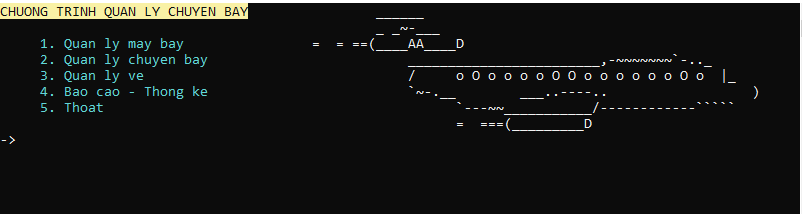
Quản Lý Chuyến Bay – C++ Console App

# Chức năng

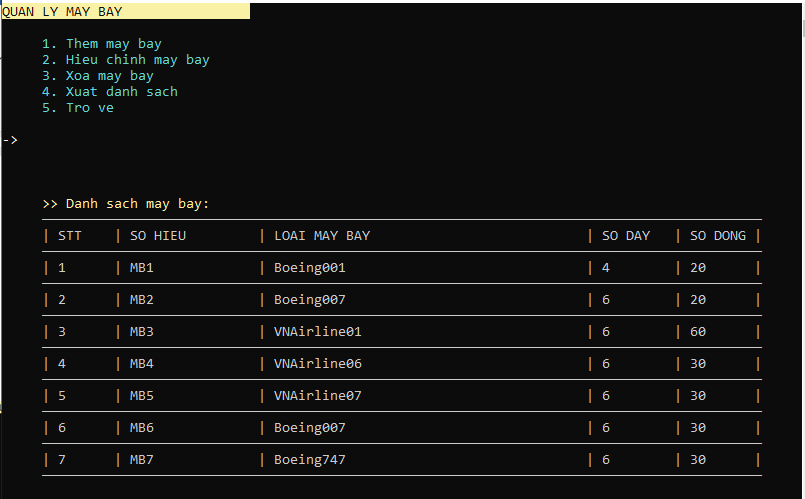
* Quản lý máy bay: Thêm, xoá, hiệu chỉnh, xuất danh sách máy bay
* Quản lý chuyến bay: Thêm, xoá, hiệu chỉnh, xuất danh sách chuyến bay
* Quản lý vé: Đặt vé, huỷ vé, xuất danh sách vé
* Báo cáo – Thống kê:
* Danh sách hành khách thuộc chuyến bay theo mã chuyến bay
* Danh sách chuyến bay khởi hành trong ngày x, tới nơi y.
* Danh sách vé trống của chuyến bay theo mã chuyến bay
* Thống kê số lượt thực hiện chuyến bay của mỗi máy bay theo thứ tự giảm dần.

# Giao diện

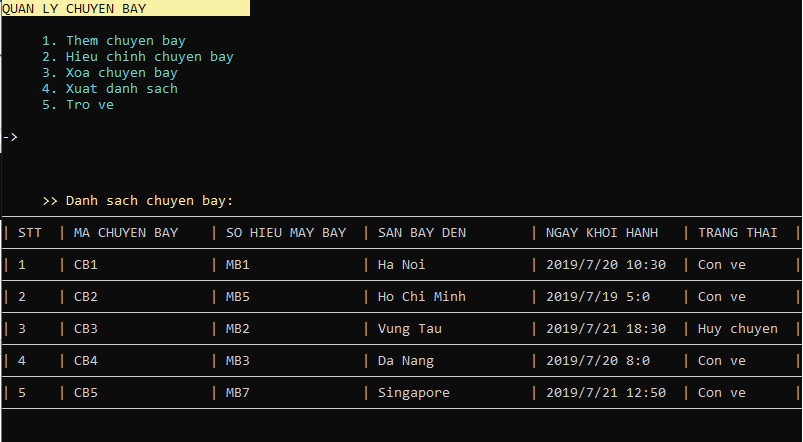
* Màn hình chính:



