BÁO CÁO KỸ THUẬT LẬP TRÌNH

BÀI TẬP LỚN SỐ 02

Sinh viên: Nguyễn Việt Thành
Mã số sinh viên: 20202524
Mã lớp: 136089
Giảng viên HD: TS. Hoàng Đức Chính

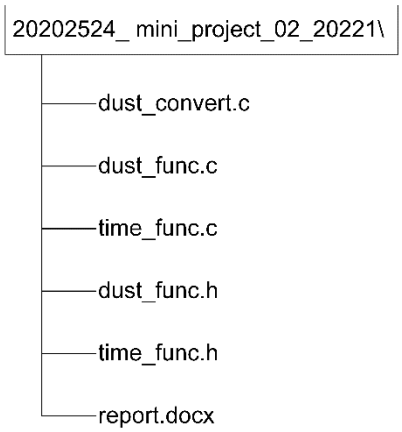
***Tóm tắt* – Báo cáo bài tập lớn số 2 môn Kỹ thuật lập trình với đề tài "Thiết kế chương trình phần mềm với các hàm và cấu trúc dữ liệu thích hợp để xử lý dữ liệu cảm biến.". Chương trình sử dụng ngôn ngữ C trên hệ điều hành Windows để xử lí dữ liệu mô phỏng cảm biến PM2.5 đo nồng độ hạt bụi có trong không khí.**

I. Ý TƯỞNG THIẾT KẾ

Yêu cầu của bài tập lớn cần thiết kế chương trình có thể thực hiện các chế độ sau:

- Đọc dữ liệu cảm biến dạng text và chuyển sang dạng chuỗi số hexa.
- Đọc dữ liệu cảm biến dạng chuỗi số hexa và chuyển sang dạng text.
- Sắp xếp dữ liệu theo yêu cầu khi chuyển từ dạng hexa sang dạng text.

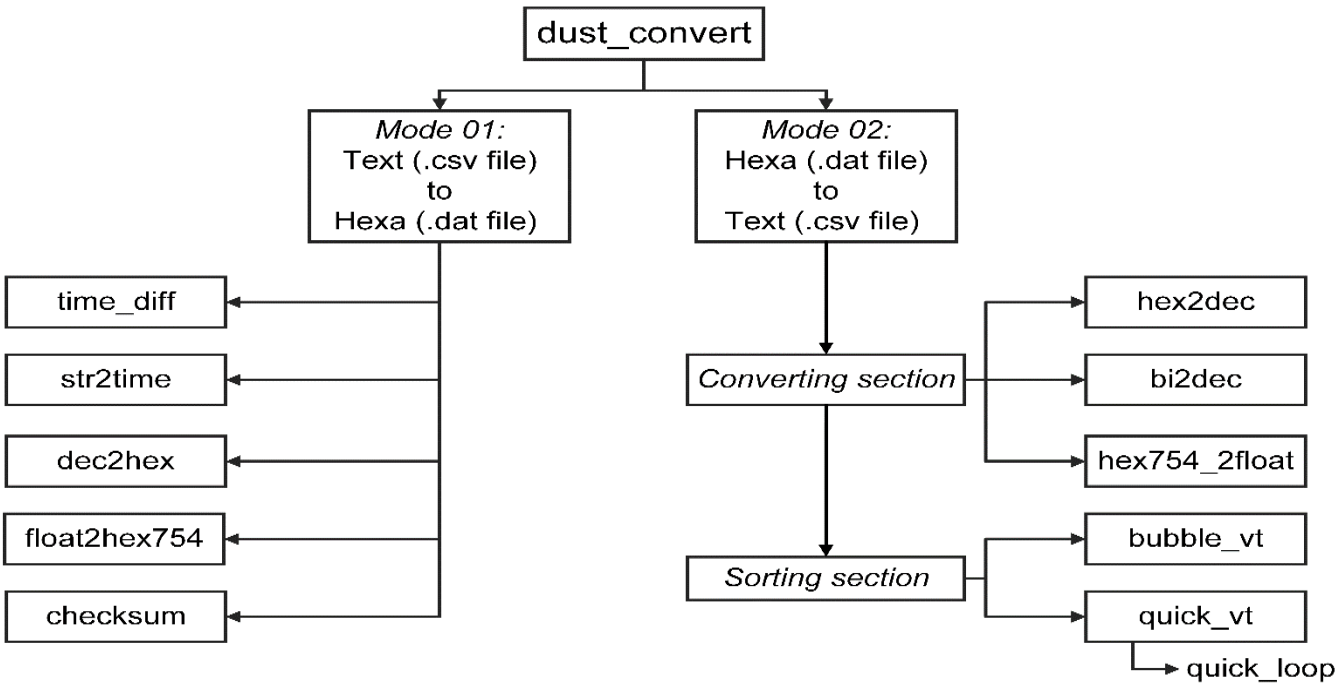
Do nội dung của các nhiệm vụ có mối liên hệ với nhau, cách tiếp cận từ trên xuống được lựa chọn để xây dựng chương trình. Hình I.1 mô tả sơ đồ phương pháp tiếp cận từ trên xuống được sử dụng. Chương trình được thiết kế để thực hiện 2 nhiệm vụ như yêu cầu. Chế độ tạo lập bản tin truyền thông (*Mode 01*) gồm 5 hàm chính: *time_diff*, *str2time*, *dec2hex*, *float2hex754*, và *checksum*. Đối với chế độ biên dịch bản tin truyền thông (*Mode 02)*, sử dụng 3 hàm:



