**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA ĐIỆN- ĐIỆN TỬ**

Logo

Description automatically generated

**ĐỀ TÀI CUỐI KỲ**

**MÔN: Giải thuật và Cấu trúc dữ liệu**

**QUẢN LÝ CÁC CHUYẾN BAY NỘI ĐỊA THUỘC 1 HÃNG HÀNG KHÔNG**

|  |  |
| --- | --- |
| SVTH: | MSSV |
| NGUYỄN THANH KIÊN | 20119351 |
| NGUYỄN ĐỨC BÌNH  TRẦN TRƯỜNG CA  PHAN HOÀNG PHÚC | 20119320  20119323  20119088 |

Khóa: 2020

Tp. Hồ Chí Minh, tháng 5 năm 2023

**MỤC LỤC**

[**CHƯƠNG 1: GIẢI THUẬT VÀ CẤU TRÚC DỮ LIỆU TỪNG PHẦN** 1](#_Toc134995597)

[**1.1. Máy bay** 1](#_Toc134995598)

[**1.1.1. Định nghĩa, cách hoạt động và tính chất cơ bản của danh sách tuyến tính** 1](#_Toc134995599)

[**1.1.2. Giải thuật và cấu trúc dữ liệu phần Máy bay** 2](#_Toc134995600)

[**1.2. Hành khách** 6](#_Toc134995601)

[**1.2.1. Định nghĩa, cách hoạt động và tính chất cơ bản của cây tìm kiếm nhị phân** 6](#_Toc134995602)

[**1.2.2. Giải thuật và cấu trúc dữ liệu phần Hành Khách** 7](#_Toc134995603)

[**1.3 Chuyến Bay** 14](#_Toc134995604)

[**1.3.1. Định nghĩa, cách hoạt động và tính chất cơ bản của danh sách liên kết đơn** 14](#_Toc134995605)

[**1.3.2. Giải thuật và cấu trúc dữ liệu phần Chuyến bay** 15](#_Toc134995606)

[**CHƯƠNG 2: TỔNG QUÁT HOẠT ĐỘNG VÀ CHỨC NĂNG CỦA HỆ THỐNG** 24](#_Toc134995607)

[**2.1. Kết quả** 24](#_Toc134995608)

[**2.1.1. Cập nhật danh sách các máy bay ( thêm/ xóa/hiệu chỉnh )** 24](#_Toc134995609)

[**2.1.2.** **Cập nhật chuyến bay** 31](#_Toc134995610)

[**2.1.3. Đặt vé** 35](#_Toc134995611)

[**2.1.4. In danh sách các hành khách thuộc 1 chuyến bay dựa vào mã chuyến bay** 38](#_Toc134995612)

[**2.1.5. In danh sách các chuyến bay khởi hành trong ngày dd/mm/yyyy đến nơi XXXX mà còn vé ( cho biết cụ thể số lượng các vé còn trống )** 39](#_Toc134995613)

[**2.1.6. In danh sách các vé còn trống của 1 chuyến bay có mã chuyến bay là X** 40](#_Toc134995614)

[**2.1.7. Thống kê số lượt thực hiện chuyến bay của từng máy bay theo thứ tự số lượt thực hiện giảm dần** 41](#_Toc134995615)

# **CHƯƠNG 1: GIẢI THUẬT VÀ CẤU TRÚC DỮ LIỆU TỪNG PHẦN**

## **1.1. Máy bay**

### **1.1.1. Định nghĩa, cách hoạt động và tính chất cơ bản của danh sách tuyến tính**

Trong đề tài cuối kỳ này, danh sách tuyến tính được sử dụng để thêm/xoá máy bay với các thuộc tính như: Số hiệu máy bay, loại máy bay, số chỗ,…

Danh sách tuyến tính là một cấu trúc dữ liệu trong lập trình, cho phép lưu trữ một tập hợp các phần tử theo thứ tự. Các phần tử trong danh sách tuyến tính được lưu trữ liên tiếp trong bộ nhớ máy tính.

Danh sách tuyến tính cung cấp các phương thức để thêm, xóa, tìm kiếm và truy cập các phần tử trong danh sách. Một số phương thức phổ biến bao gồm:

Thêm phần tử vào danh sách:

* Thêm phần tử vào đầu danh sách.
* Thêm phần tử vào cuối danh sách.
* Thêm phần tử vào vị trí bất kỳ trong danh sách.

Xóa phần tử khỏi danh sách:

* Xóa phần tử từ đầu danh sách.
* Xóa phần tử từ cuối danh sách.
* Xóa phần tử tại vị trí bất kỳ trong danh sách.

Truy cập phần tử trong danh sách:

* Truy cập phần tử từ đầu danh sách.
* Truy cập phần tử từ cuối danh sách.
* Truy cập phần tử tại vị trí bất kỳ trong danh sách.

Danh sách tuyến tính có thể được triển khai bằng các cấu trúc dữ liệu như mảng (array) hoặc danh sách liên kết (linked list). Mỗi phương pháp triển khai có những ưu điểm và hạn chế riêng, và lựa chọn phụ thuộc vào yêu cầu và tình huống cụ thể.

### **1.1.2. Giải thuật và cấu trúc dữ liệu phần Máy bay**

Ta khai báo các thư viện cần thiết để chương trình có thể hoạt động được.

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FlightForm

{

Đoạn code trên định nghĩa một lớp là MayBay, mô tả các thuộc tính và phương thức của một máy bay.

Các thuộc tính của lớp MayBay bao gồm:

* SoHieuMB: Đại diện cho số hiệu của máy bay, kiểu dữ liệu là chuỗi (string).
* LoaiMB: Đại diện cho loại máy bay, kiểu dữ liệu là chuỗi (string).
* SoCho: Đại diện cho số chỗ ngồi của máy bay, kiểu dữ liệu là số nguyên (int).
* SoChuyenBay: Đại diện cho số chuyến bay của máy bay, kiểu dữ liệu là số nguyên (int).

Lớp MayBay có hai phương thức khởi tạo:

Phương thức khởi tạo MayBay(string shmb, string lmb, int socho): Nhận vào ba đối số là shmb (số hiệu máy bay), lmb (loại máy bay) và socho (số chỗ ngồi).

Phương thức khởi tạo MayBay(string shmb, string lmb, int socho, int sochuyenbay): Nhận vào bốn đối số là shmb (số hiệu máy bay), lmb (loại máy bay), socho (số chỗ ngồi) và sochuyenbay (số chuyến bay).

public class MayBay

{

public string SoHieuMB { get; set; }

public string LoaiMB { get; set; }

public int SoCho { get; set; }

public int SoChuyenBay { get; set; }

public MayBay(string shmb, string lmb, int socho)

{

this.SoHieuMB = shmb;

this.LoaiMB = lmb;

this.SoCho = socho;

}

public MayBay(string shmb, string lmb, int socho, int sochuyenbay)

{

this.SoHieuMB = shmb;

this.LoaiMB = lmb;

this.SoCho = socho;

this.SoChuyenBay = sochuyenbay;

}

}

Đoạn code dưới thực hiện 1 lớp DanhSachMayBay với các chức năng như ThemMayBay, XoaMayBay, TimMayBay, HieuChinh, DocData, GhiData.

public class DanhSachMayBay

{

Đoạn code dưới khai báo và khởi tạo một mảng đối tượng dsMayBay kiểu MayBay với kích thước tối đa là Define.MAX\_MAYBAY. Define.MAX\_MAYBAY được giả sử là một hằng số được định nghĩa là 300 tức là danh sách máy bay có 300 phân tử tối đa. Mỗi phần tử trong mảng dsMayBay đại diện cho một đối tượng máy bay.

Tiếp theo, biến soLuong được khởi tạo với giá trị là 0. Biến này sẽ được sử dụng để lưu trữ số lượng máy bay hiện có trong mảng dsMayBay. Ban đầu, vì soLuong được khởi tạo là 0, có nghĩa là chưa có máy bay nào được thêm vào mảng. Khi một máy bay mới được thêm vào mảng, giá trị của soLuong sẽ được tăng lên để đồng bộ với số lượng máy bay hiện có.

public static MayBay[] dsMayBay = new MayBay[Define.MAX\_MAYBAY];

public static int soLuong = 0;

Phương thức ThemMayBay, có ba tham số đầu vào là shmb (Số hiệu máy bay), lmb (Loại máy bay) và socho (Số chỗ trong máy bay).

Bước đầu tiên, phương thức kiểm tra xem số lượng máy bay hiện có (soLuong) có vượt quá giới hạn tối đa (Define.MAX\_MAYBAY) hay không. Nếu vượt quá, nghĩa là không còn chỗ trống để thêm máy bay mới, phương thức trả về giá trị -1 để báo lỗi.

Nếu số lượng máy bay chưa vượt quá giới hạn, phương thức tiếp tục thêm một đối tượng máy bay mới vào mảng dsMayBay tại vị trí soLuong bằng cách tạo một đối tượng MayBay mới. Sau đó, giá trị của soLuong được tăng lên một đơn vị để đồng bộ với số lượng máy bay đã được thêm vào.

Cuối cùng, phương thức trả về giá trị 1 để chỉ ra rằng máy bay đã được thêm thành công.

public static int ThemMayBay(string shmb, string lmb, int socho)

{

if (soLuong >= Define.MAX\_MAYBAY)

{

return -1; // nếu quá tải thì báo lỗi

}

dsMayBay[soLuong] = new MayBay(shmb, lmb, socho);

soLuong++;

return 1; // nếu còn chỗ thì thêm chuyến bay

}

Phương thức XoaMayBay, có một tham số đầu vào là viTri (kiểu số nguyên).

Phương thức XoaMayBay được sử dụng để xóa một máy bay khỏi mảng dsMayBay dựa trên vị trí viTri được chỉ định.

Một vòng lặp for được sử dụng để di chuyển qua các phần tử trong mảng dsMayBay, từ viTri và lặp đến soLuong - 1. Trong mỗi vòng lặp, phần tử dsMayBay[i] được gán bằng phần tử tiếp theo trong mảng, tức là dsMayBay[i + 1]. Điều này có nghĩa là phần tử tại vị trí viTri sẽ bị ghi đè bởi phần tử tiếp theo trong mảng. Quá trình này tiếp tục cho đến khi đến phần tử cuối cùng trong mảng.

Sau vòng lặp, phần tử cuối cùng trong mảng dsMayBay (tức là dsMayBay[soLuong - 1]) được gán giá trị null, để xóa đối tượng máy bay cuối cùng khỏi mảng.

Cuối cùng, giá trị của soLuong được giảm đi một đơn vị để đồng bộ với số lượng máy bay trong mảng sau khi xóa.

public static void XoaMayBay(int viTri)

{

for (int i = viTri; i < soLuong - 1; i++)

{

dsMayBay[i] = dsMayBay[i + 1];

}

dsMayBay[soLuong - 1] = null;

soLuong--;

}

Phương thức "TimMayBay" được định nghĩa với một tham số đầu vào là "soHieuMB" kiểu chuỗi và kiểu trả về là một số nguyên.

Ý nghĩa của đoạn code này là tìm kiếm một máy bay trong danh sách các máy bay dựa trên số hiệu máy bay (soHieuMB).

Phương thức sử dụng vòng lặp for để duyệt qua từng phần tử trong danh sách máy bay (biến dsMayBay là một mảng chứa các đối tượng máy bay). Biến soLuong là một biến kiểu số nguyên (int) lưu trữ số lượng máy bay trong danh sách.

Trong mỗi lần lặp, phương thức kiểm tra nếu số hiệu máy bay của phần tử hiện tại (dsMayBay[i].SoHieuMB) khớp với số hiệu máy bay được truyền vào (soHieuMB), thì phương thức trả về chỉ số của phần tử trong danh sách (return i) để chỉ ra vị trí tìm thấy máy bay.

Nếu không tìm thấy máy bay nào khớp với số hiệu được truyền vào, thì phương thức trả về giá trị -1 để thể hiện việc không tìm thấy máy bay nào trong danh sách.

public static int TimMayBay(string soHieuMB)

{

for (int i = 0; i < soLuong; i++)

{

if (dsMayBay[i].SoHieuMB == soHieuMB)

{

return i;

}

}

return -1;

}

public static void HieuChinh(int viTri, MayBay mb)

{

dsMayBay[viTri] = mb;

}

Phương thức DocFile được sử dụng để đọc các dữ liệu đã nạp vào trước đó để lấy thông tin các máy bay như SohieuMB, LoaiMB, SoCho, SoChuyenBay.

public static void DocFile()

{

string path = "..\\data\_MB.txt";

try

{

StreamReader sr = new StreamReader(path);

string line = sr.ReadLine();

while (line != null)

{

string[] arr = line.Split(';');

if (arr.Length == 4)

{

ThemMayBay(arr[0].Trim(), arr[1].Trim(), int.Parse(arr[2].Trim()));

}

line = sr.ReadLine();

}

sr.Close();

}

catch (Exception ex)

{

throw ex;

}

}

Phương thức LuuDanhSachMayBayVaoFile được sử dụng để lưu các dữ liệu thông tin các máy bay như SohieuMB, LoaiMB, SoCho, SoChuyenBay vào 1 file .txt.

public static void LuuDanhSachMayBayVaoFile()

{

using (StreamWriter writer = new StreamWriter("..\\data\_MB.txt"))

{

for (int i = 0; i < soLuong; i++)

{

string line = dsMayBay[i].SoHieuMB + ";" + dsMayBay[i].LoaiMB + ";" + dsMayBay[i].SoCho + ";" + dsMayBay[i].SoChuyenBay;

writer.WriteLine(line);

}

}

}

}

}

## **1.2. Hành khách**

### **1.2.1. Định nghĩa, cách hoạt động và tính chất cơ bản của cây tìm kiếm nhị phân**

Cây tìm kiếm nhị phân (Binary Search Tree) là một loại cấu trúc dữ liệu cây trong khoa học máy tính và công nghệ thông tin. Nó được sử dụng để lưu trữ và tìm kiếm các phần tử trong một tập dữ liệu mà có thể được so sánh và có thứ tự.

Trong cây tìm kiếm nhị phân, mỗi nút trong cây đại diện cho một phần tử và có tối đa hai nhánh con, được gọi là nhánh trái và nhánh phải. Giá trị của tất cả các nút trong nhánh trái của một nút cụ thể phải nhỏ hơn giá trị của nút đó, trong khi giá trị của tất cả các nút trong nhánh phải phải lớn hơn giá trị của nút đó. Điều này giúp cho việc tìm kiếm và sắp xếp dữ liệu trở nên hiệu quả hơn.

Khi thực hiện thao tác tìm kiếm trong cây tìm kiếm nhị phân, ta so sánh giá trị cần tìm với giá trị của nút hiện tại và tiếp tục tìm kiếm trong nhánh trái hoặc nhánh phải của cây dựa trên kết quả so sánh đó. Quá trình tìm kiếm được tiếp tục cho đến khi tìm thấy giá trị cần tìm hoặc không còn nút nào để duyệt.

Cây tìm kiếm nhị phân cũng cung cấp khả năng thêm, xóa và sắp xếp dữ liệu một cách hiệu quả, và được sử dụng trong nhiều ứng dụng, bao gồm cơ sở dữ liệu, các thuật toán tìm kiếm và sắp xếp, và nhiều lĩnh vực khác.

Với những đặt điểm kể trên cây tìm kiếm nhị phân phù hợp với việc dùng để lưu trữ hành khách, thêm/xoá hành khách, tìm kiếm hành khách,… trong đề tài cuối kỳ này.

### **1.2.2. Giải thuật và cấu trúc dữ liệu phần Hành Khách**

Ta khai báo các thư viện cần thiết để chương trình có thể hoạt động được.

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace FlightForm

{

Đoạn code trên định nghĩa một lớp là HanhKhach, mô tả các thuộc tính và phương thức của một hành khách trên chuyến bay.

Các thuộc tính của lớp HanhKhach bao gồm:

* CMND: Đại diện cho số CMND của khách hàng.
* Ho: Đại diện cho Họ của khách hàng
* Ten: Đại diện cho tên của khách hàng.
* Phai: Đại diện giới tính của khách hàng,
* left và right: Được sử dụng để tạo các liên kết tới các hành khách khác trong cây tìm kiếm nhị phân. Thuộc tính left chứa con trỏ đến hành khách bên trái và right chứa con trỏ đến hành khách bên phải. Nếu không có hành khách con nào, giá trị của left và right được đặt là null.

Lớp HanhKhach có hai phương thức khởi tạo:

Phương thức khởi tạo HanhKhach(string cmnd, string ho, string ten, string phai): Đây là một phương thức khởi tạo được sử dụng để tạo mới một đối tượng HanhKhach với các giá trị CMND, Họ, Tên, và Phái được cung cấp. Trong phương thức này, các thuộc tính của đối tượng HanhKhach được thiết lập với các giá trị tương ứng và các con trỏ left và right được đặt là null.

