بدنه اصلی کد از خطوط ۴۴ تا ۹۵ میباشد. یعنی قسمت مربوط به کد زیر

```cpp
int main() {

    // define requierment variables
    std::string user_choise = "0";
    std::vector<Airline> airlines;

    while(user_choise != "5") {

        // Update airlines list in each step
        airlines = updateAirlines("Airlines.txt");

        // Update airlines passengers list in each step
        updatePassengers("Passengers.txt", airlines);

        // initial message to user
        std::cout << "I would be happy to know what I can do for you?" << std::endl;
        std::cout << "1. Add passenger" << std::endl;
        std::cout << "2. Remove passenger" << std::endl;
        std::cout << "3. Search" << std::endl;
        std::cout << "4. Sort" << std::endl;
        std::cout << "5. Exit" << std::endl;
        std::cout << "Your choise: ";
        std::getline(std::cin, user_choise);

        // add passenger
        if (user_choise == "1") {
            addPassenger("Passengers.txt", airlines);
        }

        // remove passenger
        else if (user_choise == "2") {
            removePassenger("Passengers.txt");
        }

        // search among airlines or passengers
        else if (user_choise == "3") {
            search(airlines, "Passengers.txt");
        }

        // sort airlines or passengers
        else if (user_choise == "4") {
            sort("Passengers.txt", airlines);
        }

        // end of each process
        std::cout << "====================================" << std::endl;

    } // end of while(user_choise != "5")


    return 0;
}
```

همانطور که مشاهده میکنید یک رشته با نام user_choise تعریف کرده ایم که کاربر در هر قسمت میتواند این مقدار را به کد وارد کند. بنابراین یک حلقه while نوشته ایم که این حلقه همواره اجرا میشود به جز در مواردی که کاربر عدد 5 را وارد

کند که بیانگر خروج از برنامه است. در حلقه اصلی هم همواره خطوط زیر در خروجی چاپ میشوند که به کاربر راهنمایی کنند که کدام یک از عملیات را میخواهد انجام دهد.

```
I would be happy to know what I can do for you?
1. Add passenger
2. Remove passenger
3. Search
4. Sort
5. Exit
Your choise: 4
```

سپس چهار تا شرط if را نوشته ایم که اگر گزینه ورودی کاربر هر یک از این موارد باشد وارد یک قسمت از کدهای مان میشود. اگر کاربر عدد یک را وارد کند وارد قسمت اول که مربوط به اضافه کردن یک مسافر به یک airline است میشود و اگر عدد دو را وارد کرده باشد وارد قسمت مربوط به حذف کردن یک مسافر از یکی از airline ها میشود و اگر عدد سه را وارد کرده باشد وارد قسمت سرچ کردن میشود و اگر عدد ۴ را وارد کند وارد قسمت sort کردن میشود. دقت کنید که این حلقه دائما انجام میشود و در هر مرحله از آن شما میتوانید تحت عنوان کاربر یکی از دستورات گفته شده را اجرا کنید. مگر این که کاربر عدد ۵ را وارد کند که از حلقه بیرون آمده و برنامه تمام میشود. دقت کنید که شرط ورودی حلقه این است که ورودی عدد ۵ نباشد بنابراین منطقی است که اگر عدد ۵ وارد شد حلقه تمام شده و دیگر اجرا نشود. دقت کنید که اگر کاربر هر چیزی غیر از اعداد یک تا ۵ وارد کند هیچ اتفاقی نمیافتد و دائما به شما به صورت تکراری تصویر زیر را نشان میدهد.

```
I would be happy to know what I can do for you?
1. Add passenger
2. Remove passenger
3. Search
4. Sort
5. Exit
Your choise: 4
```

حال در ادامه باید به سراغ دو مرحله بعدی برویم. مرحله اول این است که برای ساختن هر مسافر و هر airline از چه ساختار داده ای استفاده کرده ایم و مرحله دوم هم این است که توابعی که ما در این جا برای هر کدام از option های ورودی نوشته ایم به چه صورت کار میکنند.