Hình I.2: Cấu trúc thư mục chương trình

hex2dec, *bi2dec*, *hex754_2float* để chuyển đổi từ dạng chuỗi số hexa sang dạng text, và 2 hàm: *bubble_vt*, *quick_vt* để sắp xếp dữ liệu chuyển đổi.

Cấu trúc thư mục của toàn bộ chương trình được thể hiện ở hình I.2, gồm chương trình chính *dust_convert.c*, các hàm quan trọng và các hàm phụ trợ được thiết lập thành 2 thư viện: *time_func.h* và *dust_func.h*. Ngoài ra, sử dụng thêm các thư viện chuẩn của C: *stdio.h*, *stdlib.h*, *string.h*, *time.h*, *và winsock.h* để hỗ trợ cho việc thiết kế chương trình.



Hình I.1: Sơ đồ tiếp cận từ trên xuống

II. THIẾT KẾ CHI TIẾT

A. Chế độ tạo lập bản tin truyền thông (Mode 01)

Xuất phát từ một file csv định dạng cho trước, cần tạo lập được bản tin truyền thông. Các nhiệm vụ cần thực hiện:

- ID: chuyển từ số thập phân sang hệ 16 (1 byte)
- Time: chuyển từ số thập phân sang hệ 16 (4 bytes)
- Giá trị đo: chuyển từ số thực sang hexadecimal chuẩn IEEE 754 (4 bytes)
- AQI: chuyển từ số thập phân sang hệ 16 (2 byte)
- Checksum: tính toán với các byte hệ 16 (1 byte)

Trước khi thực hiện việc chuyển đổi, cần kiểm tra tính đúng đắn của dữ liệu đọc vào gồm lọc bỏ các dữ liệu bị lỗi cú pháp và các dữ liệu bị trùng lặp. Khi đã có được tập dữ liệu hợp lệ, em xây dựng các hàm chính ở *Mode 01*:

1. dec2hex

Cú pháp: void dec2hex (int decimalNumber, int bytes_num, char *hex_bytes_display);

Đầu vào: số nguyên và độ dài byte cần biểu diễn

Đầu ra: chuỗi biểu diễn bằng số hexa với độ dài yêu cầu

Lưu đồ thuật toán của hàm được thể hiện ở hình II.1

2. float2hex754

Cú pháp: void float2hex754(double realNumber, int bytes_num, char* hex_bytes_display)

Đầu vào: số thực dương và độ dài byte cần biểu diễn.

Đầu ra: chuỗi byte hexa biểu diễn số thực trên theo chuẩn IEEE754

Để xây dựng được hàm float2hex754, em sử dụng thêm các hàm phụ:

- void dec2bi (int decimalNumber, char* binary): chuyển thập phân sang nhị phân.
- void float2bi (double realNumber, char* binary): chuyển số thực dương nhỏ hơn 1 sang nhị phân.

Lưu đồ thuật toán của hàm được thể hiện ở hình II.2.