Phương thức khởi tạo HanhKhach(): Đây là một phương thức khởi tạo không tham số, được sử dụng để tạo một đối tượng HanhKhach mà các thuộc tính và con trỏ left và right được thiết lập là null.

public class HanhKhach

{

public string CMND { get; set; }

public string Ho { get; set; }

public string Ten { get; set; }

public string Phai { get; set; }

public HanhKhach left;

public HanhKhach right;

public HanhKhach(string cmnd, string ho, string ten, string phai)

{

this.CMND = cmnd;

this.Ho = ho;

this.Ten = ten;

this.Phai = phai;

this.left = null;

this.right = null;

}

public HanhKhach()

{

CMND = null;

Ho = null;

Ten = null;

Phai = null;

left = null;

right = null;

}

}

Đoạn code dưới thực hiện 1 lớp DanhSachHanhKhach với các chức năng như: thêm hành khách, xoá hành khách, tìm kiếm hành khách, chỉnh sửa thông tin hành khách và ghi dữ liệu hành khách vào file data.

public class DanhSachHanhKhach

{

public static HanhKhach root;

public DanhSachHanhKhach()

{

root = null;

}

public static void ThemHK(HanhKhach hk)

{

Nếu root (gốc của cây) là null, tức là cây đang rỗng, ta gán root bằng đối tượng HanhKhach mới hk.

Ngược lại, nếu cây đã có các nút, ta di chuyển tiếp để tìm vị trí thích hợp để chèn hk vào.

if (root == null)

{

root = hk;

}

Tạo các nút nodeRun và nodeTruoc và sử dụng một vòng lặp để tìm vị trí chèn:

Ban đầu, nodeRun và nodeTruoc đều được gán bằng root.

Tiếp tục lặp trong khi nodeRun khác null.

Trong vòng lặp, nodeTruoc sẽ lưu trữ nút trước đó và nodeRun sẽ di chuyển đến con trái hoặc con phải tùy thuộc vào giá trị của hk.CMND so với giá trị của nodeRun.CMND.

* Nếu hk.CMND nhỏ hơn nodeRun.CMND, nodeRun được cập nhật thành con trái của nodeRun.
* Nếu hk.CMND lớn hơn nodeRun.CMND, nodeRun được cập nhật thành con phải của nodeRun.
* Nếu hk.CMND bằng nodeRun.CMND, tức là hành khách đã tồn tại trong cây, ta kết thúc phương thức.

else

{

HanhKhach nodeRun = root;

HanhKhach nodeTruoc = root;

while (nodeRun != null)

{

nodeTruoc = nodeRun;

if (hk.CMND.CompareTo(nodeRun.CMND) < 0)

{

nodeRun = nodeRun.left;

}

else if (hk.CMND.CompareTo(nodeRun.CMND) > 0)

{

nodeRun = nodeRun.right;

}

else if (hk.CMND.CompareTo(nodeRun.CMND) == 0)

{

return;

}

}

Sau khi thoát khỏi vòng lặp, ta kiểm tra giá trị của hk.CMND so với nodeTruoc.CMND để xác định vị trí chèn:

* Nếu hk.CMND lớn hơn nodeTruoc.CMND, ta chèn hk làm con phải của nodeTruoc.
* Ngược lại, ta chèn hk làm con trái của nodeTruoc.

if (hk.CMND.CompareTo(nodeTruoc.CMND) > 0)

{

nodeTruoc.right = hk;

}

else nodeTruoc.left = hk;

}

return;

}

public static bool delete(string CMND)

{

Tạo hai nút p và pp và sử dụng một vòng lặp để tìm nút cần xóa:

- Ban đầu, p được gán bằng root, tức là nút gốc của cây.

- pp được khởi tạo là null.

- Trong quá trình lặp, kiểm tra p có giá trị CMND bằng với CMND cần xóa hay không:

* Nếu không bằng, cập nhật pp thành p, sau đó di chuyển p sang nút con trái hoặc con phải dựa trên giá trị của CMND so với p.CMND.
* Tiếp tục lặp cho đến khi p trở thành null hoặc tìm được nút có giá trị CMND bằng CMND cần xóa.

HanhKhach p = root;

HanhKhach pp = null;

while (p != null && p.CMND != CMND)

{

pp = p;

if (CMND.CompareTo(p.CMND) < 0)

p = p.left;

else

p = p.right;

}

Nếu p bằng null, tức là không tìm thấy nút có giá trị CMND bằng CMND, phương thức trả về false để chỉ ra rằng việc xóa không thành công

if (p == null)

return false;

Nếu nút cần xóa có cả left và right:

- Tìm nút s là nút kế tiếp lớn nhất trong cây left của p (nút s là nút có giá trị CMND lớn nhất nhỏ hơn giá trị CMND của p).

- Gán giá trị CMND của s cho p.

- Cập nhật p và pp thành s và ps (nút cha của s).

if (p.left != null && p.right != null)

{

HanhKhach s = p.left;

HanhKhach ps = p;

while (s.right != null)

{

ps = s;

s = s.right;

}

p.CMND = s.CMND;

p = s;

pp = ps;

}

- Gán c thành left hoặc right của p (nếu có).

- Kiểm tra xem nút cần xóa có phải là nút gốc (root) không:

* Nếu là nút gốc, gán root thành null.
* Ngược lại, cập nhật left hoặc right của pp thành c tùy thuộc vào vị trí của p.

HanhKhach c = null;

if (p.left != null)

c = p.left;

else

c = p.right;

if (p == root)

root = null;

else

{

if (p == pp.left)

pp.left = c;

else

pp.right = c;

}

return true;

}

public static HanhKhach search(string cmnd)

{

Phương thức search được sử dụng để tìm kiếm một hành khách dựa trên số chứng minh nhân dân (CMND) được cung cấp.

Nếu số CMND không tôn tại kết thúc quá trình tìm kiếm.

if (!isItExsit(root, cmnd))

{

return null;

}

Sau khi kiểm tra xem hành khách có tồn tại trong cây hay không bằng cách sử dụng phương thức "isItExsit", phương thức sử dụng một vòng lặp while để đi qua các nút của cây theo thứ tự từ gốc đến lá. Nếu số CMND của hành khách được tìm thấy bằng số CMND của nút hiện tại, nút đó sẽ trả về. Nếu không, nút hiện tại được cập nhật để tiếp tục tìm kiếm trên cây con bên trái hoặc bên phải của nút hiện tại, cho đến khi tìm thấy hoặc đi đến một nút null. Nếu không tìm thấy, phương thức trả về null.

HanhKhach NodeRun = root;

while (NodeRun != null && (cmnd.CompareTo(NodeRun.CMND) != 0))

{

if (cmnd.CompareTo(NodeRun.CMND) > 0)

NodeRun = NodeRun.right;

else

NodeRun = NodeRun.left;

}

return NodeRun;

}

Phương thức "isItExsit" thực hiện việc kiểm tra xem một nút cụ thể của cây có chứa hành khách có số CMND tương ứng hay không. Nếu tìm thấy, phương thức trả về true, nếu không, phương thức đệ quy gọi lại chính nó trên nút con bên trái hoặc bên phải của nút hiện tại cho đến khi tìm thấy nút chứa hành khách có số CMND tương ứng hoặc cho đến khi nút hiện tại là null.

public static bool isItExsit(HanhKhach temproot, string cmnd)

{

if (temproot != null)

{

if (cmnd.CompareTo(temproot.CMND) == 0)

return true;

else if (cmnd.CompareTo(temproot.CMND) < 0)

return isItExsit(temproot.left, cmnd);

else if (cmnd.CompareTo(temproot.CMND) > 0)

return isItExsit(temproot.right, cmnd);

}

return false;

}

Phương thức DocFileHK được sử dụng để đọc các dữ liệu đã nạp vào trước đó để lấy thông tin các khách hàng như Số CMND, họ, tên, phái của khách hàng.

public static void DocFileHK()

{

string path = "..\\data\_HK.txt";

try

{

StreamReader sr = new StreamReader(path);

string line = sr.ReadLine();

while (line != null)

{

string[] arr = line.Split(';');

if (arr.Length == 4)

{

HanhKhach hk = new HanhKhach(arr[0].Trim(), arr[1].Trim(), arr[2].Trim(), arr[3].Trim());

ThemHK(hk);

}

line = sr.ReadLine();

}

sr.Close();

}

catch (Exception ex)

{

throw ex;

}

}

Phương thức LuuDanhSachHanhKhachVaoFile được sử dụng để lưu các dữ liệu thông tin khách hàng như số CMND, Họ, Tên, Phái của khách hàng vào 1 file .txt.

public static void LuuDanhSachHanhKhachVaoFile(HanhKhach root)

{

using (StreamWriter writer = new StreamWriter("..\\data\_HK.txt"))

{

HanhKhach current = root;

while (current != null)

{

if (current.left == null)

{

string CMND = current.CMND.Trim();

string Ho = current.Ho.Trim();

string Ten = current.Ten.Trim();

string Phai = current.Phai.Trim();

string line = CMND + ";" + Ho + ";" + Ten + ";" + Phai;

writer.WriteLine(line);

current = current.right;

}

else

{

HanhKhach prev = current.left;

while (prev.right != null && prev.right != current)

{

prev = prev.right;

}

if (prev.right == null)

{

prev.right = current;

current = current.left;

}

else

{

string CMND = current.CMND.Trim();

string Ho = current.Ho.Trim();

string Ten = current.Ten.Trim();

string Phai = current.Phai.Trim();

string line = CMND + ";" + Ho + ";" + Ten + ";" + Phai;

writer.WriteLine(line);

prev.right = null;

current = current.right;

}

}

}

}

}

}

}

## **1.3 Chuyến Bay**

### **1.3.1. Định nghĩa, cách hoạt động và tính chất cơ bản của danh sách liên kết đơn**

Danh sách liên kết đơn là một cấu trúc dữ liệu sử dụng trong lập trình để lưu trữ và truy xuất dữ liệu theo thứ tự. Nó được gọi là "liên kết" vì nó bao gồm một chuỗi các đối tượng dữ liệu, gọi là các "nút", mỗi nút kết nối với nút tiếp theo trong danh sách thông qua một liên kết.

Mỗi nút trong danh sách liên kết đơn bao gồm hai thành phần chính: dữ liệu và một liên kết tới nút tiếp theo. Thông qua liên kết này, ta có thể truy xuất các phần tử trong danh sách bằng cách đi từ nút đầu tiên (gọi là "nút gốc" hoặc "head") và theo dấu liên kết cho đến khi đạt được phần tử mong muốn hoặc hết danh sách.

Ưu điểm của danh sách liên kết đơn là khả năng chèn và xóa phần tử ở bất kỳ vị trí nào trong danh sách một cách hiệu quả, đối với danh sách có số lượng phần tử thay đổi thường xuyên. Tuy nhiên, điểm yếu của danh sách liên kết đơn là thời gian truy cập ngẫu nhiên chậm hơn so với các cấu trúc dữ liệu như mảng, vì ta phải đi qua các liên kết từ đầu danh sách để đạt được phần tử mong muốn.

### **1.3.2. Giải thuật và cấu trúc dữ liệu phần Chuyến bay**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace FlightForm

{

public class ChuyenBay

{

public string MaCB { get; set; }

public DateTime NgayGioKhoiHanh { get; set; }

public string SanBayDen { get; set; }

public int TrangThai { get; set; }

public string SoHieuMB { get; set; }

public List<string> DanhSachVe;

public int SoVe { get; set; }

public int SoVeDaDat { get; set; }

public ChuyenBay next;

//khoi tao

public ChuyenBay(string maCB, DateTime ngayGioKhoiHanh, string sanBayDen, string soHieuMB)

{

this.MaCB = maCB;

this.NgayGioKhoiHanh = ngayGioKhoiHanh;

this.SanBayDen = sanBayDen;

this.TrangThai = 1;

this.SoHieuMB = soHieuMB;

this.SoVe = DanhSachMayBay.dsMayBay[DanhSachMayBay.TimMayBay(soHieuMB)].SoCho;

this.DanhSachVe = new List<string>(Enumerable.Repeat("", this.SoVe));

this.SoVeDaDat = 0;

this.next = null;

}

public ChuyenBay(string maCB, DateTime ngayGioKhoiHanh, string sanBayDen, int trangThai, string soHieuMB)

{

this.MaCB = maCB;

this.NgayGioKhoiHanh = ngayGioKhoiHanh;

this.SanBayDen = sanBayDen;

this.TrangThai = trangThai;

this.SoHieuMB = soHieuMB;

this.SoVe = DanhSachMayBay.dsMayBay[DanhSachMayBay.TimMayBay(soHieuMB)].SoCho;

this.DanhSachVe = new List<string>(Enumerable.Repeat("", this.SoVe));

this.SoVeDaDat = 0;

this.next = null;

}

public static bool DatVe(HanhKhach hk, int viTri, ChuyenBay cb)

{

for (int i = 0; i < cb.SoVe; i++)

{

if (cb.DanhSachVe.Contains(hk.CMND))

{

//Console.WriteLine("Hanh khach da ton tai.");

return false;

}

}

if (cb.DanhSachVe[viTri] == "")

{

cb.DanhSachVe[viTri] = hk.CMND;

cb.SoVeDaDat++;

return true;

//Console.WriteLine("Hanh khach da ton tai.");

}

return false;

}

}

public class DanhSachChuyenBay

{

public static ChuyenBay head;

public static ChuyenBay tail;

public static int size;

public DanhSachChuyenBay()

{

head = null;

tail = null;

size = 0;

}

public static int length()

{

return size;

}

public static bool isEmpty()

{

return size == 0;

}

public static void ThemCB(ChuyenBay cb)

{

ChuyenBay newest = cb;

if (isEmpty())

head = newest;

else

tail.next = newest;

tail = newest;

size = size + 1;

}

public static ChuyenBay TimMaCB(string MaCB)

{

ChuyenBay tmp = head;

while (tmp != null)

{

if (tmp.MaCB == MaCB)

return tmp;

tmp = tmp.next;

}

return null;

}

public static bool HuyChuyen(ChuyenBay temp)

{

ChuyenBay tmp = head;

while (tmp != null)

{

if (tmp.MaCB == temp.MaCB)

{

if (temp.TrangThai == Define.CONVE || temp.TrangThai == Define.HETVE)

{

temp.TrangThai = Define.HUYCHUYEN;

size--;

return true;

}

}

tmp = tmp.next;

}

return false;

}

public static bool ChinhNgayGio(ChuyenBay temp, DateTime a)

{

ChuyenBay tmp = head;

while (tmp != null)

{

if (tmp.MaCB == temp.MaCB)

{

temp.NgayGioKhoiHanh = a;

return true;

}

tmp = tmp.next;

}

return false;

}

public static void DatVe(string MaCB,int viTri, string CMND)

{

ChuyenBay tmp = head;

while (tmp != null)

{

if (tmp.MaCB == MaCB && tmp.SoVe > viTri)

{

tmp.DanhSachVe[viTri] = CMND;

tmp.SoVeDaDat++;

}

tmp = tmp.next;

}

}

public static bool DaDuKhach(string MaCB)

{

ChuyenBay tmp = head;

while (tmp != null)

{

if (tmp.MaCB == MaCB)

{

if (tmp.SoVeDaDat < tmp.SoVe) return false;

else

{

tmp.TrangThai = Define.HETVE;

return true;

}

}

tmp = tmp.next;

}

return false;

}

public static int SoVeChuaBan(string MaCB)

{

ChuyenBay tmp = head;

while (tmp != null)

{

if (tmp.MaCB == MaCB)

{

break;

}

tmp = tmp.next;

}

return tmp.SoVe - tmp.SoVeDaDat;

}

public static void checkHoanTat(ChuyenBay temp)

{

if (temp.TrangThai == Define.HETVE || temp.TrangThai == Define.CONVE)

{

if (temp.NgayGioKhoiHanh <= DateTime.Now)

{

temp.TrangThai = Define.HOANTAT;

DanhSachMayBay.dsMayBay[DanhSachMayBay.TimMayBay(temp.SoHieuMB)].SoChuyenBay++;

}

}

}

public static void checkHoanTatAll()

{

ChuyenBay tmp = head;

while (tmp != null)

{

checkHoanTat(tmp);

tmp = tmp.next;

}

}

public static void checkHoanTat2(ChuyenBay temp)

{

if (temp.TrangThai == Define.HOANTAT)

{

DanhSachMayBay.dsMayBay[DanhSachMayBay.TimMayBay(temp.SoHieuMB)].SoChuyenBay++;

}

}

public static void checkHoanTatAll2()

{

ChuyenBay tmp = head;

while (tmp != null)

{

checkHoanTat2(tmp);

tmp = tmp.next;

}

}

public static void LuuDanhSachChuyenBayVaoFile(ChuyenBay head)

{

using (StreamWriter writer = new StreamWriter("..\\data\_CB.txt"))

{

ChuyenBay temp = head;

while (temp != null)

{

string IDCB = temp.MaCB.Trim();

string Time = temp.NgayGioKhoiHanh.ToString("dd/MM/yyyy hh:mm:ss tt");

string Desti = temp.SanBayDen.Trim();

string Type = temp.TrangThai.ToString().Trim();

string IDMB = temp.SoHieuMB.Trim();

string line = IDCB + ";" + Time + ";" + Desti + ";" + Type + ";" + IDMB;

writer.WriteLine(line);

temp = temp.next;

}

}

}

public static void LuuDanhSachDatVeChuyenBayVaoFile(ChuyenBay head)

{

using (StreamWriter writer = new StreamWriter("..\\data\_DSV.txt"))

{

ChuyenBay temp = head;

while (temp != null)

{

string IDCB = temp.MaCB.Trim();

string line = IDCB + ";";

for (int i = 0; i < temp.DanhSachVe.Count; i++)

{

if (temp.DanhSachVe[i] != "") line += i.ToString() + ";" + temp.DanhSachVe[i] + ";";

}

writer.WriteLine(line);

temp = temp.next;

}

}

}

}

}

Đoạn code trên triển khai các lớp và phương thức để quản lý thông tin chuyến bay và đặt vé của hành khách.