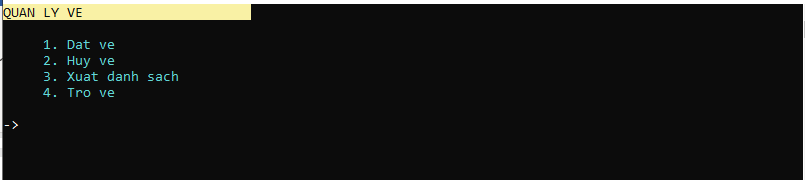
* Màn hình quản lý máy bay:



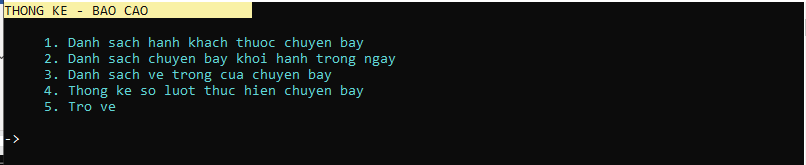
* Màn hình quản lý chuyến bay:



* Màn hình quản lý vé:



* Màn hình báo cáo – thống kê:



# Hướng dẫn sử dụng

* Thao tác phím theo yêu cầu từ console.
* Đối với các lựa chọn số (1:, 2:, …): nhập số và Enter
* Đối với các lựa chọn xác nhận (Y/N): chỉ cần nhập kí tự (‘y’ hoặc ‘Y’, ‘n’ hoặc ‘N’)
* Để dữ liệu được lưu lại cho lần chạy chương trình kế tiếp, cần chọn chức năng xuất cho mỗi đối tượng (chuyến bay, máy bay, vé)
* ***ESC để rollback từ bất kì màn hình nào.***
* ***Input đã được chặn: Kiểm tra các điều kiện khi nhập làm dữ liệu bị sai.***

# Cài đặt

* Source được build trên Visual Studio 2017.
* Khắc phục lỗi \_CRT\_SECURE\_NO\_WARNINGS:

<https://stackoverflow.com/a/20753468>

# Các tư tưởng chính

* **Màn hình console**: Là màn hình **không thuần** giao diện, chỉ có kí tự ASCII. Mọi thao tác code sẽ gồm các hoạt động chính:
* Tính toán, xử lý dữ liệu
* Di chuyển con trỏ tới vị trí chỉ định: **GotoXY(int col, int row)**
* Thiết lập màu text: **SetColor(int color)**
* In dữ liệu: **cout << text;**
* **Vòng lặp chương trình:**

Mỗi màn hình chức năng sẽ có các vòng lặp để lấy input từ người dùng và đưa ra xử lý thích hợp. Ở mỗi hàm xử lý sẽ có kết quả trả về. Nếu kết quả là -1, vòng lặp sẽ quay lại từ đầu tại bước in ra các lựa chọn và nhập input (thường thì ấn ESC sẽ trả về -1).

// In ra màn hình các lựa chọn

while (true) {

// Lấy input lựa chọn từ người dùng

switch (lựa chọn) {

case lựa chọn 1: DoSomething1(); break;

case lựa chọn 2: DoSomething2(); break;

...

case lựa chọn n : End(); return;

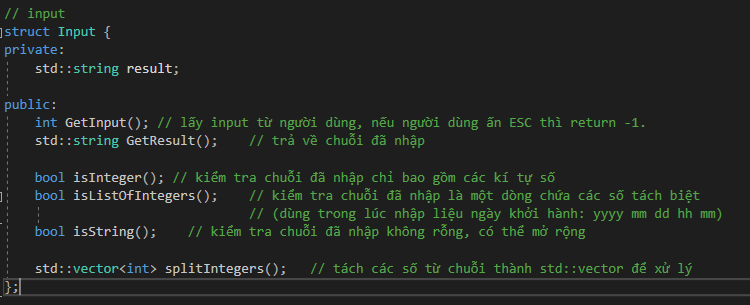
}

}

* **Input**:

Để hiện thực việc giới hạn input từ người dùng (nguyên nhân vì sao phải giới hạn? -> inbox) và việc bắt phím ESC (đây là phím đặc biệt) để xử lý, **bất cứ input nào cũng được nhập dưới dạng string**.