3. time_diff:

Cú pháp: int time_diff (struct Time *time_f_ptr, struct Time *time_l_ptr)

Đầu vào: thông tin 2 mốc thời gian

Đầu ra: khoảng cách giữa 2 mốc thời gian đó (giây)

Sử dụng hàm next_sample_time để xây dựng hàm.

4. checksum:

Cú pháp: void checksum (char hex_12[100], char* result)

Đầu vào: chuỗi ký tự chứa 12 byte hexa cách nhau bởi dấu cách

Đầu ra: kết quả checksum (1 byte)

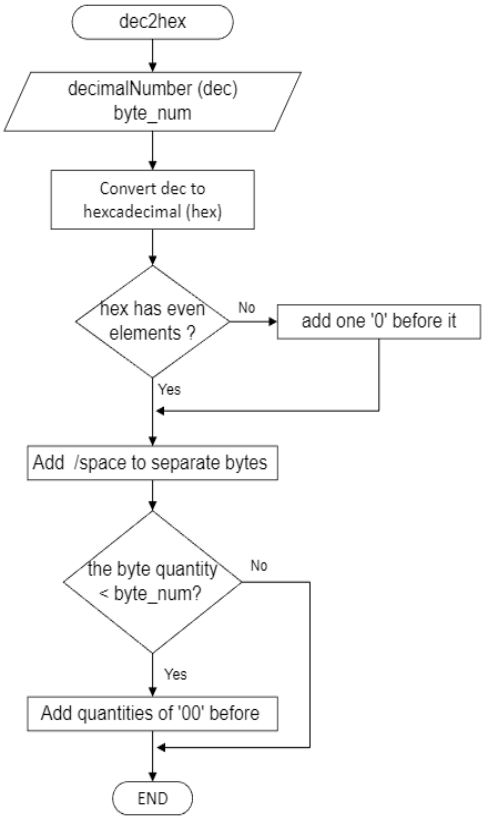
Sử dụng các hàm: hex2dec, dec2bi, dec2hex, bi2dec

5. str2time:

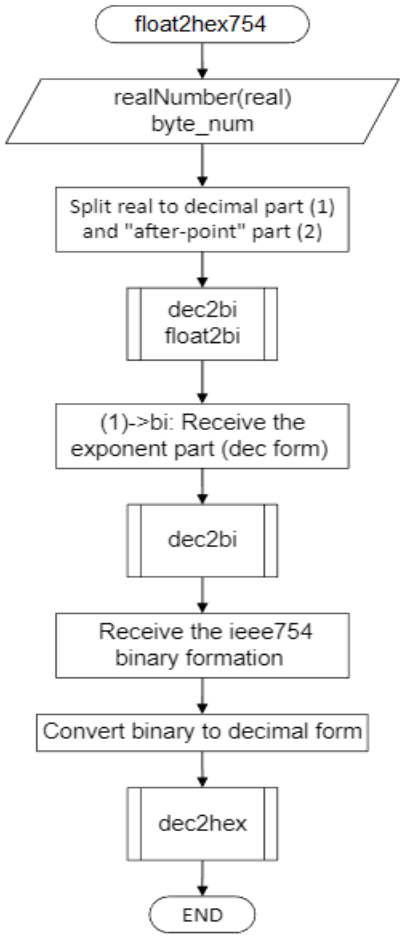
Cú pháp: void str2time (char str[100], struct Time *time)

Đầu vào: chuỗi ký tự chứa thông tin về thời gian theo định dạng YYYY:MM:DD hh:mm:ss.

Đầu ra: lưu dữ liệu vào structure Time đã định nghĩa trước đó.



Hình II.1: Lưu đồ dec2hex



Hình II.2: Lưu đồ floatt2hex754

B. Chế độ biên dịch bản tin truyền thông (Mode 02)

Với một file text có phần mở rộng là .dat chứa các chuỗi số hexa có định dạng yêu cầu, cần biên dịch và xuất ra một file csv chứa thông tin về dữ liệu cảm biến bụi dưới dạng ngôn ngữ thông thường. Tương tự ở *Mode 01*, sau khi qua quá trình để kiểm tra và lọc, cần thực hiện được các nhiệm vụ sau đây:

1. Biên dịch bản tin truyền thông.

Nhiệm vụ biên dịch chuỗi số hexa sang dạng text cần thực hiện việc:

