã chuyển đổi
* Đầu vào: *argc:* số lượng tham số truyền từ command-line và *argv[]*: mảng con trỏ hướng đến bất kì tham số nào.
* Đầu ra:

#### Hàm get\_time

* Mô tả: lựa chọn lấy thời gian theo thời gian local hoặc thời gian GMT và đưa về thời gian theo định dạng sẵn
* Kiểu trả về: string
* Tham biến: *t* (auto): biểu diễn thời gian hiện tại; *tm* (auto): chuyển đổi format cho *t;*
* Đầu vào: *local* (bool): thời gian local hoặc GMT
* Đầu ra: *str* (char): thời gian theo định dạng chuẩn

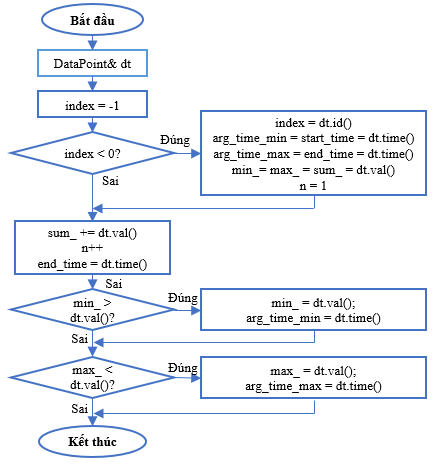
#### Hàm main:

* Mô tả: ghi dữ liệu vào file “dust\_simp.csv”, check lỗi không cho ghi đè
* Tham biến: *start* (long): thời gian bắt đầu đo; *prev* (long): thời gian của lần đo trước; *cur* (long): thời gian hiện tại; *T* (string)*:* thời gian được viết theo cấu trúc;
* Đầu vào: *argc:* số lượng tham số truyền từ command-line và *argv[]*: mảng con trỏ hướng đến bất kì tham số nào.

## Chương trình 2:

### Lưu đồ thuật toán

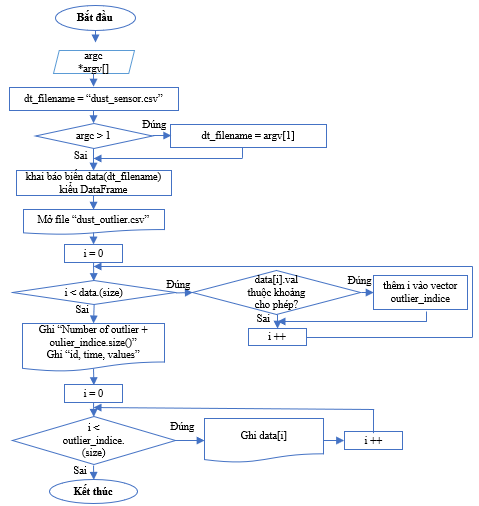
#### Hàm add() của class DataSummary:



1. Lưu đồ thuật toán hàm add chương trình 2

#### Hàm main

*Task 2.1*



1. Lưu đồ thuật toán hàm main chương trình 2

### Mô tả thiết kế hàm và các lớp:

#### Hàm logError: tương tự trên

#### Hàm get\_epoch\_time:

* Mô tả: chuyển đổi thời gian từ kiểu string sang kiểu long(time\_t)
* Kiểu trả về: long
* Tham biến: *t* (struct tm): cấu trúc thời gian {year, mon, mday, hour, min, sec}; *timeSince1900* (time\_t): thời gian tính từ năm 1900
* Đầu vào: *T* (string): thời gian kiểu string
* Đầu ra: *timeSince1900* (long): thời gian từ năm 1900

#### Hàm get\_std\_time:

* Mô tả: chuyển đổi thời gian từ kiểu long sang kiểu string
* Kiểu trả về: long
* Tham biến: *t* (time\_t): thời gian viết theo kiểu time\_t, *str[80]* (string): thời gian viết theo kiểu string
* Đầu vào: *epochtime* (long): thời gian kiểu sang kiểu long(time\_t)
* Đầu ra: *str* (string): thời gian viết theo kiểu string.