Cấu trúc dữ liệu:

- Lớp **ChuyenBay**: bao gồm thông tin của một chuyến bay, bao gồm: mã chuyến bay (MaCB), ngày giờ khởi hành (NgayGioKhoiHanh), sân bay đến (SanBayDen), trạng thái chuyến bay (TrangThai), số hiệu máy bay (SoHieuMB), danh sách vé của chuyến bay (DanhSachVe), số vé của chuyến bay (SoVe), số vé đã đặt của chuyến bay (SoVeDaDat) và con trỏ next.

- Lớp **DanhSachChuyenBay**: sử dụng cấu trúc dữ liệu danh sách liên kết đơn, bao gồm các phương thức để thêm chuyến bay, tìm kiếm chuyến bay bằng mã chuyến bay, hủy chuyến bay, cập nhật ngày giờ khởi hành của chuyến bay, và đặt vé cho hành khách.

Giải thuật:

- Phương thức **DatVe**: Kiểm tra nếu hành khách đã đặt vé trên chuyến bay này thì trả về false, ngược lại kiểm tra nếu vị trí đặt vé trống thì đặt vé cho hành khách và cập nhật số vé đã đặt của chuyến bay, sau đó trả về true.

- Phương thức **ThemCB**: Thêm một chuyến bay vào danh sách chuyến bay sử dụng chức năng AddLast của danh sách liên kết đơn.

- Phương thức **TimMaCB**:Tìm kiếm chuyến bay theo mã chuyến bay và trả về đối tượng chuyến bay đó.

- Phương thức **HuyChuyen**: Hủy chuyến bay chỉ định và cập nhật trạng thái của chuyến bay đó thành "Hủy chuyến" nếu chuyến bay đó đang ở trạng thái "Còn vé" hoặc "Hết vé".

- Phương thức **ChinhNgayGio**: Cập nhật ngày giờ khởi hành của chuyến bay chỉ định thành ngày giờ mới.

- Phương thức **DatVe**: Đặt vé cho hành khách với số CMND chỉ định trên vị trí đặt vé chỉ định của chuyến bay có mã chuyến bay chỉ định.

- Phương thức **DaDuKhach**: Kiểm tra xem số vé đã đặt của chuyến bay có mã chỉ định có bằng số vé của chuyến bay đó không. Nếu bằng trả về true, ngược lại trả về false.

**Code In\_DSCB:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FlightForm

{

public partial class In\_DSCB : Form

{

public string Ngay { get; set; }

public string Thang { get; set; }

public string Nam { get; set; }

public string Place { get; set; }

public ChuyenBay cb;

public In\_DSCB()

{

InitializeComponent();

}

void CreateColumnFordataGridView6()

{

var col\_IDCB = new DataGridViewTextBoxColumn();

var col\_SVCT = new DataGridViewTextBoxColumn();

col\_IDCB.HeaderText = "Mã Chuyến Bay";

col\_SVCT.HeaderText = "Số Vé Còn Trống";

col\_IDCB.DataPropertyName = "IDCB";

col\_SVCT.DataPropertyName = "SoVeConTrong";

col\_IDCB.Width = 300;

col\_SVCT.Width = 300;

dtgvDSCB.Columns.AddRange(new DataGridViewColumn[] { col\_IDCB, col\_SVCT });

}

void LoadDSCB()

{

dtgvDSCB.Rows.Clear();

lbDate.Text = Ngay + "/" + Thang + "/" + Nam;

lbPlace.Text = Place;

string date = Ngay + "/" + Thang + "/" + Nam;

string place = Place;

ChuyenBay tmp = DanhSachChuyenBay.head;

while (tmp != null)

{

if (tmp.NgayGioKhoiHanh.Date.ToString("dd/MM/yyyy") == date && tmp.SanBayDen == place)

{

int SVCT = 0;

for (int i = 0; i <tmp.SoVe;i++)

{

if (tmp.DanhSachVe[i] == "")

SVCT++;

}

dtgvDSCB.Rows.Add(tmp.MaCB,SVCT);

}

tmp = tmp.next;

}

}

private void In\_DSCB\_Load(object sender, EventArgs e)

{

CreateColumnFordataGridView6();

LoadDSCB();

}

Đoạn code trên triển khai các lớp và phương thức để hiển thị danh sách chuyến bay tại một địa điểm cụ thể và vào một ngày cụ thể

Cấu trúc dữ liệu:

- Lớp **In\_DSCB** kế thừa từ lớp Form của Windows Forms để tạo ra một giao diện người dùng.

- Các thuộc tính Ngay,Thang,Nam, và Place được sử dụng để lưu trữ thông tin ngày, tháng, năm và địa điểm cần tìm kiếm chuyến bay  
- Lớp **ChuyenBay** được sử dụng để lưu trữ thông tin của một chuyến bay

- **DanhSachChuyenBay** là một danh sách liên kết đơn chứa tất cả các chuyến bay

Giải thuật:

- Hàm **CreateColumnFordataGridView6()** để tạo ra các cột cho đối tượng DataGridView.

- Hàm **LoadDSCB()** để tải danh sách các chuyến bay còn trống vào DataGridView. Hàm này sẽ xóa tất cả các hàng của DataGridView, sau đó lấy thông tin ngày, tháng, năm và địa điểm từ các thuộc tính Ngay, Thang, Nam và Place. Sau đó, nó sẽ duyệt danh sách liên kết của tất cả các chuyến bay trong DanhSachChuyenBay để tìm các chuyến bay có ngày giờ khởi hành và địa điểm đến khớp với thông tin được lưu trữ trong các thuộc tính Ngay, Thang, Nam và Place. Đối với mỗi chuyến bay tìm thấy, nó sẽ đếm số lượng vé còn trống bằng cách kiểm tra danh sách vé và đếm số lượng vé không được đặt. Sau đó, nó thêm một hàng mới vào DataGridView với thông tin về mã chuyến bay và số lượng vé còn trống.

- Hàm **In\_DSCB\_Load()** được gọi khi form được tải lên và gọi hai hàm **CreateColumnFordataGridView6()** và **LoadDSCB()** để khởi tạo DataGridView và hiển thị danh sách chuyến bay còn trống.

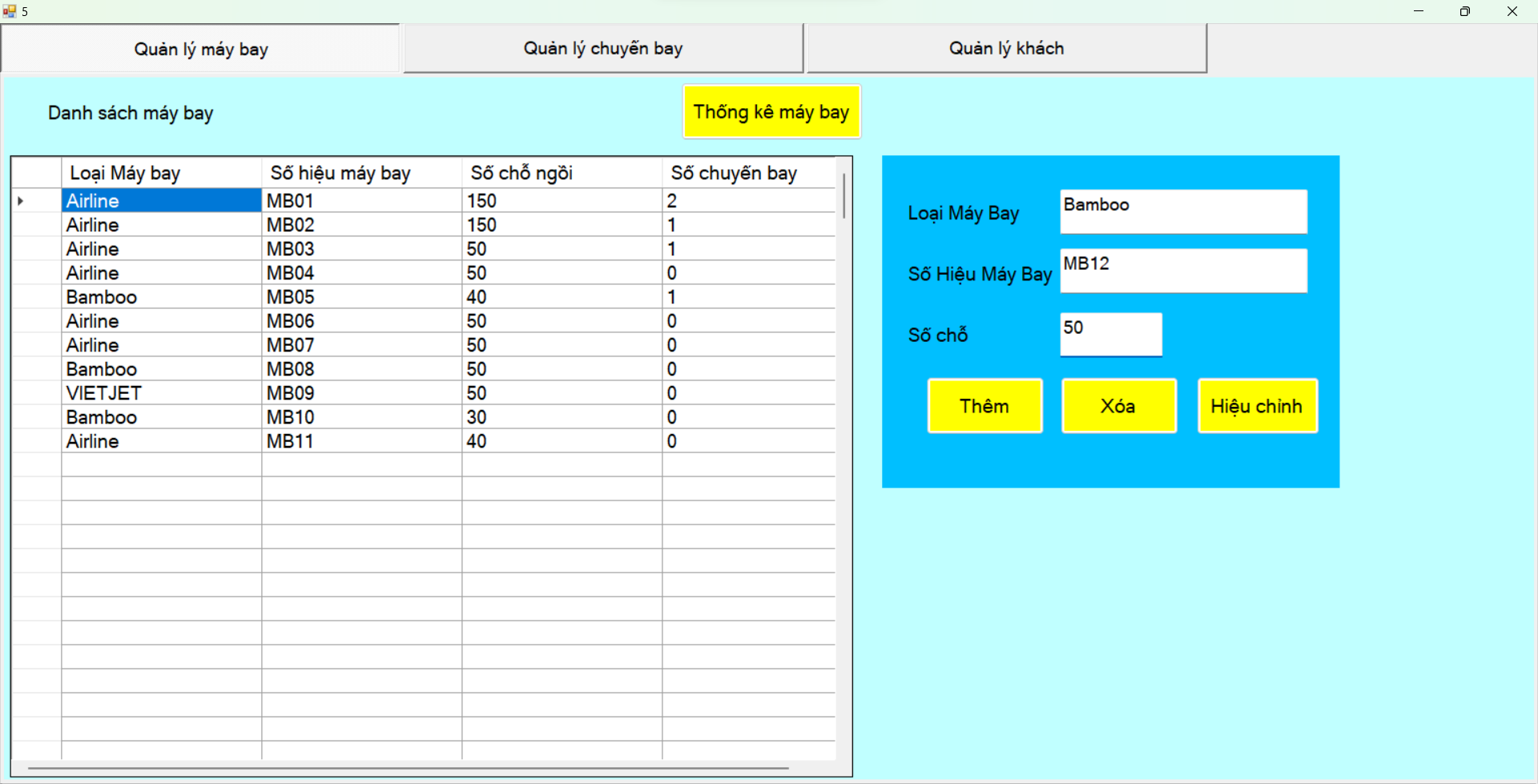
# **CHƯƠNG 2: TỔNG QUÁT HOẠT ĐỘNG VÀ CHỨC NĂNG CỦA HỆ THỐNG**

## **2.1. Kết quả**

Video kết quả: <https://youtu.be/Fd4Q8-kLFQo>

### **2.1.1. Cập nhật danh sách các máy bay ( thêm/ xóa/hiệu chỉnh )**

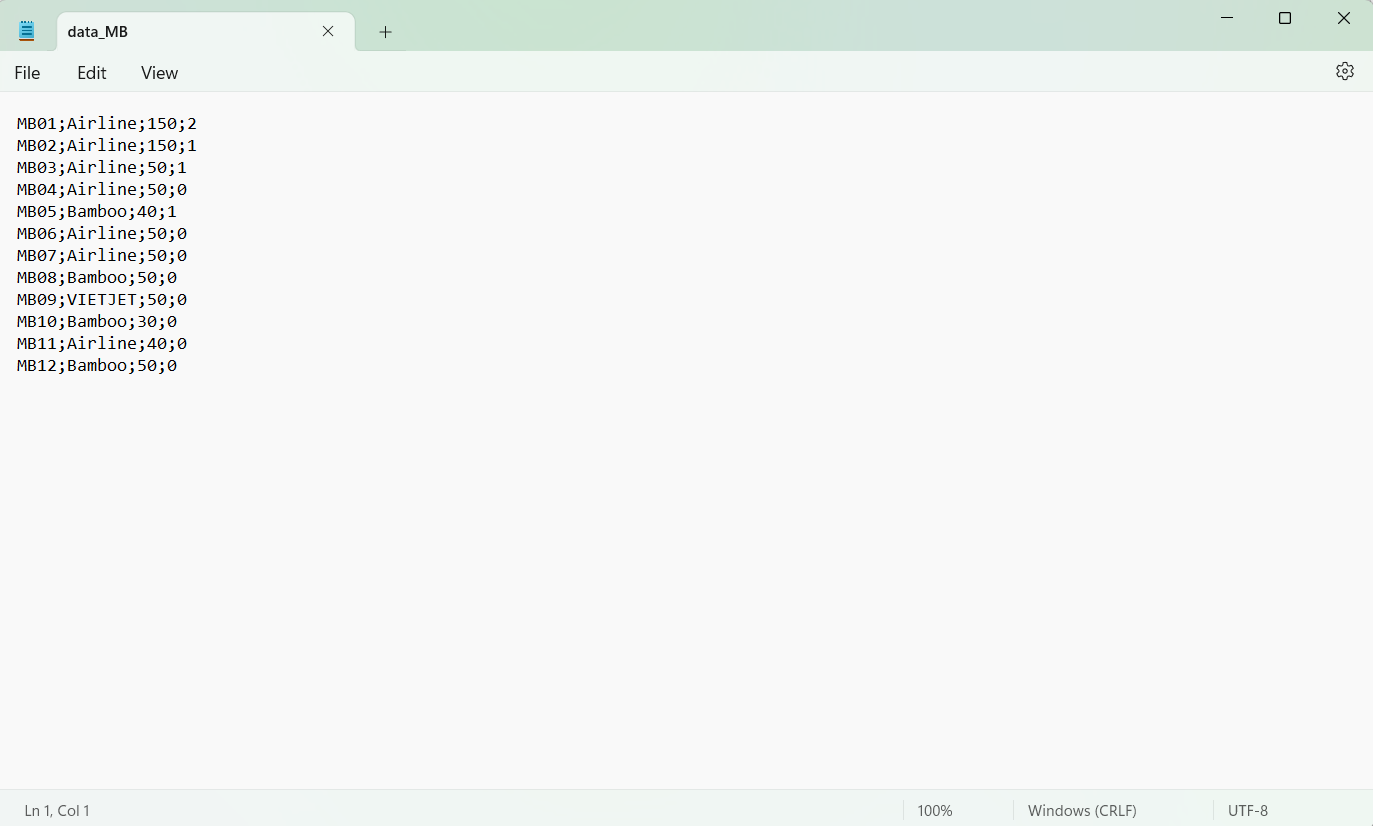
* **Thêm một máy bay vào danh sách hiện tại**:
  + Loại máy bay : Bamboo
  + Số hiệu máy bay : MB12
  + Số chỗ : 50



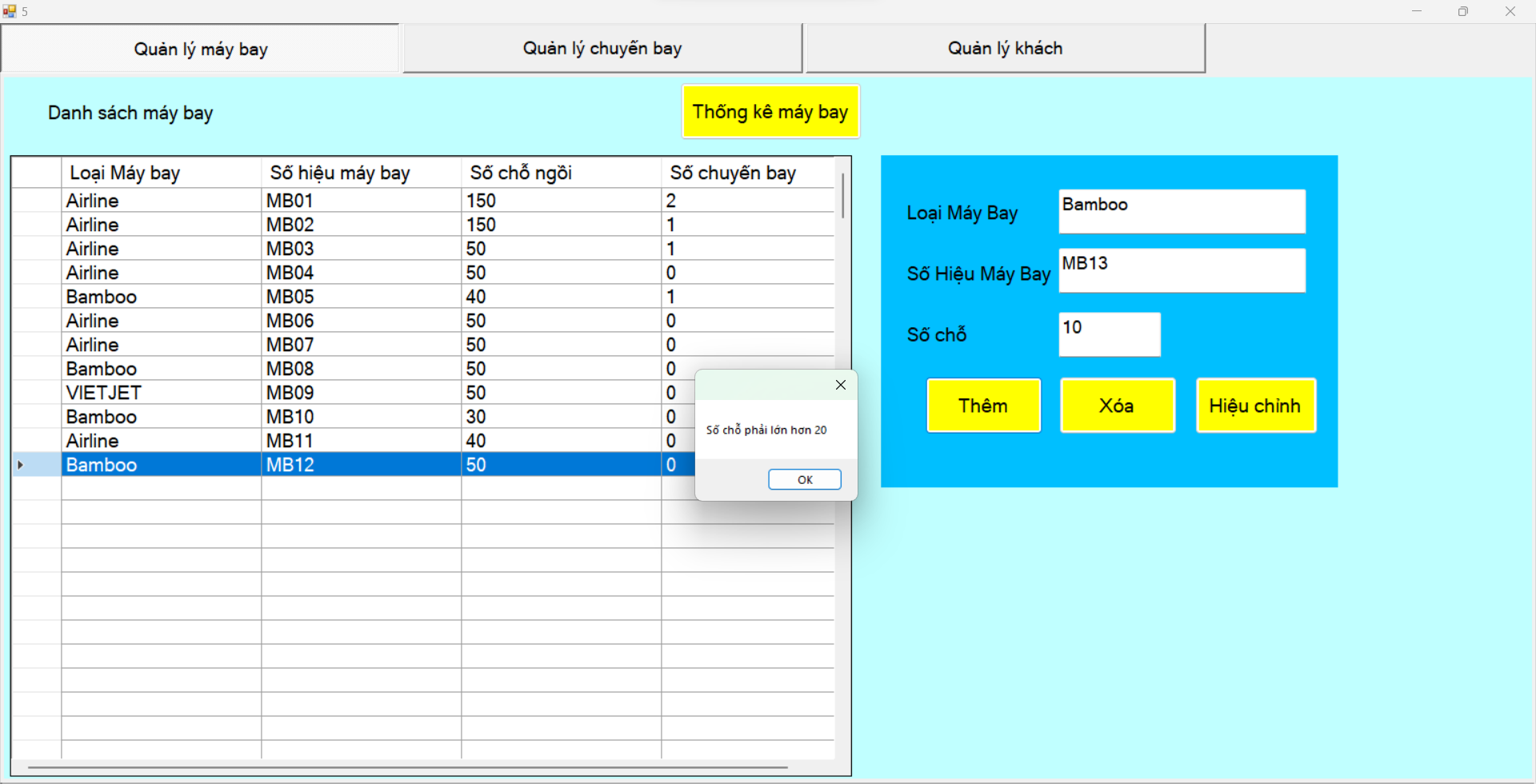


Kết quả ta có được là máy bay sẽ được cập nhật vào danh sách và thống kê máy bay:

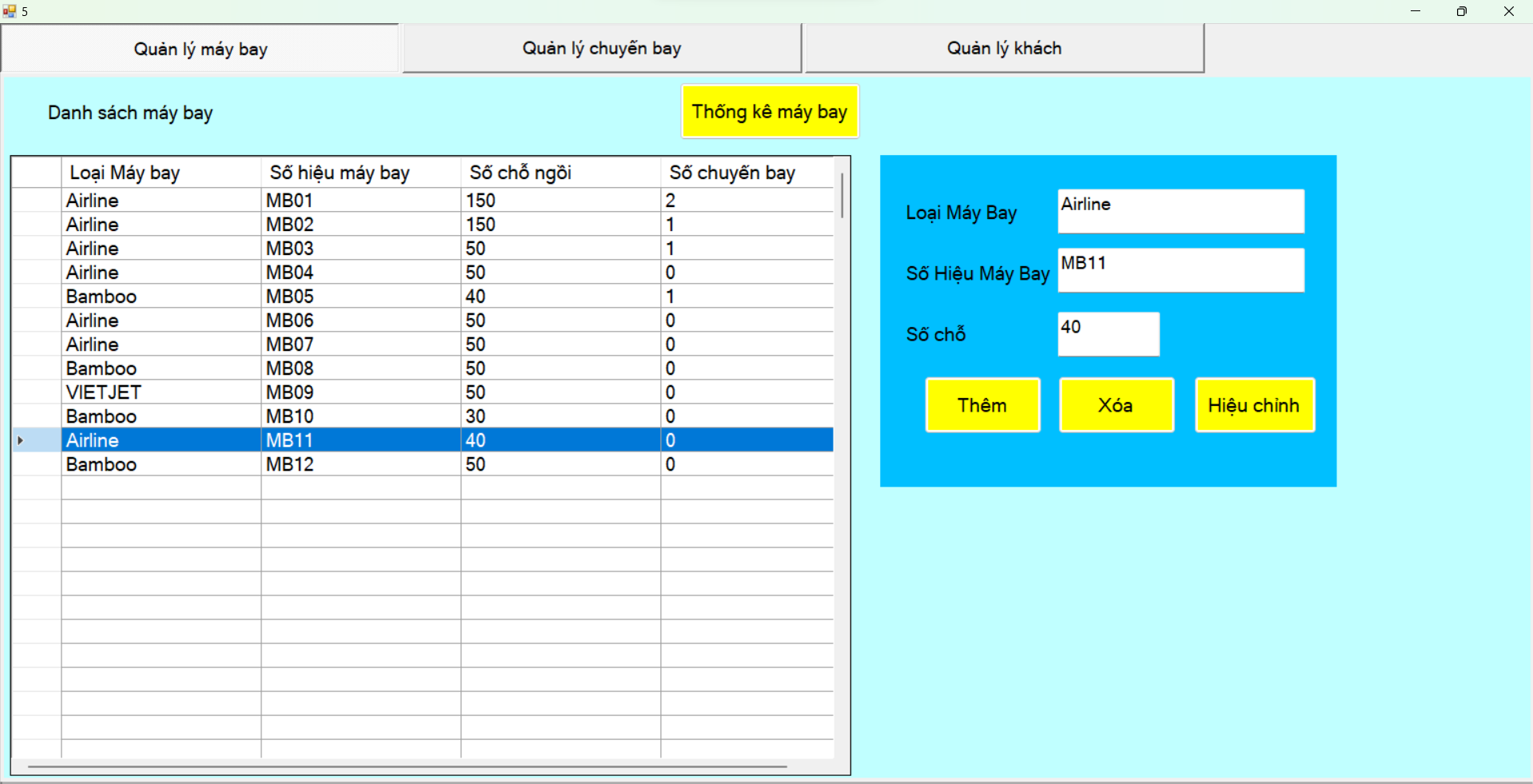




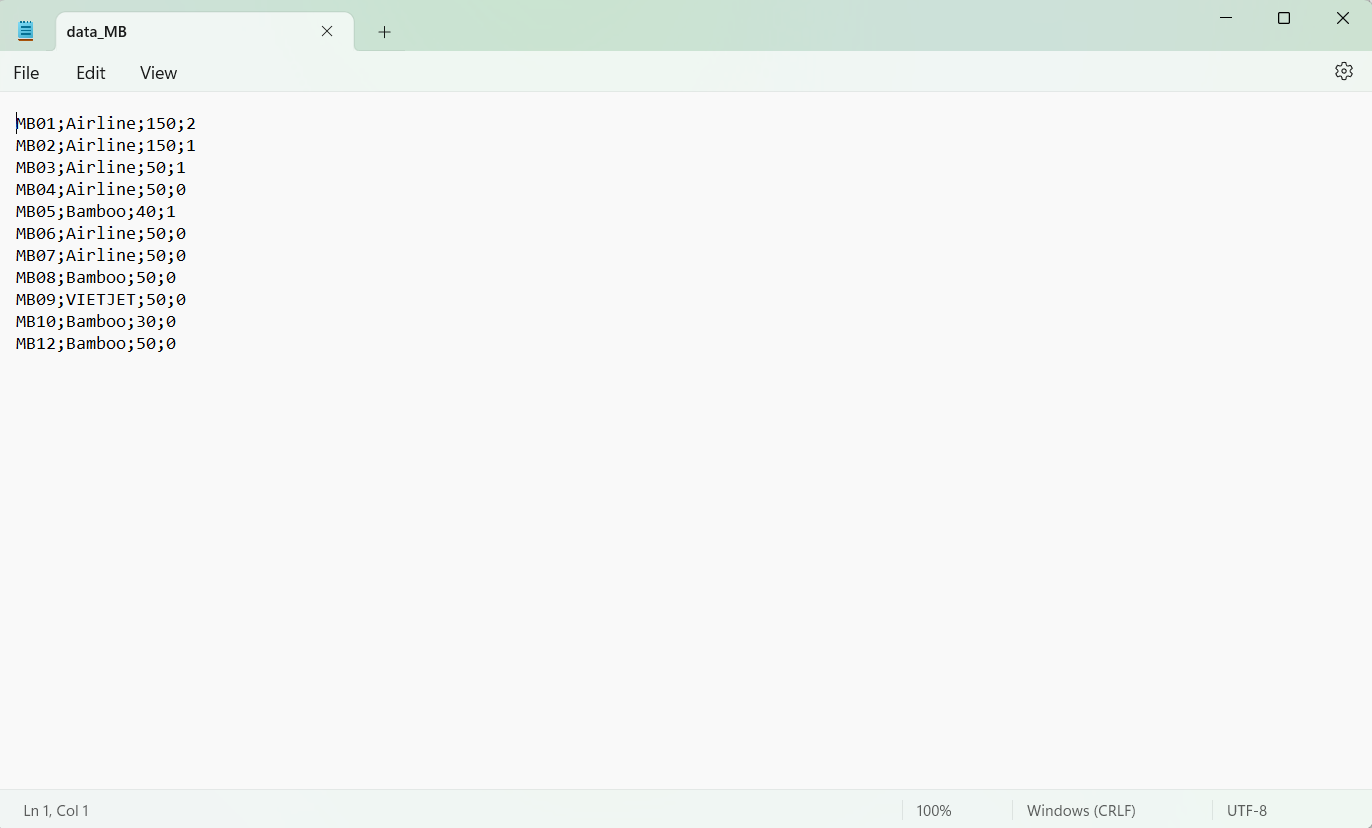
Số chỗ trên máy bay bắt buộc phải >=20 , nếu không sẽ không tạo được máy bay :



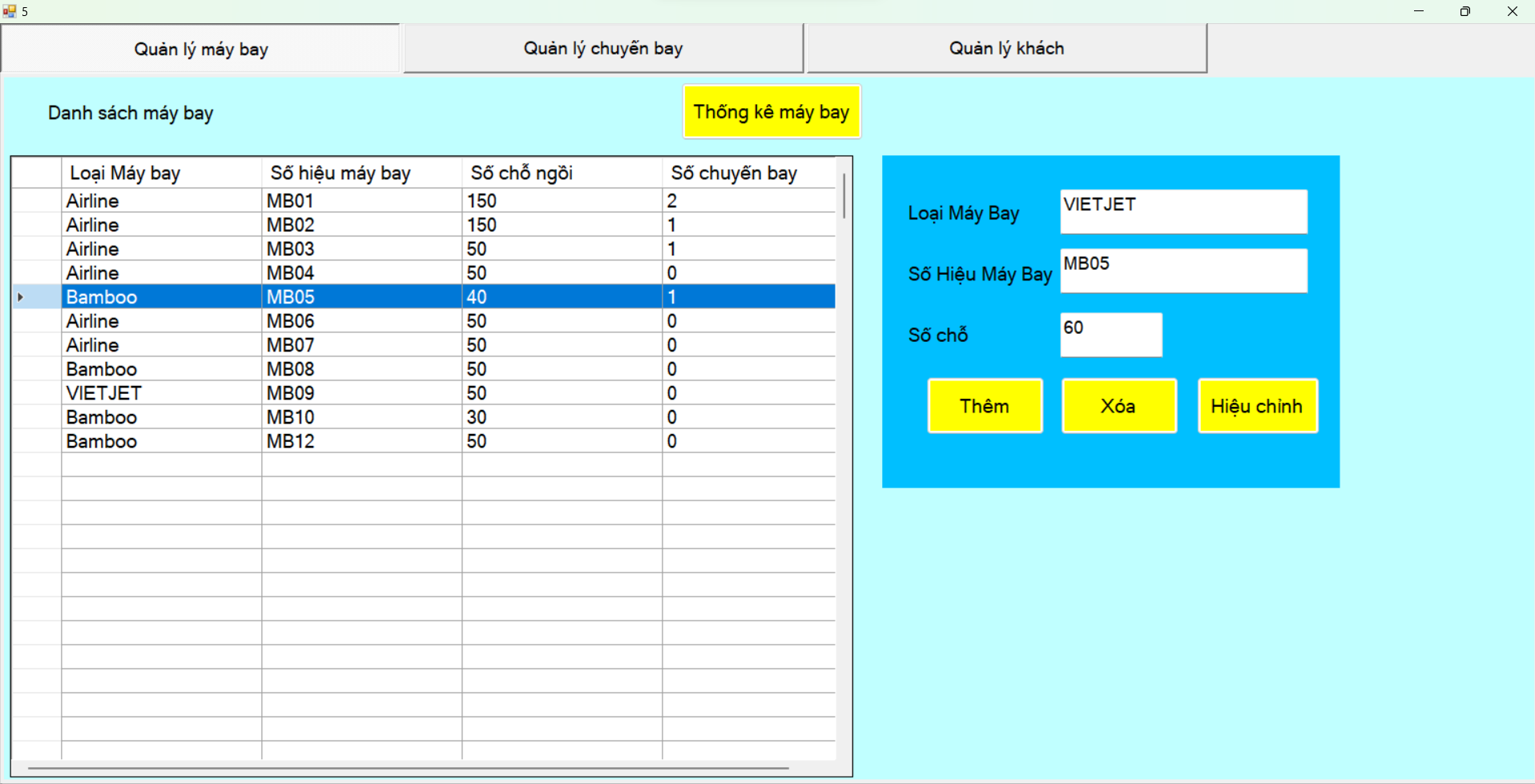
* **Xóa một máy bay**: Xóa máy bay số hiệu MB07 khỏi danh sách và bảng thống kê ta có được kết quả như sau :