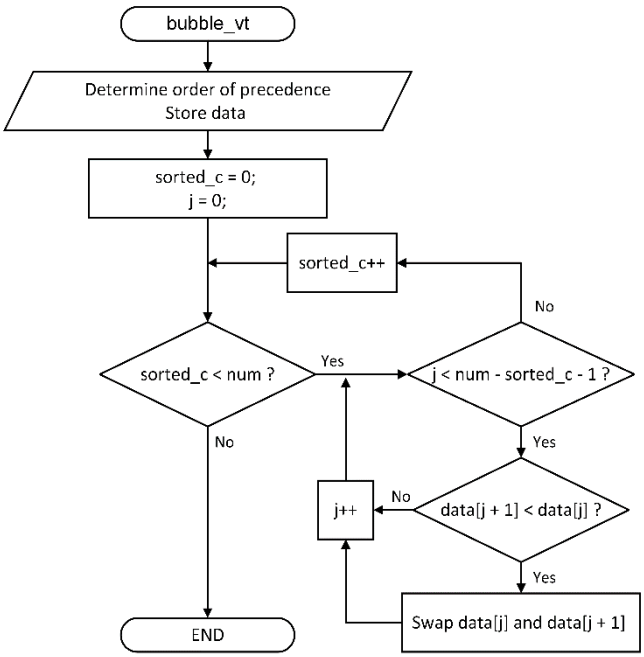
- ID: chuyển từ 1 byte hệ 16 sang hệ thập phân.
- Time: chuyển từ 4 bytes hệ 16 theo định dạng thời gian trong Unix sang thời gian theo đúng định dạng.
- Giá trị đo: chuyển từ số thực dạng hexadecimal theo chuẩn IEEE 754 (4 bytes) sang hệ thập phân.
- Start byte, Stop byte, 2 bytes chứa thông tin AQI, và 1 byte Checksum chuyển đổi phù hợp để phục vụ cho việc kiểm tra.

Việc chuyển đổi chủ yếu từ hệ 16 sang hệ thập phân, các hàm sử dụng chính: *hex2dec*, *bi2dec*, và *hex754_2float*.

- float hex754_2float (char str[9])
  Đầu vào: chuỗi 4-byte hexa biểu diễn số thực theo chuẩn IEEE754.
  Đầu ra: số thực đó ở hệ thập phân.

2. Sắp xếp dữ liệu

Với *Mode 02*, có lựa chọn cho phép người dùng có thể sắp xếp dữ liệu tăng hoặc giảm dần theo một thứ tự ưu tiên nhất định của các trường dữ liệu: Id, Time, và Values. Em lựa chọn hai thuật toán sắp xếp là *Bubble Sort* và *Quick Sort.*



Hình II.3: Lưu đồ bubble_vt

Ý tưởng xây dựng các hàm sắp xếp của em sẽ không ảnh hưởng đến dữ liệu hay trật tự của chúng đang lưu trữ. Đầu vào là dữ liệu và điều kiện của sắp xếp, đầu ra chỉ là mảng chứa số thứ tự dòng sau khi sắp xếp. Việc không thay đổi dữ liệu ban đầu sẽ làm tốn ít bộ nhớ cần lưu trữ, dễ dàng thực hiện nhiều thuật toán sắp xếp khác nhau để so sánh.

a. Bubble Sort:

Cú pháp: void bubble_vt (int *id, int *time, float *values, int num, char *pri, int sort_sign, int *result)

Đầu vào:

- Con trỏ đến dữ liệu cần sắp xếp
- Chuỗi kí tự xác định thứ tự ưu tiên sắp xếp
- Kiểu sắp xếp

Đầu ra:

- Thứ tự dữ liệu sau khi sắp xếp.