Cài đặt struct Input như sau (trong Utilities.h):



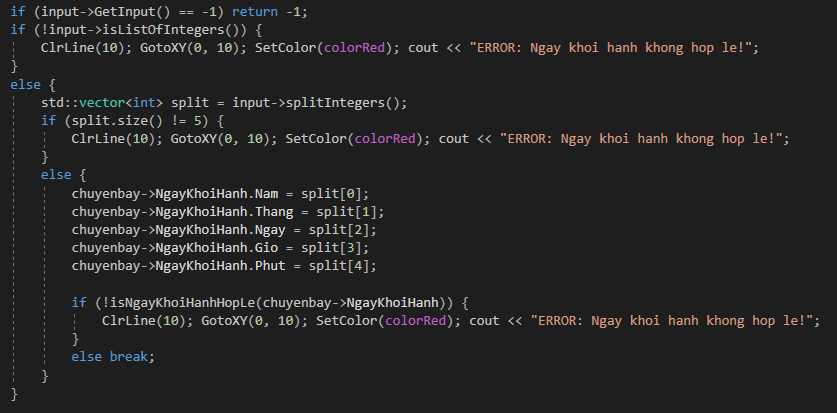
Ví dụ trường hợp nhập loại máy bay (trong hàm QuanLyMayBay::Add()):

if (input->GetInput() == -1) return -1;

maybay->LoaiMayBay = input->GetResult();

* input->GetInput(): Lấy input từ người dùng để gán vào chuỗi string::result, nếu người dùng ấn phím ESC thì kết quả hàm trả về là -1 (xem trong hiện thực hàm GetInput()). Lúc này, hàm sẽ ngay lập tức return về -1 (đây là dấu hiệu nhận biết cho yêu cầu rollBack – xem trong **vòng lặp chương trình**)
* input->GetResult(): Lấy kết quả chuỗi vừa nhập để gán cho biến maybay->LoaiMayBay

Ví dụ trường hợp nhập ngày khởi hành (trong hàm QuanLyChuyenBay::Add()): yêu cầu người dùng nhập một dãy các giá trị tương ứng cho **yyyy mm dd hh mm**



* input->GetInput(): tương tự như ở trên
* !input->isListOfIntegers(): nếu chuỗi vừa nhập không phải (kí hiệu **!**) là một dãy các số riêng biệt thì nghĩa là input không hợp lệ, thông báo lỗi.
* input->splitIntegers(): tách các giá trị của chuỗi vừa nhập và gán vào vector để tiếp tục kiểm tra
* split.size() != 5: Vì yêu cầu người dùng phải nhập **5** giá trị cho năm, tháng, ngày, giờ, phút, do đó nếu dãy các số vừa tách không đủ 5 giá trị có nghĩa là input không hợp lệ.
* Nếu input hợp lệ: Gán giá trị và kiểm tra logic của ngày tháng (hàm isNgayKhoiHanhHopLe())…
* Biến **static**

Là biến toàn cục, có quyền hạn truy cập ở mọi vị trí trong code.

* Đối với các biến được sử dụng lại nhiều lần thì việc khai báo đi khai báo lại rất là phiền phức và tốn bộ nhớ cho việc khởi tạo, do đó sử dụng từ khoá **static** sẽ giúp không cần phải khai báo lại các biến đó.

static Input \* input = new Input;

Cả chương trình chỉ cần sử dụng đúng một biến input để xử lí cho tất cả các input được nhập!

* Đối với các danh sách quản lý (danh sách hành khách, danh sách vé, danh sách máy bay, danh sách chuyến bay): Vì nó là quản lý nên nó chỉ có duy nhất 1 thực thể., và việc đặt cho nó quyền hạn **static** là điều tất yếu.

***Trong các struct quản lý sẽ có một biến static \_instance và hàm static getinstance(). Mục đích của nó là để đảm bảo rằng một thực thể quản lý là duy nhất xuyên suốt chương trình, mọi hoạt động đều phải thực hiện trên thực thể đó. Đây gọi là kỹ thuật Singleton pattern.***

QuanLyMayBay \* QuanLyMayBay::getinstance()

{

if (\_instance == NULL) \_instance = new QuanLyMayBay();

return \_instance;

}

* Nếu \_instance = NULL nghĩa là thực thể quản lý chưa được khởi tạo, cần khởi tạo nó qua toán tử new.
* Kết quả trả về của hàm là con trỏ đến thực thể quản lý đó.