****

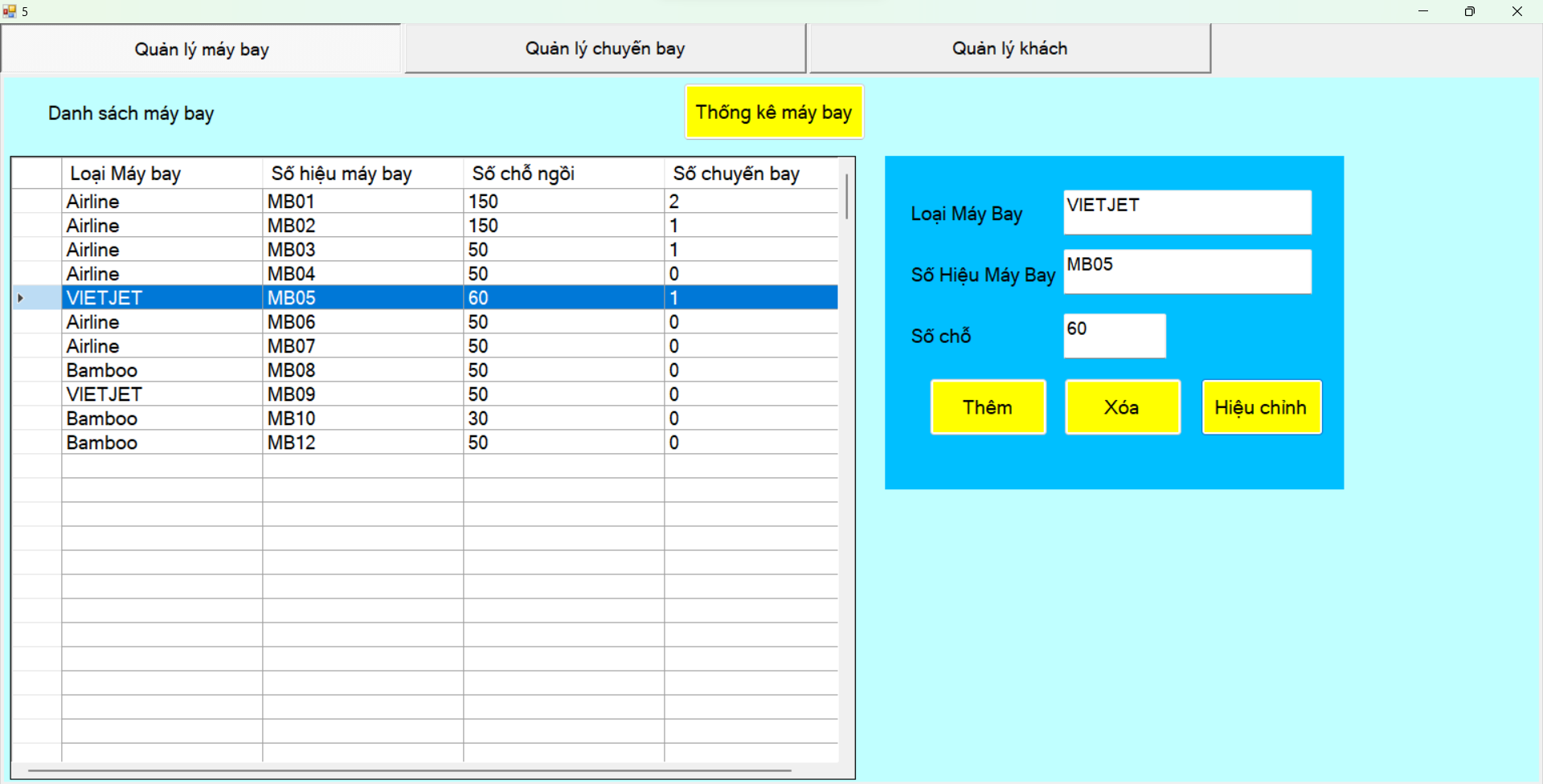
****

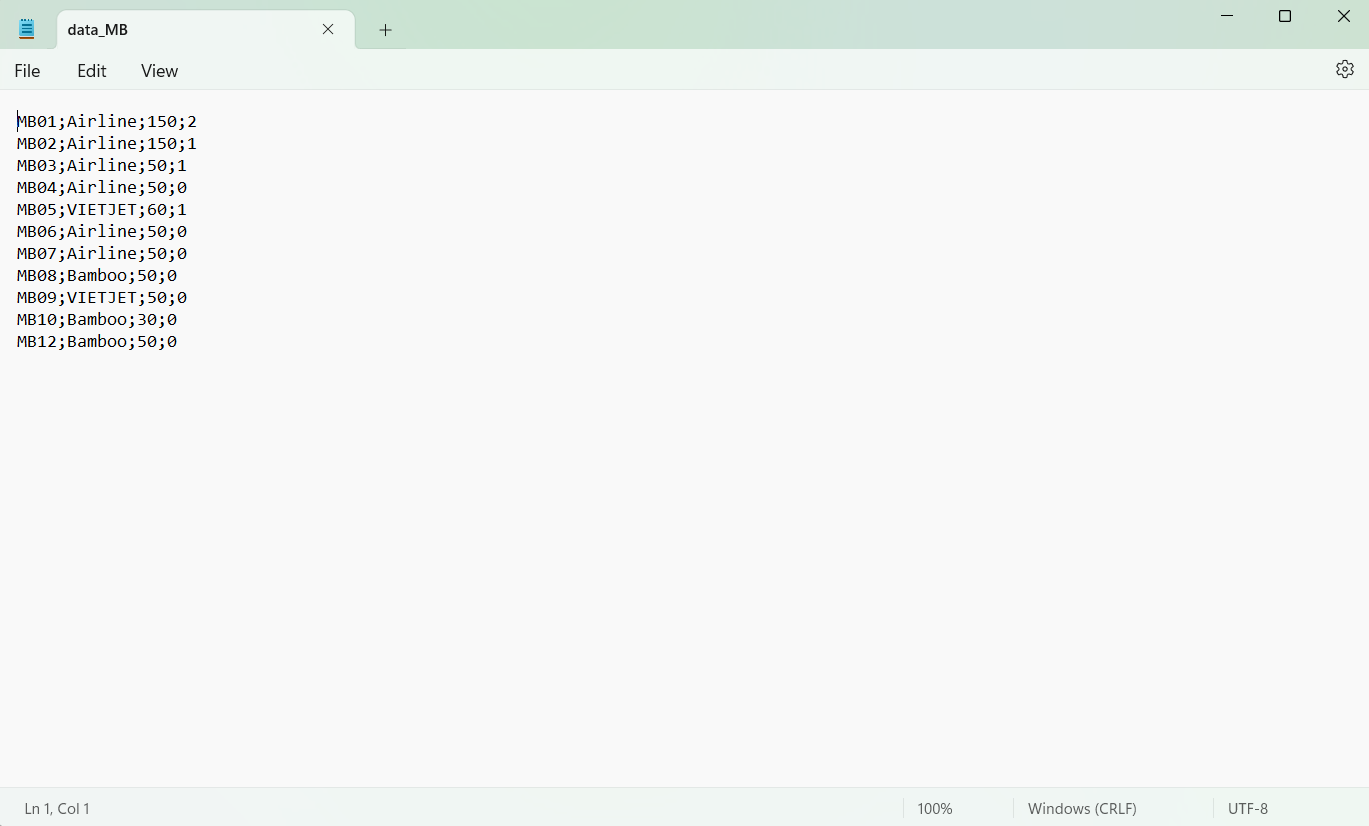
****

* **Hiệu chỉnh một máy bay** : Chỉnh sửa máy bay số hiệu MB05 thành loại máy bay VIETJET và tăng số chỗ lên 60

****

Kết quả sau khi hiệu chỉnh sẽ được cập nhật vào danh sách máy bay trong giao diện:

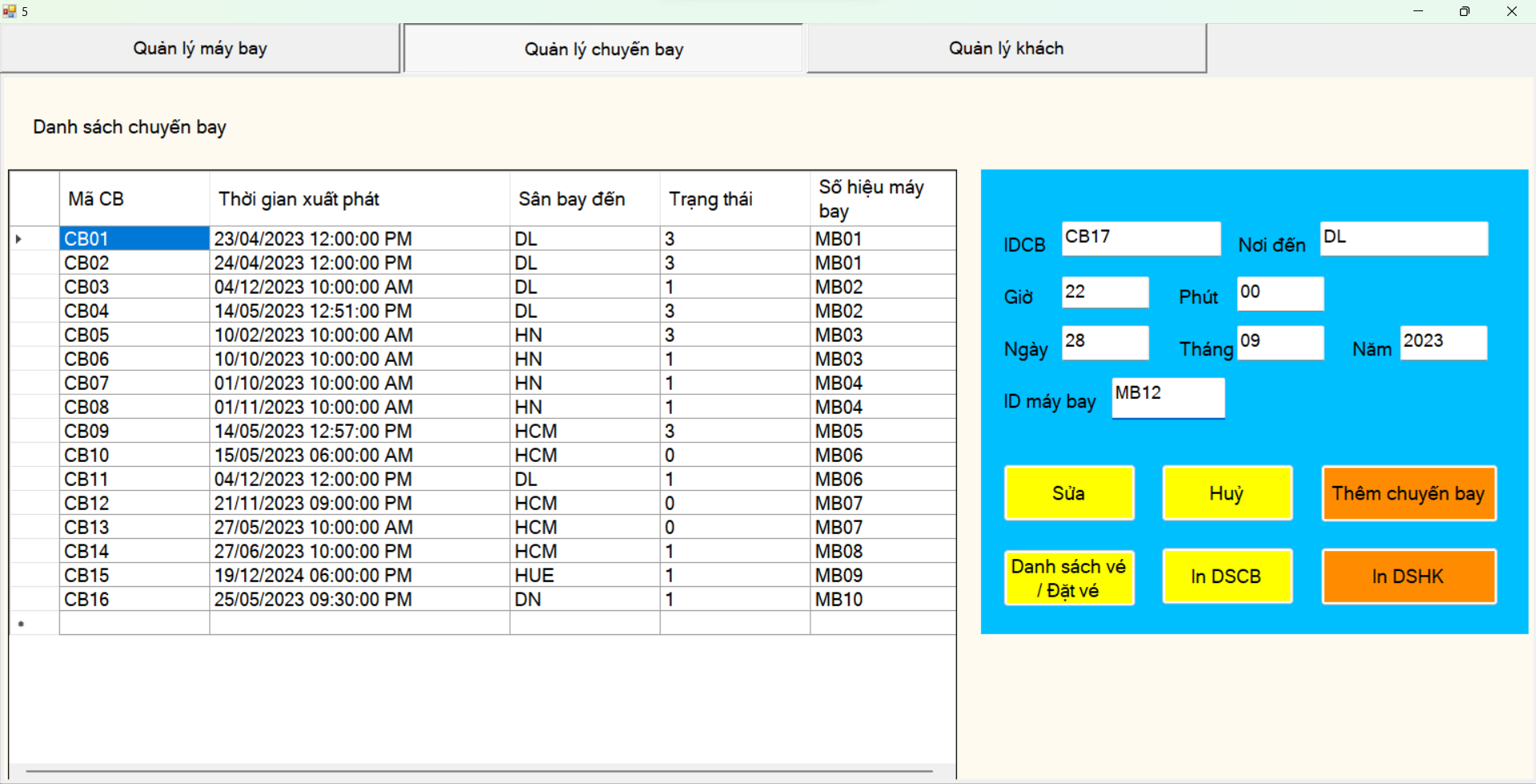


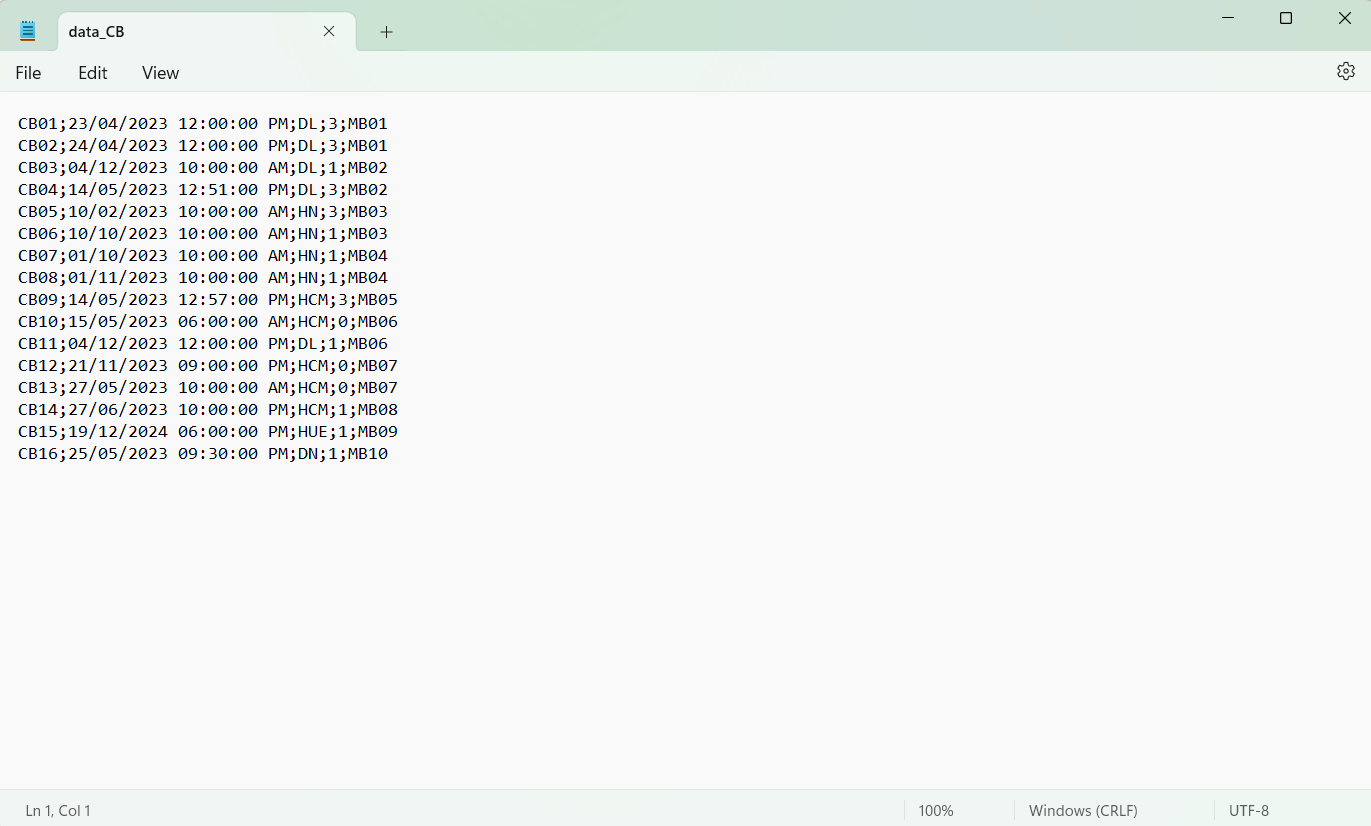


### 

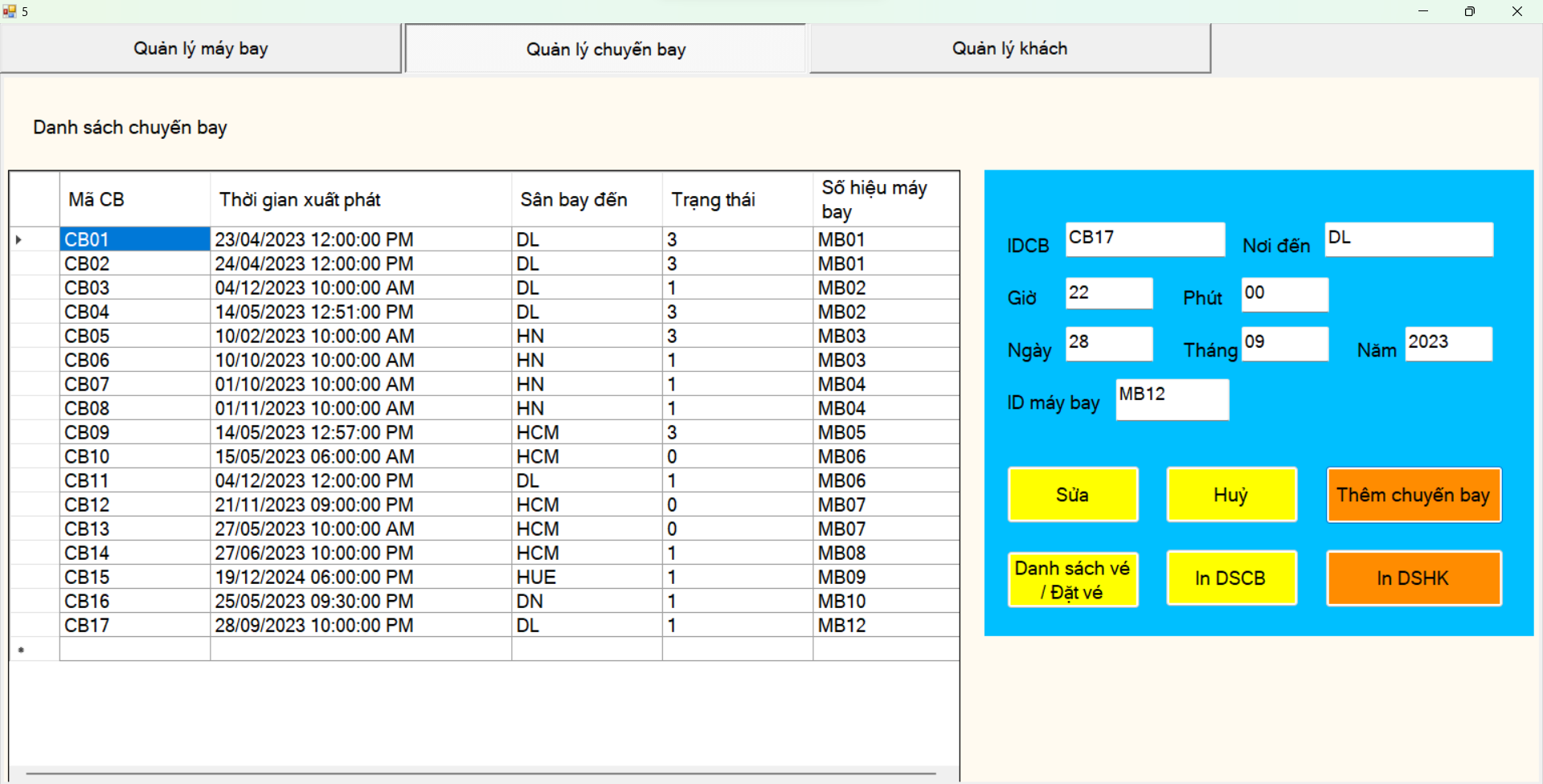
### **2.1.2.** **Cập nhật chuyến bay**

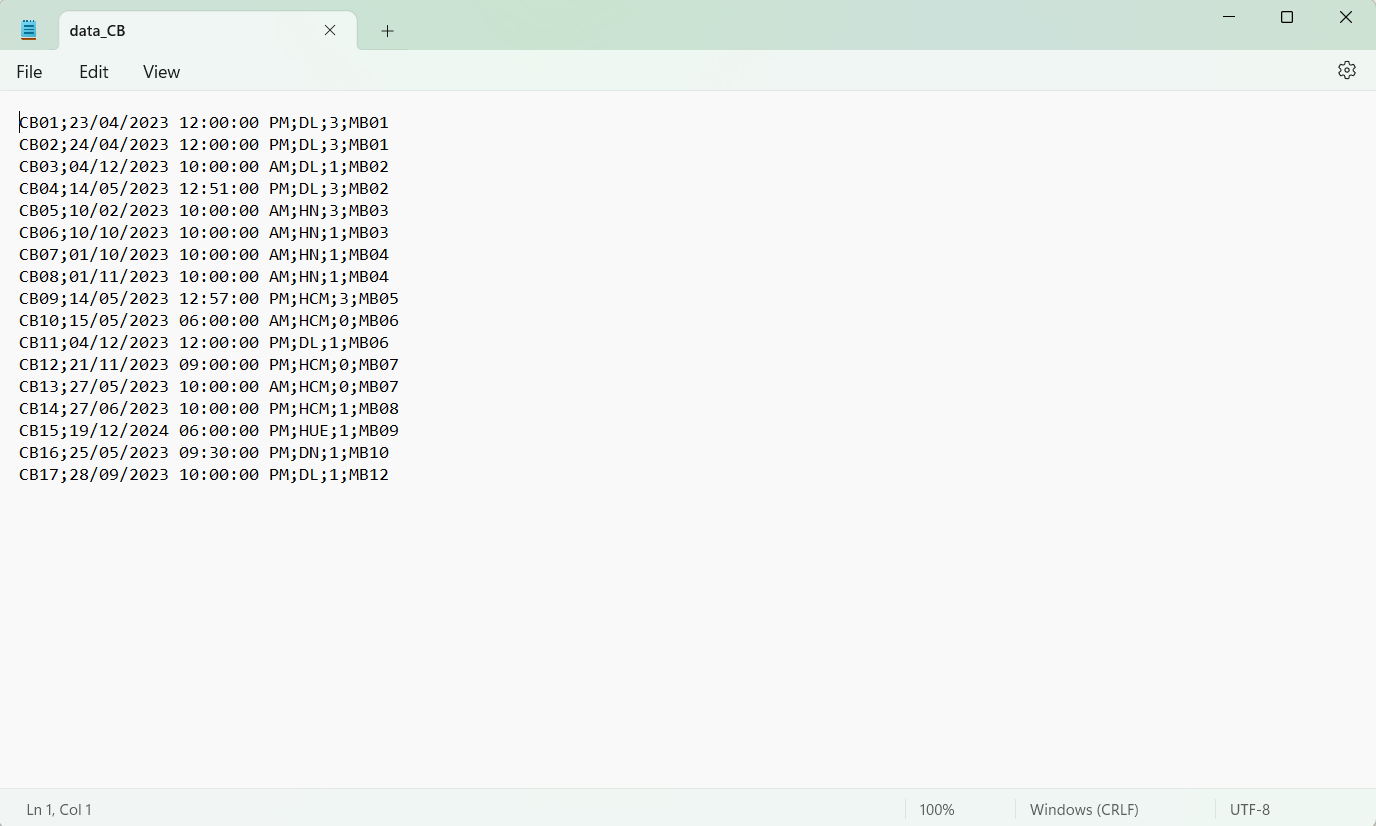
* **Lập chuyến bay mới**: Tạo một chuyến bay mới với các thông tin như sau:
  + ID CB: CB17, Nơi đến: DL
  + Giờ : 22 Phút : 00
  + Ngày 28 Tháng 09 Năm 2023
  + ID máy bay: MB12