ساختار داده ای که ما در این جا پیاده سازی کرده ایم به شرح زیر میباشد.

```
// define Passenger data structure
struct Passenger {
    std::string airline;
    std::string name;
    std::string sex;
    std::string age;
    std::string marriage;
    std::string origin;
    std::string destination;
};


// define Airline data structures
struct Airline {
    std::string name;
    std::string safety;
    std::string international;
    std::string satisfaction;
    std::string plane;
    std::string discount;
    std::string first_class;
    std::string age;
    std::string employee;
    std::vector<int> ways;
    std::vector<Passenger> passengers;
};
```

برای هر مسافر چندین رشته تعریف کرده ایم که مشخص کند آن مسافر دقیقا چه اطلاعاتی دارد. این رشته ها شامل آن airline ای است که میخواهد با آن پرواز کند، اسم او، جنسیت او، سن، تاهل، مبدا و مقصد پروازش است. برای هر airline هم اسم، درصد امنیت آن، بین المللی بودن، درصد رضایت از آن، تعداد هواپیماهای آن، تخفیف، سن و تعداد کارمندان آن است. دقت کنید ما در این جا دو آیتم مسیرها و مسافران را هم خودمان اضافه کردیم. تمامی ویژگی ها تا بدین جای کار به صورت رشته تعریف میشد اما برای مسیر ها و مسافران باید تایپ آن را تغییر دهیم. برای مسیرها حتما باید از int استفاده کنیم که بتوانیم ماتریس هر airline را در آن ذخیره کنیم. درباره این ماتریس در ادامه توضیح خواهیم داد. بنابراین این ویژگی یک بردار ای مقادیر int است. همچنین یک بردار هم از مسافران ایجاد کرده ایم که به محض این که یک مسافر به یک airline اضافه شد آن مسافر را به لیست مسافران آن airline اضافه میکنیم.

حال باید به سراغ توضیح توابع برویم. تمامی توابع ما در همان حلقه while که در اول گفتم صدا زده شده اند. اولین تابع، تابع updateAirlines() است که در خط ۵۳ داخل حلقه همواره آن را صدا میزنیم. کاربرد این تابع به این صورت است که لیست تمامی airline ها را که در داخل یک فایل به نام Airlines.txt است میخواند و داخل یک بردار به نام airlines ذخیره میکند. این بردار که در خط ۴۸ آن را تعریف کرده ایم یک بردار از چندین airline است.

تابع گفته شده به شرح زیر است که در خط ۴۵۵ آن را تعریف کرده ایم. این تابع در ورودی اسم فایل را دریافت میکند که همان طور که گفتم اسم فایل برابر است با Airlines.txt. ابتدا فایل را با دستور ifstream در خط ۴۶۰ میخواند سپس در در خط بعدی آن یک بردار از airline ها ایجاد میکند. سپس بر روی فایل خوانده شده یک حلقه میزند و تمامی خط های آن را دانه به دانه میخواند. هر کدام از خط ها را به کمک دستور های زیر میشکند به کلمات متفاوتی که بین ویرگول ها قرار گرفته اند.

```
11          std::getline(line_stream, airline.name, ',');
12          std::getline(line_stream, airline.safety, ',');
13          std::getline(line_stream, airline.international, ',');
14          std::getline(line_stream, airline.satisfaction, ',');
15          std::getline(line_stream, airline.plane, ',');
16          std::getline(line_stream, airline.discount, ',');
17          std::getline(line_stream, airline.first_class, ',');
18          std::getline(line_stream, airline.age, ',');
19          std::getline(line_stream, airline.employee, ',');
```

چرا که همان طور که میدانید در فایل اصلی تمامی airline ها به این صورت ذخیره شده اند که اطلاعات آن بین ویرگول ها قرار داده شده اند. یعنی به صورت زیر

```
1 name,safety,international,satisfaction,plane,discount,first-class,age,employee
2 Delta,80,True,94,176,True,True,54,95000
3 American,83,True,84,203,False,True,60,129700
4 Turkish,65,False,33,102,True,False,33,40264
5 Indiana,54,False,43,305,False,True,76,100143
6 Emirates,90,True,92,242,True,True,44,125000
7 Lufthansa,78,True,88,198,True,True,49,107800
8 Air France,75,True,86,176,True,False,39,97500
9 Qantas,82,True,91,221,False,True,58,135000
10 Singapore,88,True,95,215,True,True,57,145000
11 British Airways,79,True,87,203,True,True,52,120000
```