Tại sao dùng con trỏ? (inbox)

* Cấu trúc dữ liệu:
* Danh sách Máy Bay: mảng tuyến tính các con trỏ.
* Danh sách Chuyến Bay: danh sách liên kết đơn
* Danh sách Vé: danh sách liên kết đơn
* Danh sách Hành Khách: cây tìm kiếm nhị phân

# Tổ chức mã nguồn

* **main.cpp**: Chứa các lệnh khởi tạo màn hình, import dữ liệu từ file .txt và vòng lặp chính của chương trình.
* **Utilities.h và Utilities.cpp**: Chứa các hàm hữu ích liên quan tới giao diện console, các hàm xử lý chuỗi và struct Input.

int InputKey(); // lấy giá trị ASCII của phím vừa được nhập

void Clrscr(); // xoá toàn bộ màn hình

void ClrLine(int line); // xoá dòng

void GotoXY(int column, int line); // di chuyển con trỏ tới vị trí col, row

int WhereX(); // lấy vị trí cột hiện tại của con trỏ

int WhereY(); // lấy vị trí dòng hiện tại của con trỏ

void SetColor(int color); // thiết lập màu cho text

int StringToInteger(std::string s); // chuyển đổi giá trị kiểu string qua kiểu integer

std::string LineStandardize(std::string s); // chuẩn hoá chuỗi

// struct Input

struct Input {

private:

std::string result; // chuỗi kết quả của input từ người dùng

public:

int GetInput(); // hàm xử lí lấy input từ người dùng, nếu người dùng ấn ESC thì return -1.

std::string GetResult(); // trả về chuỗi đã nhập

bool isInteger(); // kiểm tra chuỗi đã nhập chỉ bao gồm các kí tự số

bool isListOfIntegers(); // kiểm tra chuỗi đã nhập là một dòng chứa các số tách biệt

// (dùng trong lúc nhập liệu ngày khởi hành: yyyy mm dd hh mm)

bool isString(); // kiểm tra chuỗi đã nhập không rỗng, có thể mở rộng

std::vector<int> splitIntegers(); // tách các số từ chuỗi thành std::vector để xử lý

};

static Input \* input = new Input; // khai báo đối tượng toàn cục

* **Objects.h**: Định nghĩa các đối tượng của chương trình

struct MayBay

{

std::string SoHieuMayBay;

std::string LoaiMayBay;

int SoDay;

int SoDong;

};

struct Ve

{

std::string MaChuyenBay;

long SoCMND;

int ViTriDay;

int ViTriHang;

};

struct ThoiGian

{

int Ngay;

int Thang;

int Nam;

int Gio;

int Phut;

bool HopLe() { // kiểm tra thời gian hợp lệ

if (Nam < 1900) return 0;

if (Gio < 0 || Gio > 23) return 0;

if (Phut < 0 || Phut > 59) return 0;

switch (Thang)

{

case 1: case 3: case 5: case 7: case 8: case 10: case 12:

if (Ngay < 0 || Ngay > 31) return 0;

break;

case 4: case 6: case 9: case 11:

if (Ngay < 0 || Ngay > 30) return 0;

break;

case 2:

if ((Nam % 4 == 0 && Nam % 100 != 0) || (Nam % 400 == 0)) {

if (Ngay < 0 || Ngay > 29) return 0;

}

else {

if (Ngay < 0 || Ngay > 28) return 0;

}

default:

return 0;

}

return 1;

}

};

struct ChuyenBay

{

std::string MaChuyenBay;

std::string SoHieuMayBay;

std::string SanBayDen;

ThoiGian NgayKhoiHanh;

int TrangThai; // 0: hủy chuyến, 1: còn vé, 2: hết vé, 3: hoàn tất

};

struct HanhKhach

{

long SoCMND;

std::string Ho;

std::string Ten;

std::string Phai;