#### Class DataPoint:

* Mô tả: cấu trúc mô tả dữ liệu bao gồm định danh, thời gian và giá trị của cảm biến
* Biến thành viên (private): *id\_* (int), *time\_* (string), *val\_* (float)
* Hàm khởi tạo tham số: *id\_*= id; *time\_* = time; *val\_* = val
* Hàm thành viên: nạp chồng toán tử xuất (<<): xuất theo cấu trúc “dt.id,dt.time,dt.val”; hàm trả về giá trị của *id\_;* hàm trả về giá trị của ­*val\_*; hàm trả về giá trị của *get\_epoch\_time(time\_)*

#### Class DataFrame:

* Mô tả: cấu trúc dữ liệu giúp lấy dữ liệu từ file csv rồi chuyển đổi và phân loại thành các vector.
* Biến thành viên (private): *indice\_* (vector<int>), *times\_* (vector<string>), *values\_* (vector<float>)
* Hàm khởi tạo tham số: Tham số truyền vào: *dt\_filename* (string); kiểm tra chất lượng file; đọc file theo từng dòng và lưu dữ liệu vào các biến thành viên.
* Hàm thành viên: nạp chồng toán tử ([]); hàm *num\_sensors()*: trả về vector gồm chuỗi các giá trị định danh của cảm biến theo thứ tự tăng dần; hàm *size*: trả về số lần đo của cảm biến.

#### Class DataSummary:

* Mô tả: cấu trúc dữ liệu gồm các hàm tính toán giá trị lớn nhất (max), giá trị nhỏ nhất (min) và giá trị nồng độ bụi trung bình (mean) cũng như thời gian xuất hiện của chúng.
* Biến thành viên (protected): *arg\_time\_min* (long), *arg\_time\_max* (long), *start\_time* (long), *end\_time* (long), *min\_* (float), *max\_* (float), *sum\_* (float), *n* (int), *index = -1* (int).
* Hàm thành viên: hàm *add(const DataPoint& dt)*: tính toán các giá trị max, min và mean; nạp chồng toán tử xuất (<<): xuất theo định dạng như yêu cầu

#### Class HourAverageMeter:

* Mô tả: cấu trúc dữ liệu gồm các hàm tính toán giá trị aqi và mức độ ô nhiễm
* Biến thành viên (private): *start\_time* (long), *sum* (float), *n* (int), *index = -1* (int).
* Hàm thành viên: hàm *add(const DataPoint& dt)*: tính tổng giá trị cảm biến, đếm số lần đo và kiểm tra xem giá trị được thêm vào hay chưa; hàm *aqi()*: tính toán chỉ số AQI; hàm *pollution():* kiểm tra mức độ ô nhiễm; nạp chồng toán tử xuất (<<): xuất theo định dạng như yêu cầu

#### Hàm is\_outlier:

* Mô tả: kiểm tra xem x có nằm ngoài giá trị cho phép không
* Kiểu trả về: bool
* Đầu vào: *x* (float): giá trị cần kiểm tra
* Đầu ra: *true/false*

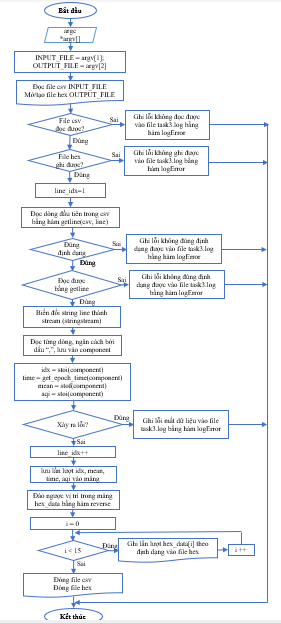
#### Hàm main:

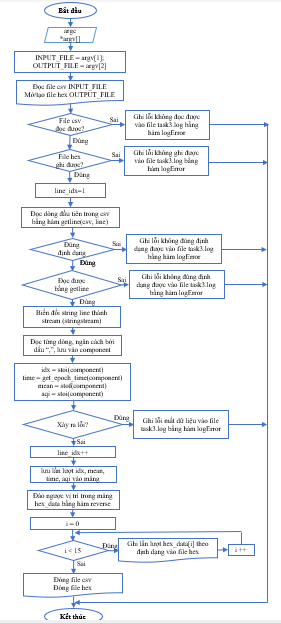
* Mô tả: ghi dữ liệu vào các file
* Đầu vào: *argc:* số lượng tham số truyền từ command-line và *argv[]*: mảng con trỏ hướng đến bất kì tham số nào.