Lưu đồ thuật toán của hàm được thể hiện ở hình II.3

b. Quick Sort:

Cú pháp void quick_vt (int *id, int *time, float *values, int num, char *pri, int sort_sign, int *result)

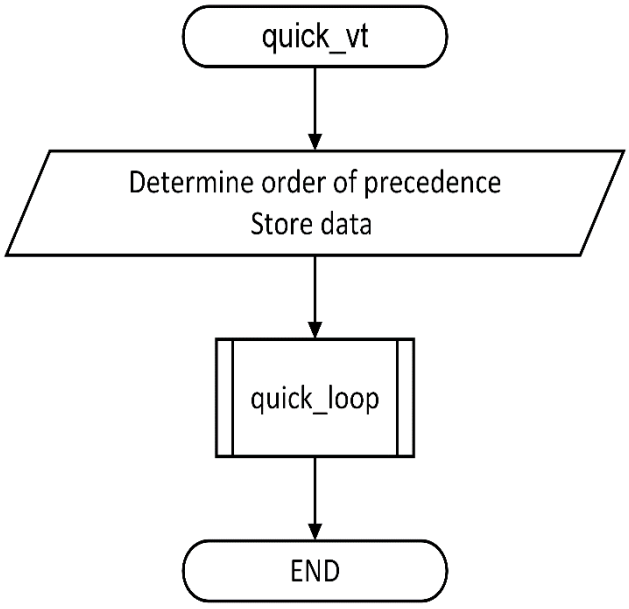
Đầu vào:

- Con trỏ đến dữ liệu cần sắp xếp
- Chuỗi kí tự xác định thứ tự ưu tiên sắp xếp
- Kiểu sắp xếp

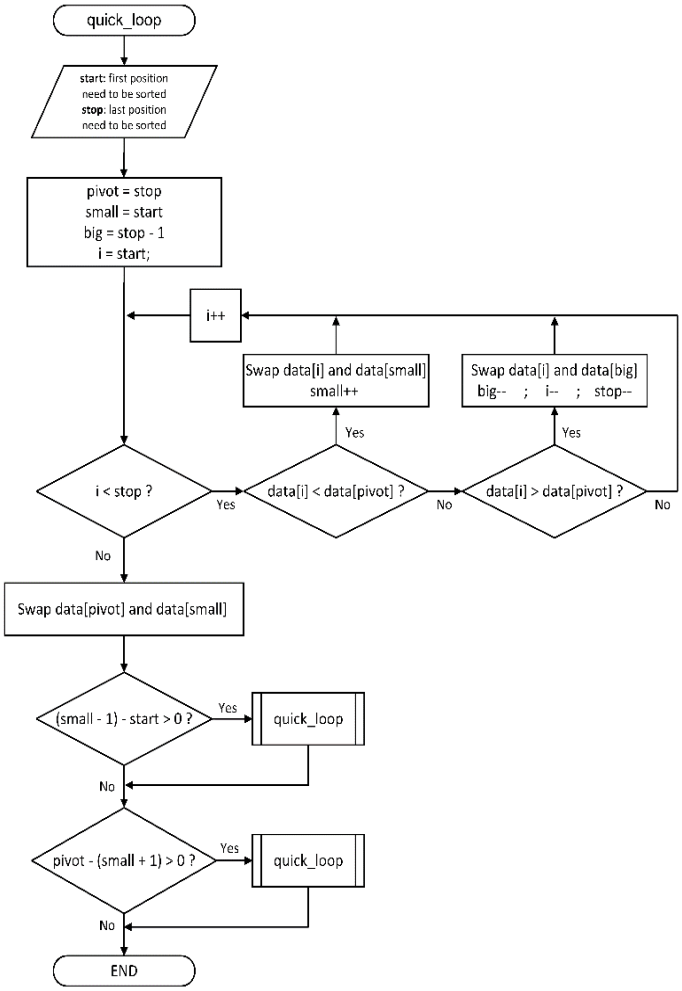
Đầu ra:

- Thứ tự dữ liệu sau khi sắp xếp.

Trong *quick_vt* xây dựng hàm *quick_loop* để thực hiện sắp xếp theo phương pháp đệ quy. Cú pháp: void quick_loop (float *data, int num, int start, int stop, int *result). *quick_loop* sẽ liên tục sắp xếp và chia nhỏ từng đoạn dữ liệu đến khi việc sắp xếp theo thuật toán sắp xếp nhanh hoàn tất. Lưu đồ thuật toán của *quick_vt* và *quick_loop* thể hiện lần lượt ở hình II.4a và hình II.4b.