};

* **QuanLyMayBay.h và QuanLyMayBay.cpp**: Định nghĩa struct Quản Lý Máy Bay, hiện thực theo mảng tuyến tính các con trỏ, các hàm xử lí, hiển thị cho các chức năng của màn hình (thêm, xoá, sửa, xuất)

struct QuanLyMayBay

{

private:

int SoLuongMayBay;

MayBay\* DanhSach[300]; // mảng tuyến tính con trỏ chứa các đối tượng

public:

static QuanLyMayBay\* \_instance; // biến thực thể toàn cục

QuanLyMayBay(); // constructor

static QuanLyMayBay \* getinstance(); // lấy biến thực thể toàn cục

int data\_import(); // nhập dữ liệu từ file MayBay.txt

int data\_export(); // xuất dữ liệu ra file MayBay.txt

void add(MayBay \* value); // thêm máy bay vào danh sách

void delete\_byposition(int i); // xoá máy bay khỏi danh sách dựa vào vị trí

void update\_byposition(int i, MayBay \* value); // cập nhật giá trị máy bay dựa vào vị trí

std::string Build\_SoHieuMayBay(); // tự động tạo mới số hiệu máy bay

bool isLoaiHopLe(std::string str); // kiểm tra chuỗi nhập vào hợp lệ cho Loại máy bay

MayBay \* getBy\_SoHieuMayBay(std::string sohieu); // tìm kiếm máy bay dựa vào số hiệu

std::vector<MayBay> getDanhSachMayBay(); // lấy danh sách máy bay, trả về dưới dạng vector

// display

int Menu(); // hiển thị menu và xử lí input

void ShowList(); // hiển thị danh sách máy bay

int Add(); // hiển thị màn hình thêm máy bay và xử lý input

int Modify(); // hiển thị màn hình hiệu chỉnh máy bay và xử lý input

int Delete(); // hiển thị màn hình xoá máy bay và xử lý input

int Export(); // thực hiện xuất dữ liệu

};

* **QuanLyChuyenBay.h và QuanLyChuyenBay.cpp**: Định nghĩa struct Quản Lý Chuyến Bay, hiện thực theo danh sách liên kết đơn, các hàm xử lí, hiển thị cho các chức năng của màn hình (thêm, xoá, sửa, xuất)

struct node\_ChuyenBay // khai báo cấu trúc node của danh sách liên kết đơn

{

ChuyenBay data;

node\_ChuyenBay \*next;

};

struct QuanLyChuyenBay

{

private:

int SoLuongChuyenBay;

node\_ChuyenBay \*head, \*tail; // head, tail của danh sách liên kết đơn

public:

static QuanLyChuyenBay\* \_instance; // biến thực thể toàn cục

QuanLyChuyenBay(); // constructor

static QuanLyChuyenBay \* getinstance(); // lấy biến thực thể toàn cục

int data\_import(); // nhập dữ liệu từ ChuyenBay.txt

int data\_export(); // xuất dữ liệu ra ChuyenBay.txt

void insert(ChuyenBay value); // thêm dữ liệu vào danh sách liên kết (thêm đuôi)

void update\_byposition(int pos, ChuyenBay value); // cập nhật dựa vào số thứ tự trong danh sách

void delete\_byposition(int pos); // xoá dựa vào số thứ tự trong danh sách

ChuyenBay get\_byposition(int pos); // lấy thông tin dựa vào số thứ tự trong danh sách

ChuyenBay \* get\_bymachuyenbay(std::string machuyenbay); // tìm kiếm chuyến bay dựa vào mã chuyến bay