## Chương trình 3:

### Lưu đồ thuật toán:

#### Luồng chương trình chính:





1. Lưu đồ thuật toán hàm main chương trình 2

### Mô tả thiết kế hàm:

#### Hàm logError: tương tự trên

#### Hàm get\_epoch\_time: tương tự trên

#### Hàm main:

* Mô tả: đọc từng dòng từ file dữ liệu, chuyển đổi dữ liệu sang mã hex, lưu vào file dat
* Đầu vào: *argc:* số lượng tham số truyền từ command-line và *argv[]*: mảng con trỏ hướng đến bất kì tham số nào.

# KẾT QUẢ VÀ ĐÁNH GIÁ

## Kết quả chạy

### Chương trình 1:

1. KẾT QUẢ CHẠY CHƯƠNG TRÌNH 1

|  |  |  |
| --- | --- | --- |
| **TT** | **Test case** | **Kết quả** |
| 1 | .\dust\_sim.exe -n 3 -st 30 -si 24 | Chạy đúng yêu cầu |
| 2 | Thiếu 1 trong các tham số vào | Chạy đúng yêu cầu |
| 3 | .\dust\_sim.exe -n 3 -st 30 -si -3 | Chạy đúng yêu cầu, báo lỗi trong file task1.log |
| 4 | .\dust\_sim.exe -n 3 -st 30 -si hai | Báo lỗi nhưng không ghi lỗi vào file task1.log |
| 5 | Chạy nhưng mở file csv trong ứng dụng khác | Báo lỗi không mở được file vào task1.log |

### Chương trình 2:

1. KẾT QUẢ CHẠY CHƯƠNG TRÌNH 2

|  |  |  |
| --- | --- | --- |
| **TT** | **Test case** | **Kết quả** |
| 1 | .\dust\_process.exe dust\_sensor.csv | Chạy đúng yêu cầu |
| 2 | Chạy nhưng mở file csv trong ứng dụng khác | Báo lỗi không mở được file vào task2.log |
| 3 | Xóa dữ liệu trong file csv | Báo lỗi thiếu dữ liệu trong task2.log |
| 4 | Xóa dòng đầu trong file csv | Báo lỗi sai định dạng trong task2.log |

### Chương trình 2:

1. KẾT QUẢ CHẠY CHƯƠNG TRÌNH 3

|  |  |  |
| --- | --- | --- |
| **TT** | **Test case** | **Kết quả** |
| 1 | .\dust\_process.exe dust\_sensor.csv hex\_packet.dat | Chạy đúng yêu cầu |
| 2 | Chạy nhưng mở file csv trong ứng dụng khác | Báo lỗi không mở được file vào task2.log |
| 3 | Xóa dữ liệu trong file csv | Báo lỗi thiếu dữ liệu trong task2.log |
| 4 | Xóa dòng đầu trong file csv | Báo lỗi sai định dạng trong task2.log |

## Đánh giá chất lượng chương trình

### Độ tin cậy:

Chương trình có khả năng ghi lại các lỗi theo yêu cầu đề bài, nhưng vẫn còn một số lỗi chưa ghi lại được. Các lỗi được ghi lại chưa có phương án tự động sửa mà dừng chương trình.

### Khả năng kiểm tra và khả năng dễ hiểu:

Chương trình được viết phân chia thành các hàm và có chú thích. Các tên biến và tên hàm được đặt dễ hiểu và có ý nghĩa. Tuy nhiên, luồng chương trình chính còn dài, cần được tổ chức thành các hàm nhỏ hơn để dễ kiểm tra hơn.

### Khả năng tái sử dụng:

Chương trình có khả năng tái sử dụng các hàm (hàm thời gian, hàm duyệt tham số,..).

### Thời gian chạy:

Thời gian chạy các chương trình trong mức cho phép.

# Kết luận

Chương trình hoàn thành cơ bản các nhiệm vụ theo yêu cầu đề bài, thời gian chạy và các đánh giá chất lượng chương trình ở mức tốt. Tuy nhiên vẫn có thể còn nhiều lỗi chưa được phát hiện do các test case còn hạn chế.