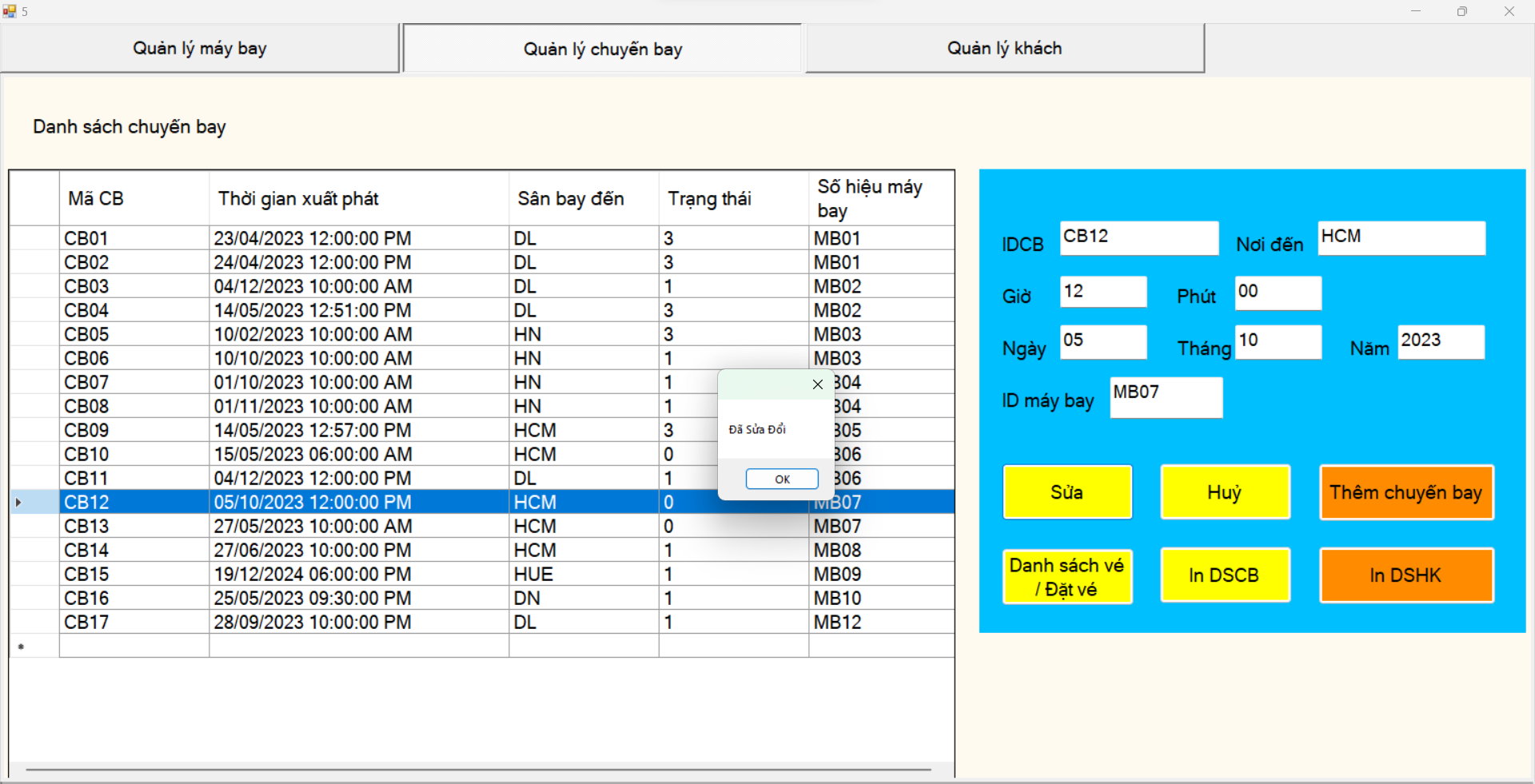


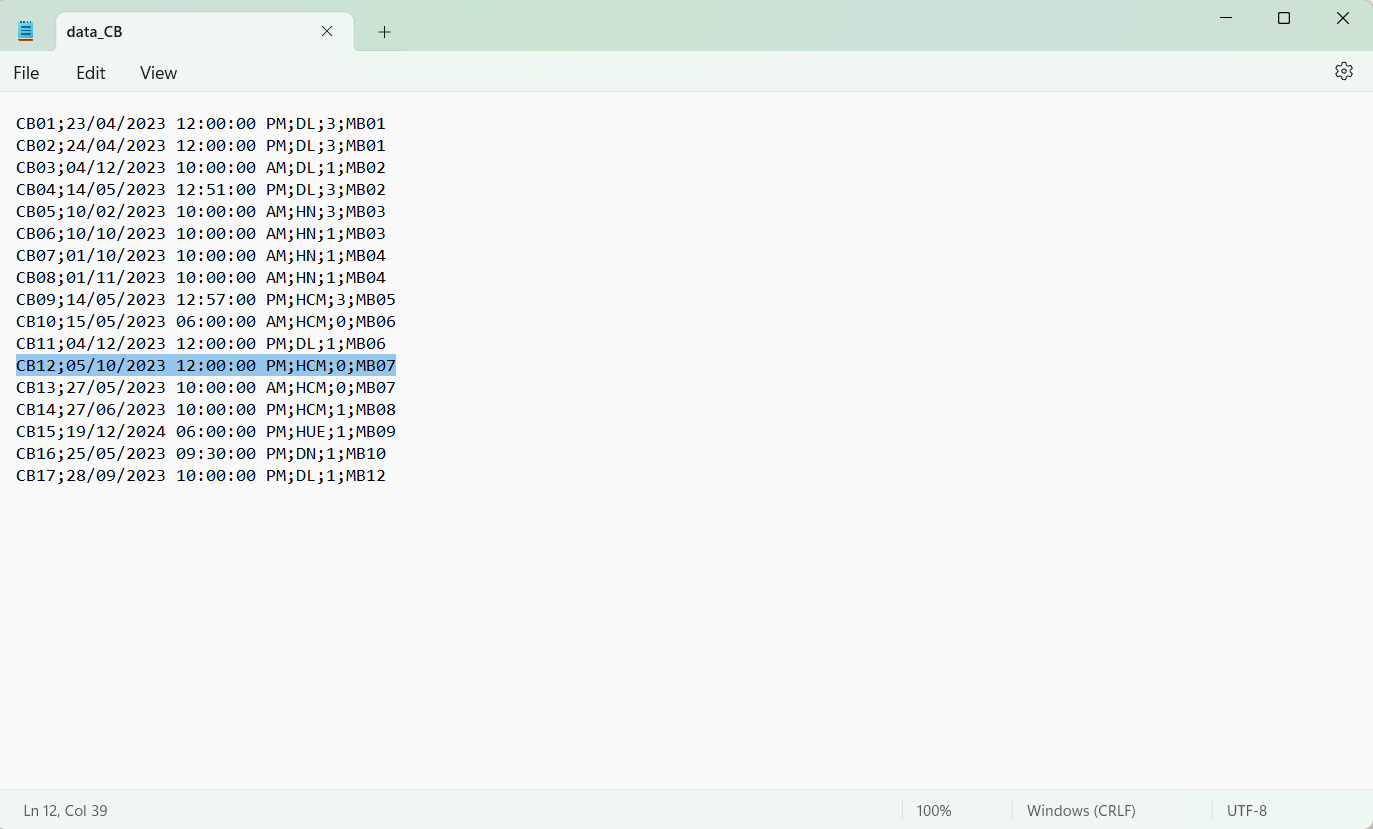
Kết quả ta có được là chuyến bay này sẽ cập nhật vào cuối danh sách hiện có và file data :



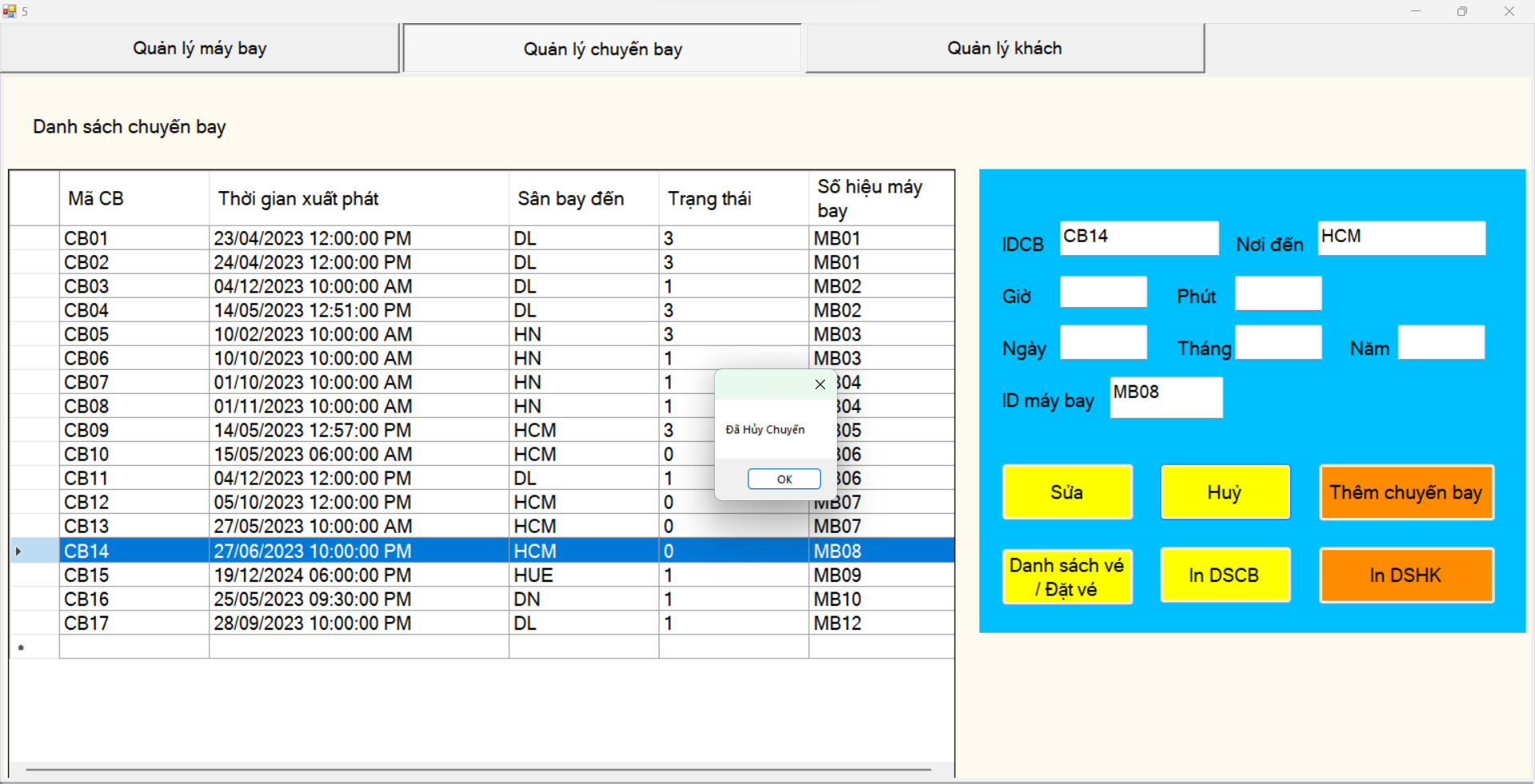


* **Hiệu chỉnh ngày giờ khởi hành**: hiệu chỉnh lại ngày giờ mã chuyến bay CB12 ngày 21/11/2023 9:00 PM thành ngày 05/10/2023 12:00 PM





* **Hủy chuyến** : Hủy chuyến bay với số hiệu CB14 thì trạng thái sẽ chuyển thành 0 ( Hủy chuyến)



### **2.1.3. Đặt vé**

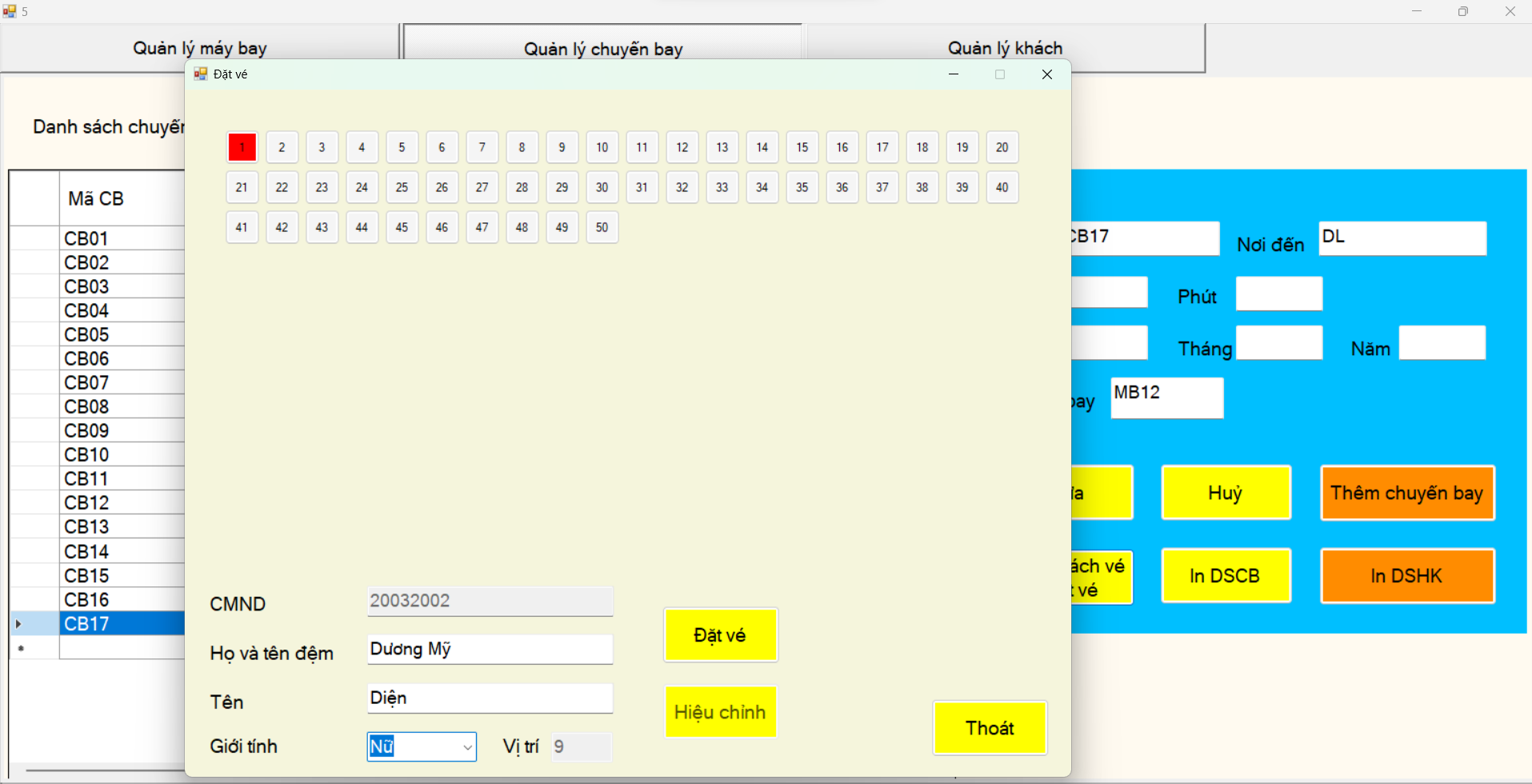
Ta đặt vé ở chuyến bay CB17 vừa tạo ở vị trí thứ 9 trên tổng 50 chỗ của máy bay thuộc chuyến bay này với các thông tin như sau :