MayBay \* getMayBay\_bymachuyenbay(std::string machuyenbay); // lấy thông tin của máy bay dựa vào mã chuyến bay

std::vector<ChuyenBay> getAll(); // lấy danh sách chuyến bay, trả về kiểu vector

std::vector<ChuyenBay> getAll(ThoiGian t, std::string sanbayden); // lấy danh sách chuyến bay dựa vào ngày khởi hành và sân bay đến, trả về kiểu vector

std::string Build\_MaChuyenBay(); // tự động tạo mới mã chuyến bay

bool isSoHieuMayBayHopLe(std::string str); // kiểm tra số hiệu hợp lệ

bool isSanBayDenHopLe(std::string str); // kiểm tra sân bay đến hợp lệ

bool isNgayKhoiHanhHopLe(ThoiGian tg); // kiểm tra ngày khởi hành hợp lệ

bool isTrangThaiHopLe(int trangthai); // kiểm tra trạng thái hợp lệ

// display

int Menu(); // hiển thị menu và xử lý input

void ShowList(); // hiển thị danh sách chuyến bay

void ShowTime(ThoiGian c); // định dạng in ra của giá trị thời gian

void ShowState(int state); // định dạng in ra của giá trị trạng thái

int Add(); // hiển thị màn hình thêm chuyến bay và xử lý input

int Modify(); // hiển thị màn hình hiệu chỉnh chuyến bay và xử lý input

int Delete(); // hiển thị màn hình xoá chuyến bay và xử lý input

int Export(); // thực hiện xuất dữ liệu

};

* **QuanLyVe.h và QuanLyVe.cpp**: Định nghĩa struct Quản lý Vé, hiện thực theo danh sách liên kết đơn, các hàm xử lý, hiển thị cho các chức năng của màn hình (đặt vé, huỷ vé)

struct node\_Ve // định nghĩa node của danh sách liên kết đơn

{

Ve data;

node\_Ve \*next;

};

struct QuanLyVe

{

private:

int SoLuongVe;

node\_Ve \*head, \*tail; // head, tail

public:

static QuanLyVe\* \_instance; // biến thực thể toàn cục

QuanLyVe(); // constructor

static QuanLyVe \* getinstance(); // lấy biến thực thể toàn cục

int data\_import(); // nhập dữ liệu từ file Ve.txt

int data\_export(); // xuất dữ liệu ra file Ve.txt

void add(Ve value); // thêm dữ liệu (thêm đuối)

void remove(std::string machuyenbay, long soCMND); // xoá dữ liệu dựa vào mã chuyến bay và số CMND của hành khách

Ve \* find(std::string machuyenbay, long soCMND); // tìm kiếm thông tin vé dựa vào mã chuyến bay và số CMND của hành khách

std::vector<Ve> getAll(std::string machuyenbay); // lấy danh sách vé đã đặt của chuyến bay dựa vào mã chuyến bay

int KiemTraMaChuyenBay(std::string str); // kiểm tra input mã chuyến bay

int KiemTraSoCMND(std::string str); // kiểm tra input số CMND

int KiemTraDieuKienDatVe(std::string machuyenbay, long soCMND); // mỗi hành khách chỉ được đặt 1 vé trên chuyến bay

int KiemTraViTriHopLe(int vitriday, int vitrihang); // kiểm tra số ghế có bị trùng lúc đặt hay không

bool isHoTenHopLe(std::string str); // kiểm tra input họ tên

bool isPhaiHopLe(int phai); // kiểm tra input phái

bool isViTriDayHopLe(std::string str, int soday); // kiểm tra vị trí dãy hợp lệ

bool isViTriDongHopLe(std::string str, int sodong); // kiểm tra vị trí dòng hợp lệ

// display

int Menu(); // hiển thị menu và xử lý input

int Add(); // hiển thị màn hình đặt vé và xử lý input

int Remove(); // hiển thị màn hình huỷ vé và xử lý input

int Export(); // thực hiện xuất danh sách vé ra file

void ShowInfoChuyenBay(ChuyenBay \* t); // hiển thị thông tin chuyến bay

void ShowInfoHanhKhach(HanhKhach \* t); // hiển thị thông tin của hành khách

void ShowInfoVe(Ve \* t); // hiển thị thông tin vé

};

* **QuanLyHanhKhach.h và QuanLyHanhKhach.cpp**: Định nghĩa struct Quản Lý Hành Khách, hiện thực theo cây tìm kiếm nhị phân. Vì yêu cầu của đề tài không có chức năng quản lý hành khách, do đó màn hình thêm hành khách sẽ được gom vào chức năng Đặt vé của Quản Lý Vé (xem yêu cầu chức năng 3).