بنابراین با شکستن هر خط به کلماتی که بین ویرگول ها قرار گرفته اند میتوانیم اطلاعات مربوط به هر airline را استخراج کنیم. کلمه اول قبل از ویرگول مربوط به اسم، کلمه دوم مربوط به میزان امن بودن و ... میباشد.

```cpp
// function 5) update airlines list
std::vector<Airline> updateAirlines (std::string file_name){

    std::string file_line;

    // read the file
    std::ifstream file(file_name);

    // create output vector
    std::vector<Airline> airlines;

    while (std::getline(file, file_line)) {

        Airline airline;

        std::istringstream line_stream(file_line);

        std::getline(line_stream, airline.name, ',');
        std::getline(line_stream, airline.safety, ',');
        std::getline(line_stream, airline.international, ',');
        std::getline(line_stream, airline.satisfaction, ',');
        std::getline(line_stream, airline.plane, ',');
        std::getline(line_stream, airline.discount, ',');
        std::getline(line_stream, airline.first_class, ',');
        std::getline(line_stream, airline.age, ',');
        std::getline(line_stream, airline.employee, ',');

        // create ways matrix
        if (airline.name == "Delta") {
            airline.ways = {0, 0, 1, 0,
                            1, 0, 1, 1,
                            0, 1, 0, 1,
                            1, 1, 0, 0};

            // store each airline in a vector
            airlines.push_back(airline);
        }
        else if(airline.name == "American") {
            airline.ways = {0, 1, 0, 0,
                            1, 0, 1, 0,
                            0, 1, 0, 1,
                            0, 0, 1, 0};

            // store the airline in a vector
            airlines.push_back(airline);
        }
        else if(airline.name == "Turkish") {
            airline.ways = {0, 1, 0, 1,
                            1, 0, 1, 0,
                            0, 1, 0, 1,
                            1, 0, 1, 0};
```

حال باید برای هر کدام از airline ها ماتریس مجاورت را بسازیم. این ماتریس ها همان هایی هستند که در فایل دستور کار در اختیار ما قرار گرفته شده اند. ما در این جا هر کدام از airline ها را به کمک if پیدا کرده و ماتریس مجاورت را برای هر کدام به صورت جداگانه تشکیل داده ایم. یعنی به صورت زیر.

```
        // create ways matrix
        if (airline.name == "Delta") {
            airline.ways = {0, 0, 1, 0,
                            1, 0, 1, 1,
                            0, 1, 0, 1,
                            1, 1, 0, 0};

            // store each airline in a vector
            airlines.push_back(airline);
        }
        else if(airline.name == "American") {
            airline.ways = {0, 1, 0, 0,
                            1, 0, 1, 0,
                            0, 1, 0, 1,
                            0, 0, 1, 0};

            // store the airline in a vector
            airlines.push_back(airline);
        }
        else if(airline.name == "Turkish") {
            airline.ways = {0, 1, 0, 1,
                            1, 0, 1, 0,
                            0, 1, 0, 1,
                            1, 0, 1, 0};

            // store the airline in a vector
            airlines.push_back(airline);
        }
        else if(airline.name == "Indiana") {
            airline.ways = {0, 1, 0, 0,
                            1, 0, 1, 1,
                            0, 1, 0, 0,
                            0, 1, 0, 0};

            // store the airline in a vector
            airlines.push_back(airline);
        }
        else if(airline.name == "Emirates") {
            airline.ways = {0, 1, 1, 1,
                            1, 0, 1, 0,
                            1, 1, 0, 1,
                            1, 0, 1, 0};

            // store the airline in a vector
            airlines.push_back(airline);
        }
```

این ماتریس ها مشخص میکنند که برای هر airline از هر کشور به کدام کشور میتوانیم پرواز داشته باشیم. برای مثال اگر یک عنصر در سطر اول و ستون دوم این ماتریس برابر با یک باشد یعنی از کشور امارات به آمریکا پرواز دارد و اگر صفر باشد یعنی از امارات به آمریکا پروازی ندارد. دقت کنید که سطر ها از اول تا چهارم مربوط به کشورهای امارات، آمریکا، آلمان و کانادا است و ترتیب ستون ها نیز به همین صورت است.