Hình II.4a: Lưu đồ quick_vt



Hình II.4b: Lưu đồ quick_loop

3. Các yêu cầu kĩ thuật khác

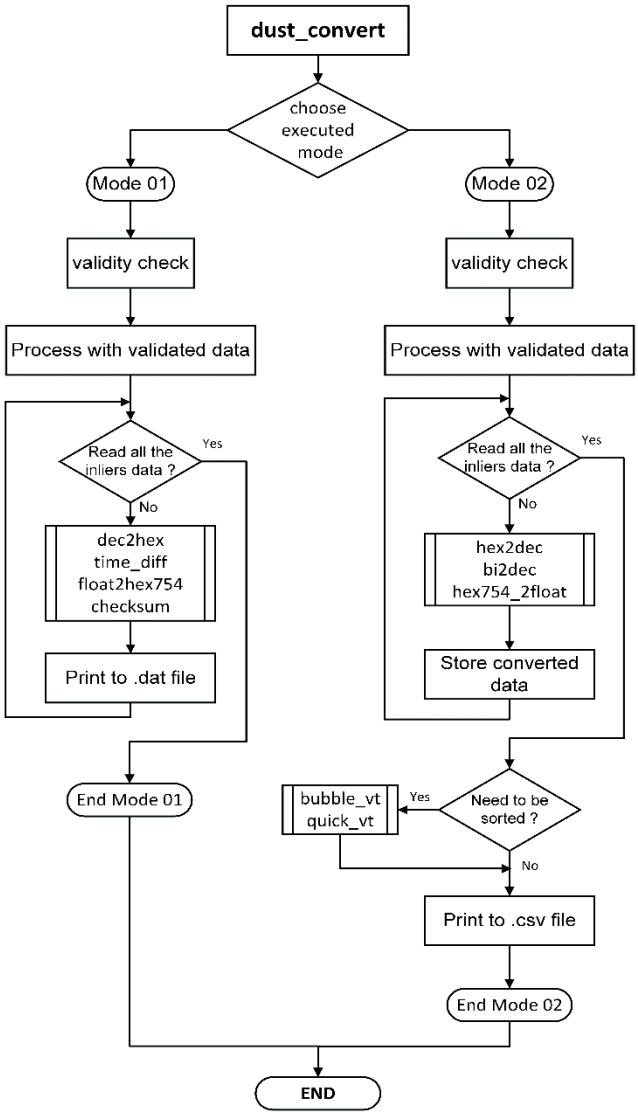
Chương trình chạy với command-line sẽ lưu lại các lỗi xảy ra vào 1 log file có tên là dust_convert_error.log. Các lỗi có thể xảy ra được định nghĩa như sau:

- Error 01: Invalid command.
- Error 02: Denied access FILENAME.
- Error 03: Invalid file format.
- Error 04: Data is mising at line XY.
- Error 05: Data is duplicated at line XX, YY.
- Error 06: Invalid data packet at line XY.

Chương trình còn cần lưu lại các thông số chạy chương trình vào 1 log file có tên là dust_convert_run.log thể hiện những thông số:

- Số dòng dữ liệu đầu vào.
- Số dòng chuyển đổi thành công.
- Số dòng bị lỗi.
- Thời gian thực hiện thuật toán sắp xếp (*Mode 02)*

Sau khi việc thiết kế được hoàn thiện, lưu đồ thuật toán cho toàn bộ chương trình được thể hiện ở hình II.5.



Hình II.5: Lưu đồ chương trình dust_convert.

III. KẾT LUẬN

Tổng quan về bài tập lớn, em đã thực hiện được cơ bản những yêu cầu đặt ra:

- Tạo bản tin truyền thông (*Mode 01*): chuyển đổi và xuất ra file theo yêu cầu với chất lượng tốt, đúng, đủ.
- Biên dịch bản tin truyền thông (*Mode 02*): chuyển đổi và xuất ra file theo yêu cầu với chất lượng tốt, đúng, đủ. Các thuật toán sắp xếp hoạt động trơn tru, cho kết quả chính xác.

Tuy vậy, kết quả thu được vẫn còn nhiều hạn chế:

- Tốc độ thực hiện chưa thật sự nhanh.
- Chưa thể bao quát hết các lỗi khi thực hiện chương trình.
- Chương trình chưa được ngắn gọn, tối ưu.

Em xin cảm ơn thầy Hoàng Đức Chính đã ra bài tập lớn số 02 cũng như các nội dung học khác rất chất lượng, giúp em được vận dụng các kiến thức đã học và tìm hiểu nhiều kiến thức mới, rất thiết thực với công việc thực tế.