struct node\_HanhKhach // định nghĩa node của cây tìm kiếm nhị phân

{

HanhKhach data;

node\_HanhKhach\* left;

node\_HanhKhach\* right;

};

struct QuanLyHanhKhach

{

private:

int SoLuongHanhKhach;

node\_HanhKhach\* root; // root

public:

static QuanLyHanhKhach\* \_instance; // biến thực thể toàn cục

QuanLyHanhKhach(); // constructor

~QuanLyHanhKhach(); // destructor

static QuanLyHanhKhach \* getinstance(); // lấy biến thực thể toàn cục

int data\_import(); // nhập dữ liệu từ file HanhKhach.txt

int data\_export(); // xuất dữ liệu ra file HanhKhach.txt

void recursion\_data\_export(std::ofstream &out, node\_HanhKhach\* t); // vì cây tìm kiếm nhị phân hiện theo thuật toán đệ quy, do đó việc duyệt từng node để xuất ra file cũng phải thực hiện theo đệ quy

node\_HanhKhach\* clear(node\_HanhKhach\* t); // huỷ cây nhị phân

node\_HanhKhach\* insert(HanhKhach x, node\_HanhKhach\* t); // thêm dữ liệu (đệ quy)

node\_HanhKhach\* findMin(node\_HanhKhach\* t); // tìm kiếm giá trị nhỏ nhất

node\_HanhKhach\* findMax(node\_HanhKhach\* t); // tìm kiếm giá trị lớn nhất

node\_HanhKhach\* remove(HanhKhach x, node\_HanhKhach\* t); // huỷ liên kết giá trị trong cây nhị phân

node\_HanhKhach\* find(node\_HanhKhach\* t, long soCMND); // tìm kiếm hành khách dựa vào số CMND (đệ quy)

HanhKhach\* Find(long soCMND); // tìm kiếm hành khách dựa vào số CMND

void Insert(HanhKhach x); // thêm dữ liệu

};

* **ThongKeBaoBieu.h và ThongKeBaoBieu.cpp**: Chứa các hàm xử lí, hiển thị và xuất danh sách cho 4 yêu cầu về thống kê – báo cáo.

struct ThongKeBaoBieu

{

public:

ChuyenBay \* KiemTraMaChuyenBay(std::string machuyenbay); // kiểm tra input mã chuyến bay

int Menu(); // hiển thị menu và xử lý input

int QuanLyHanhKhach(); // hiển thị màn hình xử lý danh sách hành khách thuộc chuyến bay

int ExportQuanLyHanhKhach(ChuyenBay chuyenbay, std::vector<Ve> dsve, std::vector<HanhKhach> dshanhkhach); // Xuất danh sách ra file DanhSachHanhKhach.txt

void ShowQuanLyHanhKhach(ChuyenBay chuyenbay, std::vector<Ve> dsve, std::vector<HanhKhach> dshanhkhach); // Hiển thị danh sách ra màn hình

int QuanLyChuyenBay(); // hiển thị màn hình xử lý danh sách chuyến bay trong ngày + nơi đến

int ExportQuanLyChuyenBay(ThoiGian ngaykhoihanh, std::string sanbayden, std::vector<ChuyenBay> dschuyenbay); // Xuất DanhSachChuyenBay.txt

void ShowQuanLyChuyenBay(ThoiGian ngaykhoihanh, std::string sanbayden, std::vector<ChuyenBay> dschuyenbay); // Hiển thị

int QuanLyVeTrong(); // hiển thị màn hình xử lý danh sách vé trống của chuyến bay

int ExportQuanLyVeTrong(ChuyenBay chuyenbay, std::vector<std::vector<bool>> vitri); // Xuất DanhSachVeTrong.txt

void ShowQuanLyVeTrong(ChuyenBay chuyenbay, std::vector<std::vector<bool>> vitri); // Hiển thị