CMND : 20032002

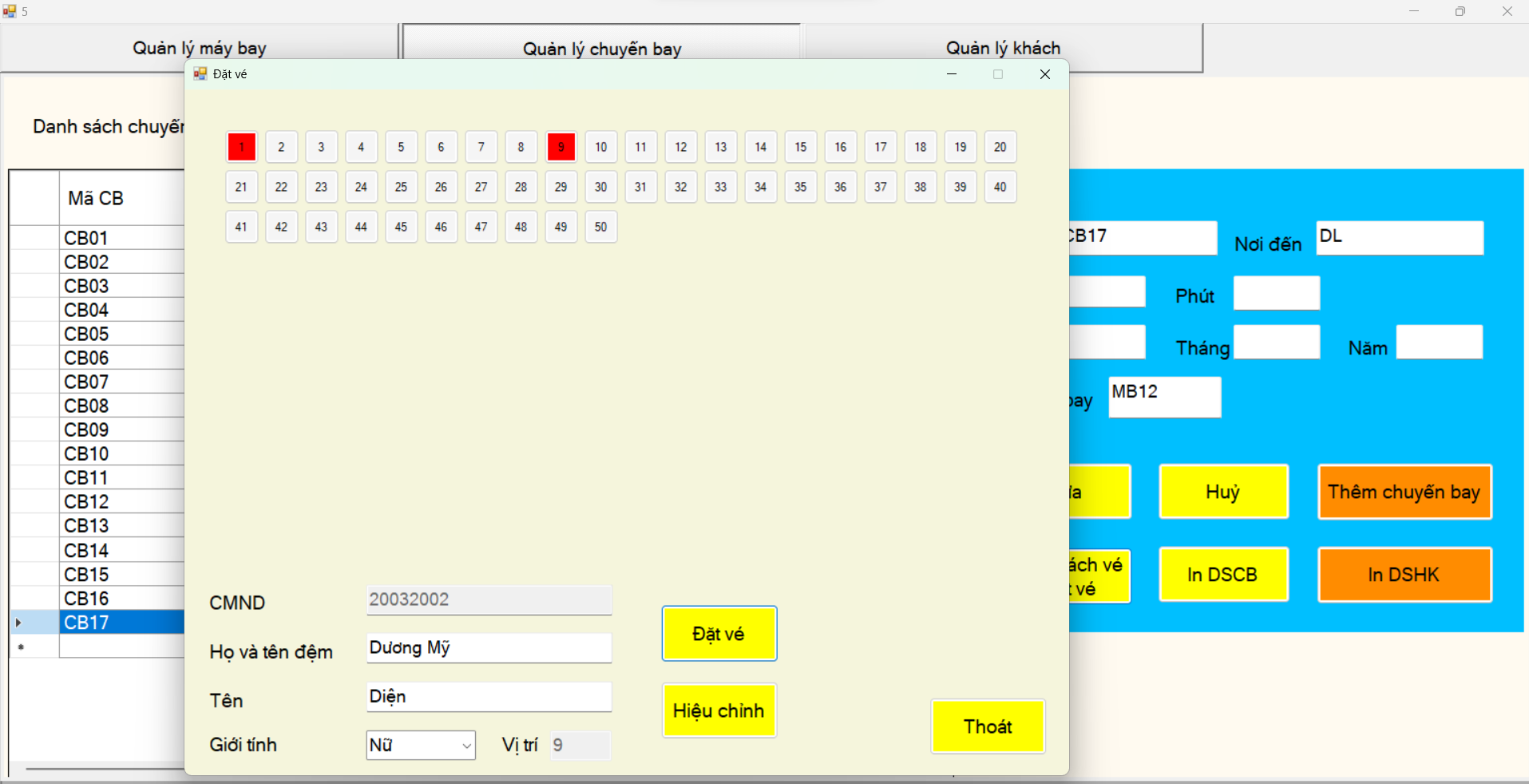
Họ và tên đệm: Dương Mỹ

Tên: Diện

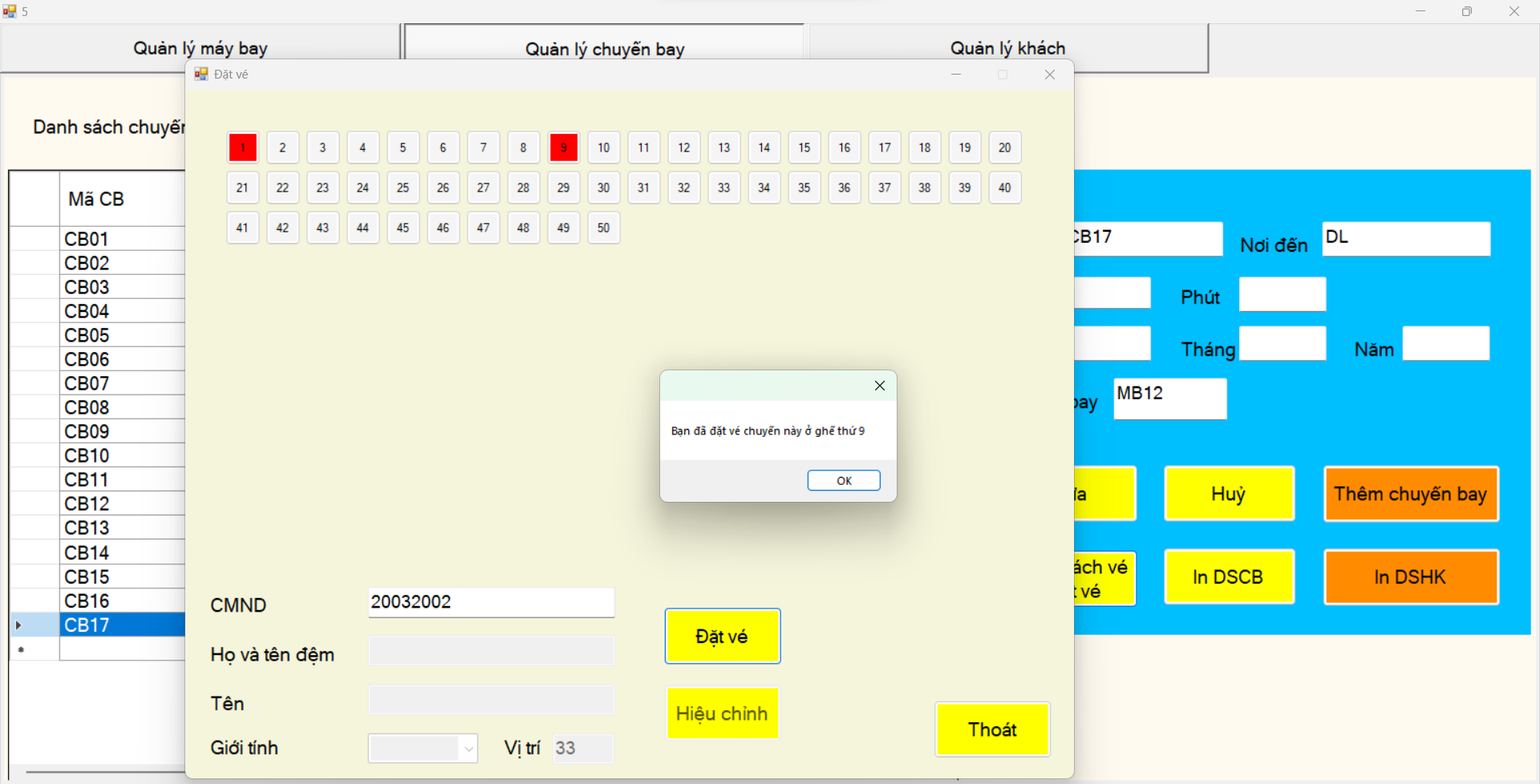
Giới tính : Nữ



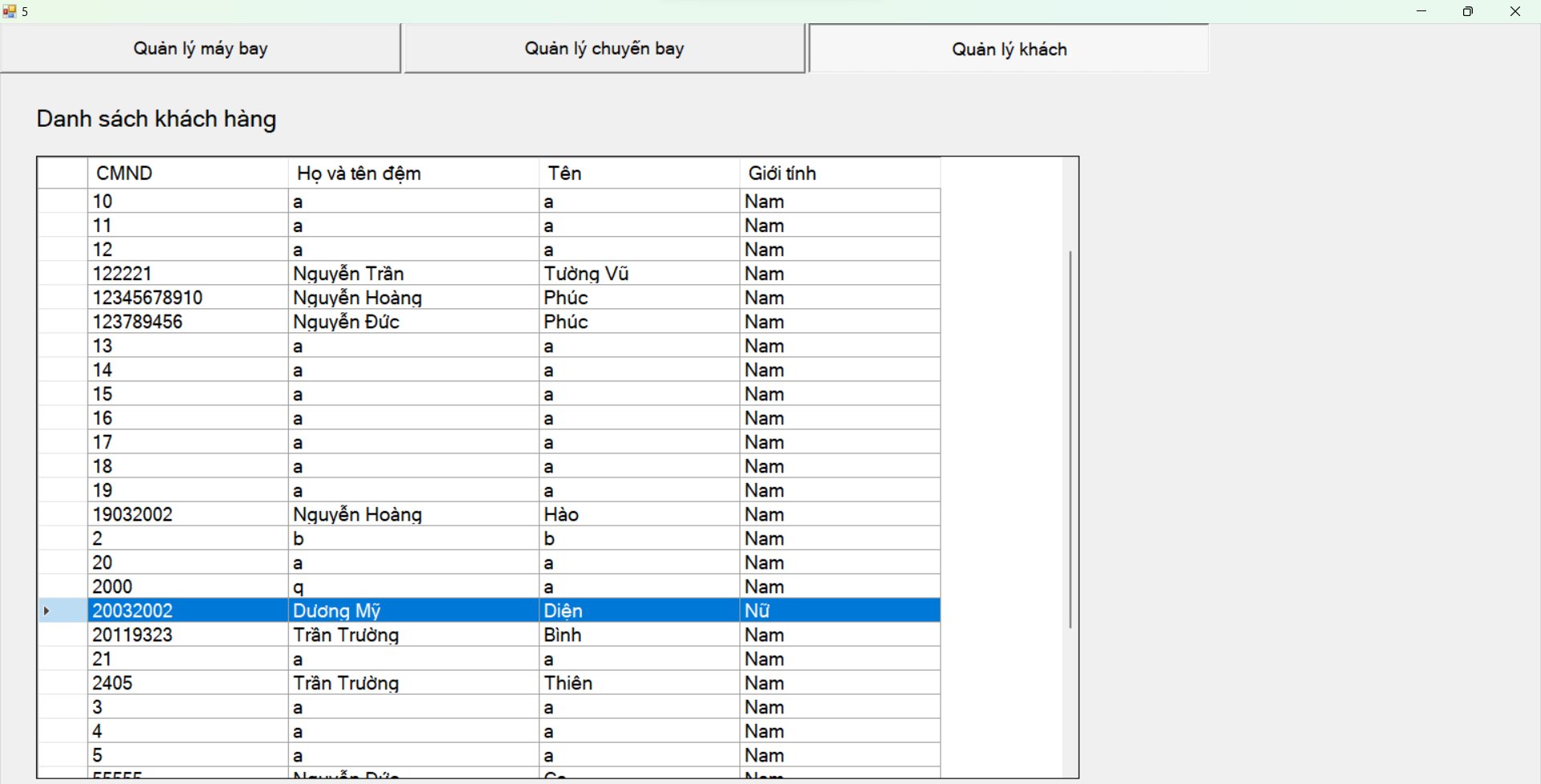
Kết quả có được khi đặt vé thành công :

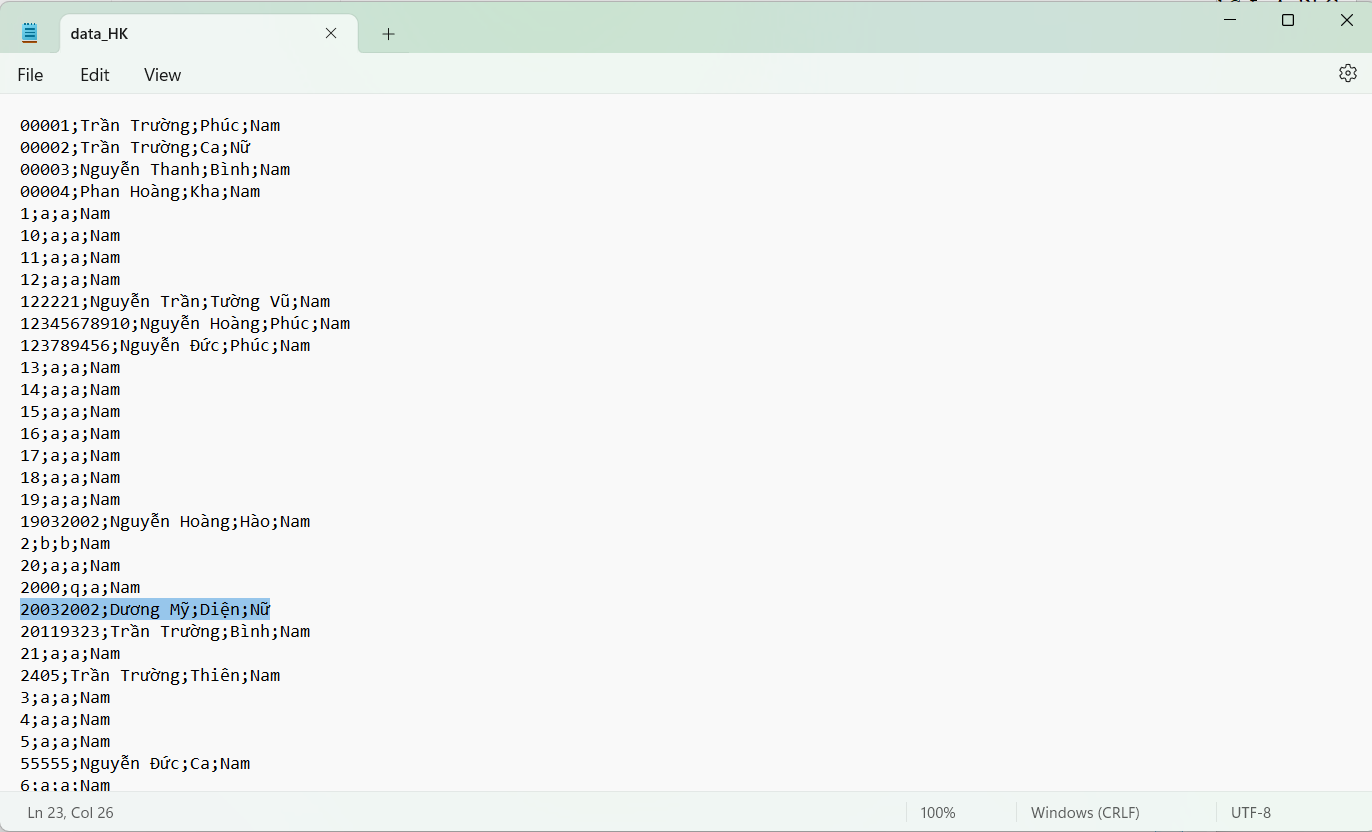


Nếu ta đặt vé trùng với số CMND thì sẽ hiển thị thông báo hành khách đã đặt chỗ :



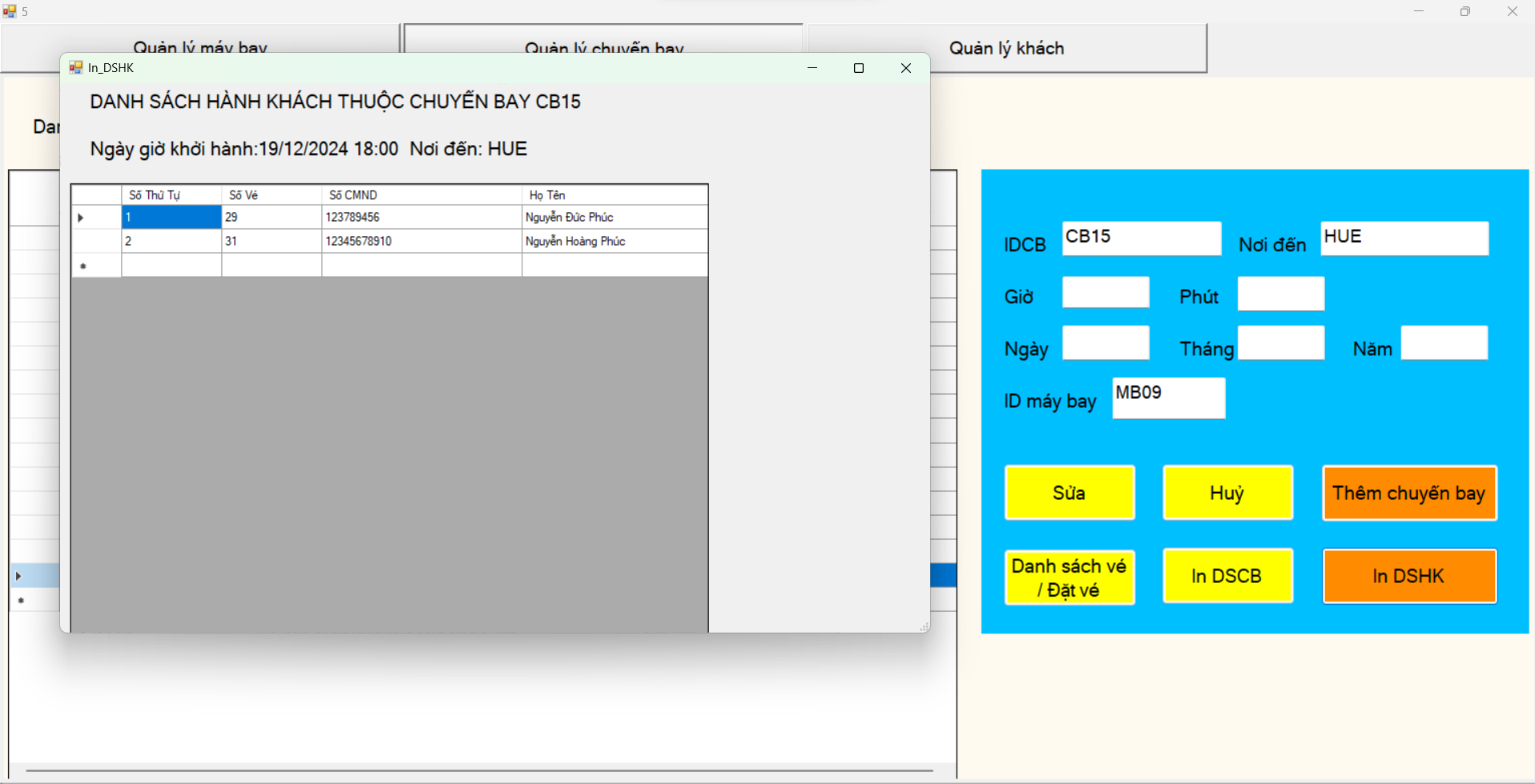
Sau khi đặt vé, thông tin người đặt vé cũng sẽ được cập nhật ở phần Quản lý khách đồng thời vào file data :





