Flight Management System

In this project we have attempted to make a software that is reminiscent of the one used by online portals like Yatra.com and Makemytrip.com. The users at a click of the mouse come to know about the flights departing from a given city or arriving at a given city.

Not only that the user can book a ticket but cancel it as well. If a certain flight is not plying for any reason, the administrator has an option to delete it from the database. We have prepared this program on a C++ platform. The functions that we use are specific to Turbo C++; so the program might not compile on other dialects of C++ like Borland C++ or Visual C++.

When the program is executed, first a welcome screen appears as defined in the function
void welcome(). Then the user hits any key of his choice to enter into the main menu.

Here he is faced with four broad options: **Flight**, **Find**, **Utility** and **Quit**. The user can toggle between these options using the right and left arrow keys. The function which controls this 'horizontal' menu is **Hmenu()** which returns the selected option as an integer.

As soon as we select any of the above options by hitting the Enter key a pull down menu showing the sub options gets displayed. For example, the sub options under the head Flight are **Add**, **View**, **Edit**, **Remove** and **Main Menu**.

The option Add calls the function **FSchedule()** where the user can add the flight data to the master database one by one. The master database of the software is the binary file "FLIGHT.DAT". At the first run of the code, this file is automatically created through the **open** function defined in the **fstream** class. When you are finished with the inputs; the file is saved through the function **close()**.

Since we want to allow for the records to be appended any time, the flag given to the open function is **ios::binary|ios::app** (recall that | stands for the bitwise OR operation). If the file is initially not present then this flag shall cause a new file to be created.

All the flight data is read into the various fields of an object of type **class FLIGHT** and finally this object is copied into the file through the function **write()**.

Next we come to the sub option **view** that calls the function **Mflightview()**. Here we open the master database in reading mode by specifying the flag as **ios::binary|ios::in** and then displaying the contents of the file one record at a time by looping till end of file. The function **read()** defined in class **fstream** is used here.

The software lends flexibility to modify the records as well. We can modify on the basis of flight number, departure, arrival, cities or even fare. All this is accomplished via the function **Mfileedit()** that is triggered through the sub option **Edit**. The entire database is searched sequentially for the desired record and then modified. To look for the record we use the functions **seekp()** and **tellg()**. Since we are reading as well as writing at the same time, the flag **ios::binary|ios::in|ios::out** is used.

The idea behind the function **fremove()** (triggered through the option **Remove**) is simple. Copy the contents of the original file into a fresh file except the object to be deleted. Delete the original file and rename the new file with the original file name. Renaming is done through the C++ function **rename()** defined in **stdlib.h**.

The option Find has the following sub options: **Flightnumber**, **Originalcity** and **Finalcity**. These options trigger the functions **sfno()**, **soc()**, and **sfc()** respectively. Here the user can search the database on the basis of flight number, Source or Destination. The tools and ideas that we use have already been discussed above.

Under the head utility we have mainly two functions: **Reserve** and **Cancel.** The functions attached to them are respectively **Bookticket()** and **Cancelticket().** The available flights in the database can be booked for a person by saving his details against the flight number. If a flight has already been booked then the software issues a warning message. Likewise an administrator can cancel and re-issue the ticket as well.

A function **Status()** is currently under development. This will at a glance let us know the list of people booked on various flights. Once this is made, we shall add it to the utility menu.

We have repeatedly used the overloaded functions DispCh(). One version of DispCh displays a single character at a location on the screen; whereas the other versions displays a string at any location. At the heart of these functions is the C++ function **gotoxy()** defined in **conio.h**.