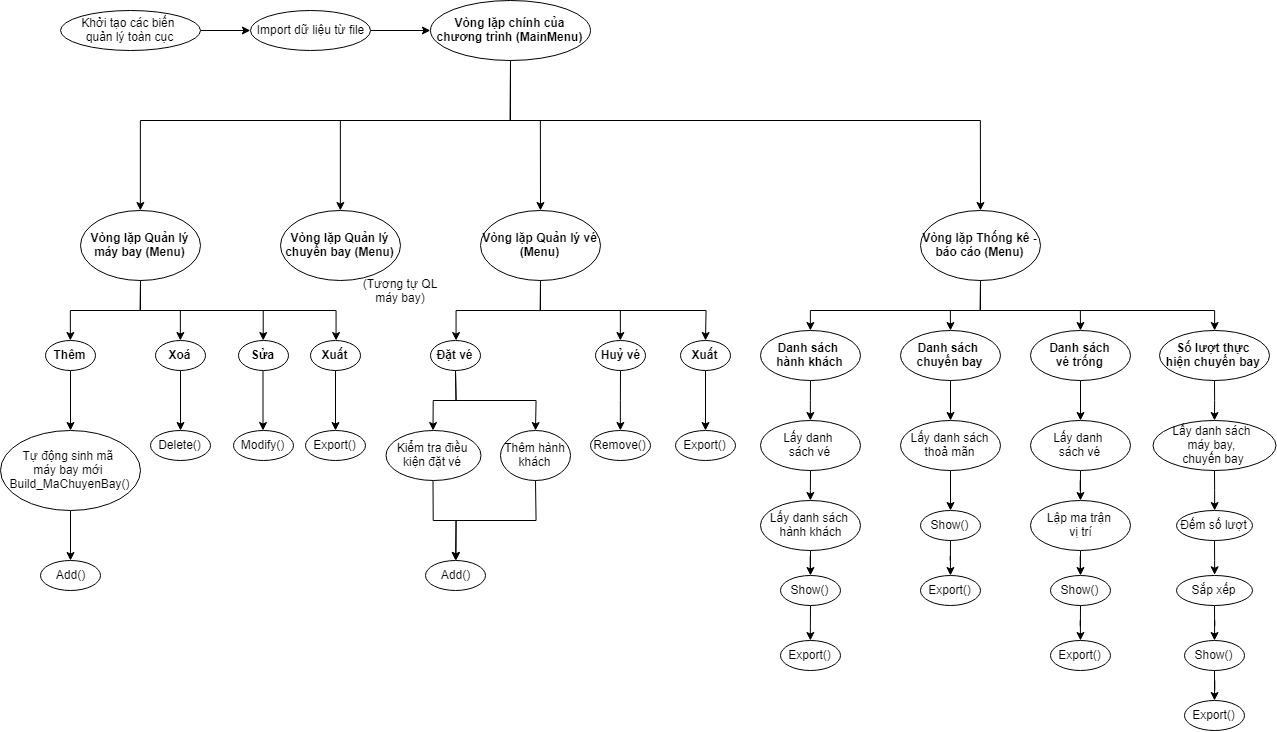
int SoLuotThucHienChuyenBay(); // hiển thị màn hình xử lý số lượt thực hiện của máy bay

int ExportSoLuotThucHienChuyenBay(std::vector<MayBay> dsmaybay, std::vector<int> soluotthuchien); // Xuất SoLuotThucHienChuyenBay.txt

void ShowSoLuotThucHienChuyenBay(std::vector<MayBay> dsmaybay, std::vector<int> soluotthuchien); // Hiển thị

};

# Sơ đồ các luồng xử lý chính



# Cấu trúc file .txt

* Gồm có 2 thư mục: data (dữ liệu danh sách các đối tượng) và thongke (chứa các danh sách được export từ chức năng thống kê – báo cáo)
* File dữ liệu:
* Dòng đầu tiên: Số lượng
* Các dòng tiếp theo: Các giá trị thuộc tính của đối tượng (xem các hàm data\_export() và data\_import() trong các struct Quản lý để hiểu rõ hơn)