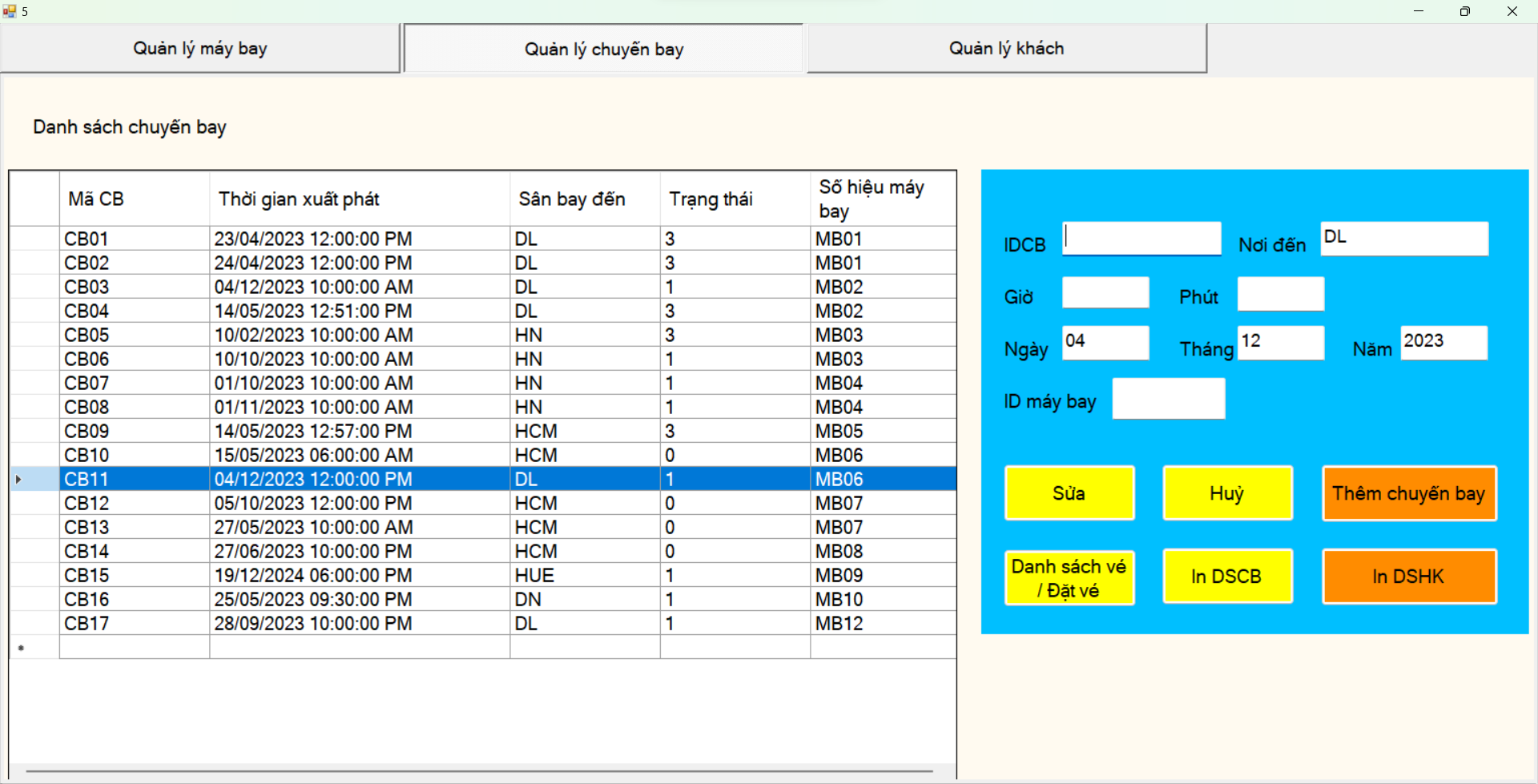
### **2.1.4. In danh sách các hành khách thuộc 1 chuyến bay dựa vào mã chuyến bay**

Ở phần quản lý chuyến bay ta thực hiện in danh sách hành khách chuyến bay với mã chuyến bay là CB15:

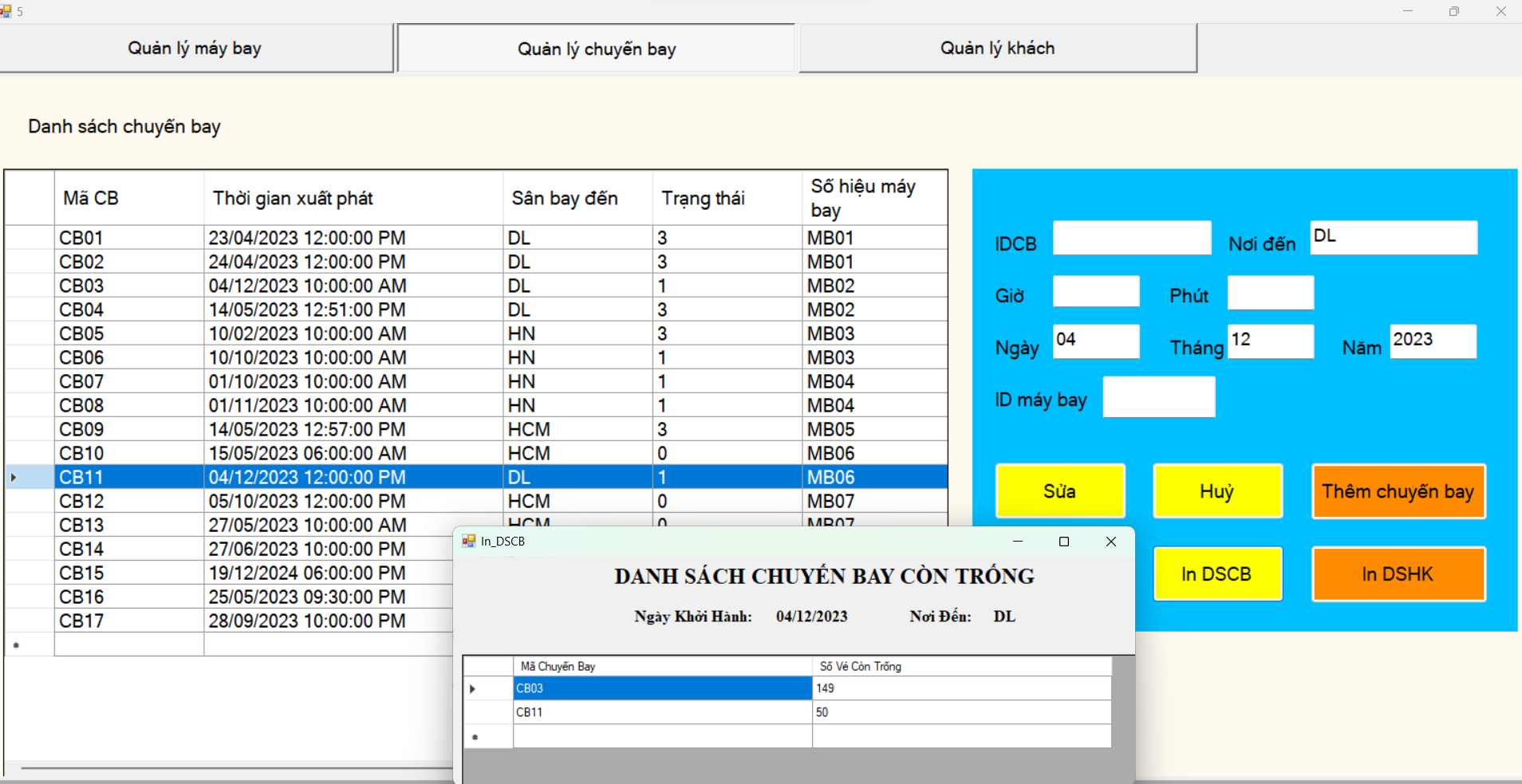


### **2.1.5. In danh sách các chuyến bay khởi hành trong ngày dd/mm/yyyy đến nơi XXXX mà còn vé ( cho biết cụ thể số lượng các vé còn trống )**

Ở phần quản lý chuyến bay ta thực hiện in danh sách các chuyến bay trong ngày 04/12/2023 nơi đến là DL:

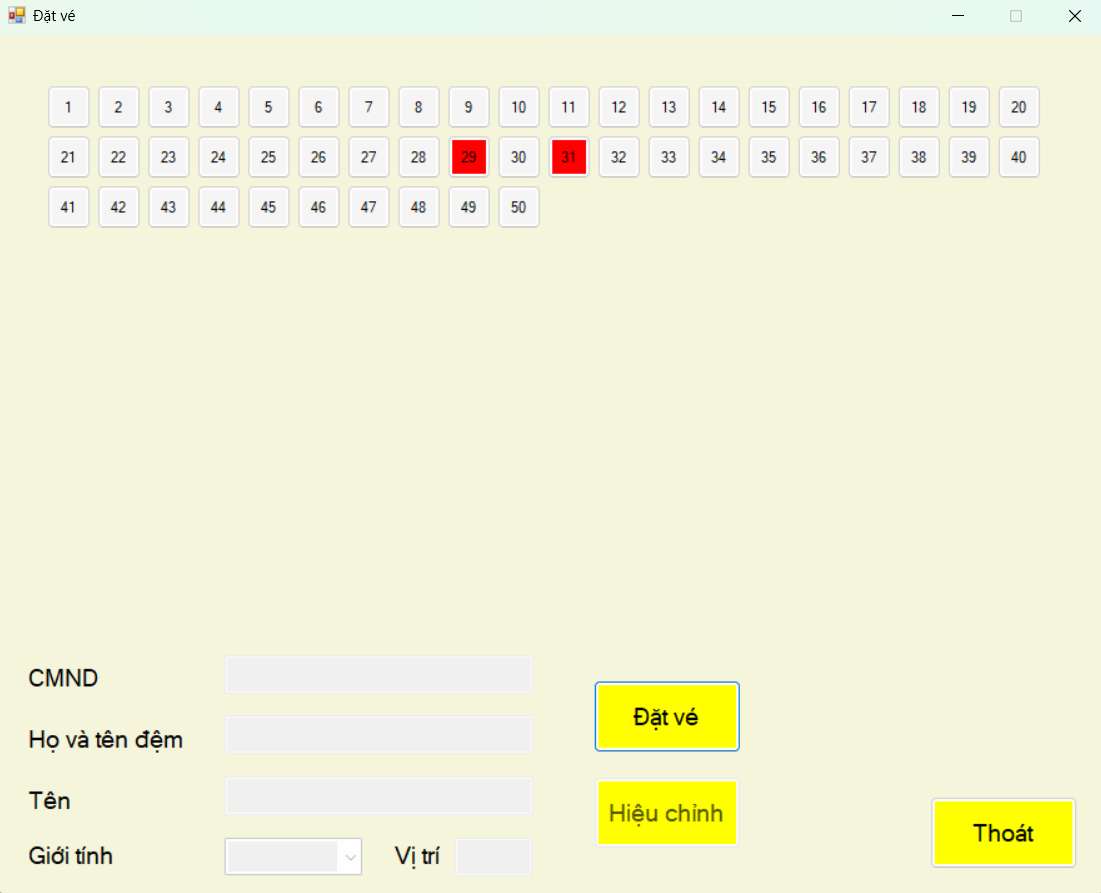


Kết quả ta có được là:

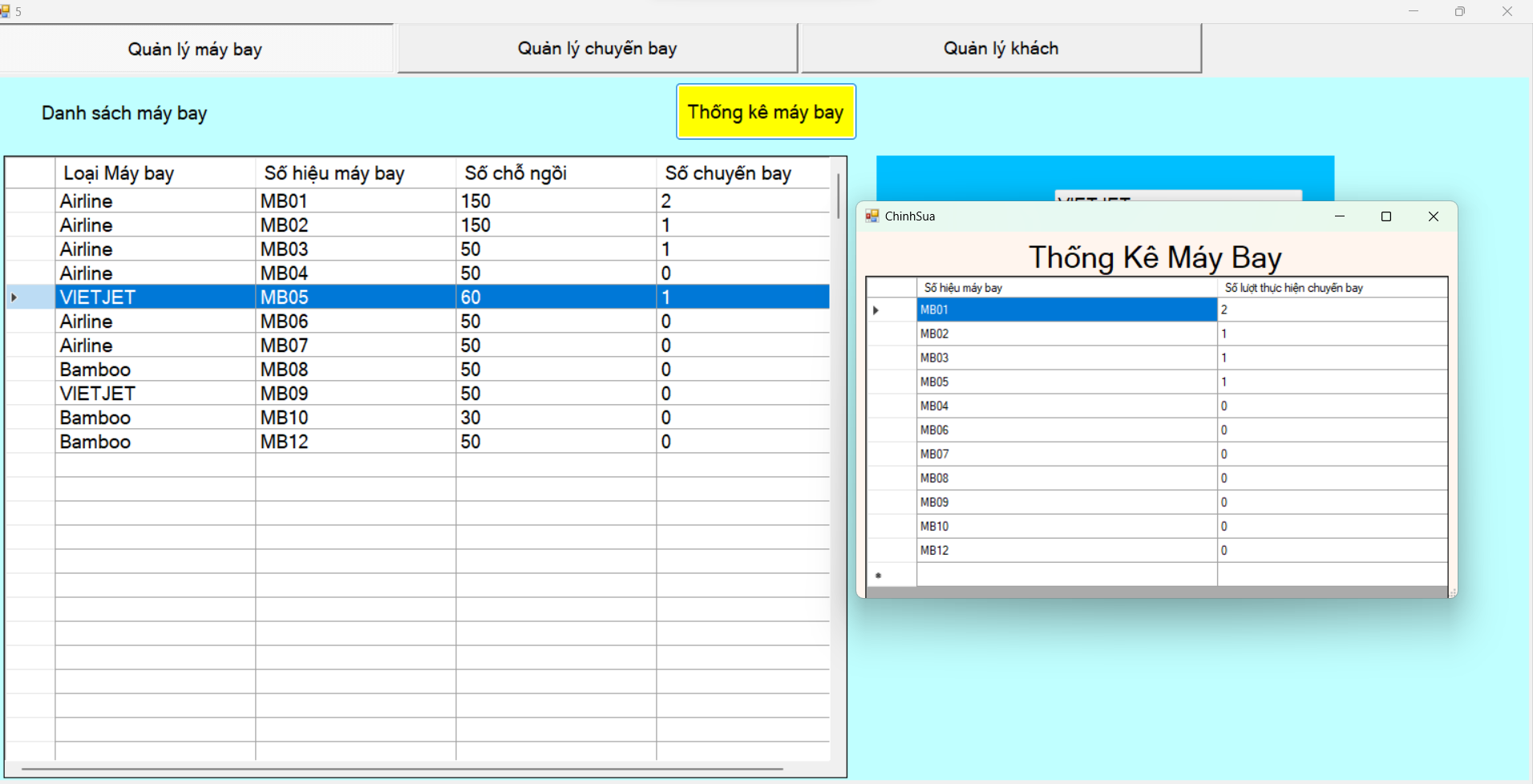


### **2.1.6. In danh sách các vé còn trống của 1 chuyến bay có mã chuyến bay là X**

Ta thực hiện in danh sách các vé còn trống ở phần đặt vé chuyến bay, hiển thị danh sách vé của chuyến bay CB15 ta thấy được những ô màu trắng là những chỗ chưa được đặt vé còn ô màu đỏ là những chỗ đã được đặt vé



### **2.1.7. Thống kê số lượt thực hiện chuyến bay của từng máy bay theo thứ tự số lượt thực hiện giảm dần**

****