بعد از این که ماتریس مجاورت را برای هر کدام از airline ها ساختیم آن airline را به برداری که کل airline های مان بود اضافه میکنیم و در ادامه آن بردار را در خروجی پاس میدهیم. یعنی به صورت زیر.

```
4     return airlines;
```

بنابراین اگر بخواهم به صورت خلاصه بگویم تابع updateAirlines() همواره در ورودی نام فایلی که در آن لیست airline ها را نوشته ایم دریافت میکند و سپس با خواند اطلاعات از آن فایل یک بردار از تمامی airline ها ایجاد میکند و آن را در خروجی بر میگرداند.

تابع بعدی که در خط ۵۶ از همان حلقه while که توضیح دادم استفاده شده است تابع updatePassengers() است. این تابع هم کاملا مشابه با همان تابع updageAirlines() عمل میکند منتها این بار برای مسافران. این فایل نام فایلی که در آن اطلاعات مسافران را ذخیره میکنیم در ورودی دریافت میکند و نیز همان بردار airlines که در آن لیستی از airline ها ساخته بودیم میگیرد و سپس با خواندن اطلاعات مسافران از فایل ورودی به صورت خط به خط هر کدام از مسافران را به airline ای که در آن ثبت نام کرده اند ذخیره میکند. این تابع در خطوط ۵۷۵ تا ۶۰۰ نوشته شده است که به صورت زیر است. همان طور که مشاهده میکنید کاملا مشابه با تابع قبل است. یعنی در ابتدا با دستور ifstream فایل را میخواند و سپس با دستور getline تک تک خطوط آن فایل را دریافت میکند سپس خط های خوانده شده را با دستور istringstream به خط های stream شده تبدیل میکند که میتوانیم در ادامه مثل همان تابع قبل کلمات بین ویرگول ها را جدا کنیم. اولین کلمه قبل از ویرگول مربوط به اسم airline ای که مسافر در آن ثبت نام کرده است میباشد. دومین کلمه بعد از ویرگول مربوط به اسم مسافر میباشد و ... . نحوه ذخیره اطلاعات مسافران را در تصویر زیرین این تابع آورده ام. همچنین بعد از این که اطلاعات هر مسافر را از هر خط دریافت کرد آن را در اطلاعات یک struct مسافر ذخیره میکند. خط ۵۸۳ از یک struct مسافر یک مسافر ساخته است و اطلاعات را در آن ذخیره کرده است. سپس در خط ۵۹۴ بر روی تمامی airline ها پیمایش کرده

است و اگر airline با اسمی که مسافر در آن ثبت نام کرده بود را پیدا بکند مسافر را به لیست مسافران آن airline اضافه میکند. در نهایت لیست مسافران تمامی airline ها به روز میشوند.

```cpp
// function 6) Update passengers
void updatePassengers(std::string file_name, std::vector<Airline>& airlines) {

    std::ifstream file(file_name);
    std::string file_line;

    while (std::getline(file, file_line)) {
        Passenger passenger;

        std::istringstream line_stream(file_line);
        std::getline(line_stream, passenger.airline, ',');
        std::getline(line_stream, passenger.name, ',');
        std::getline(line_stream, passenger.sex, ',');
        std::getline(line_stream, passenger.age, ',');
        std::getline(line_stream, passenger.marriage, ',');
        std::getline(line_stream, passenger.origin, ',');
        std::getline(line_stream, passenger.destination, ',');

        for (int i = 0; i < airlines.size(); ++i) {
            if (passenger.airline == airlines[i].name) {
                airlines[i].passengers.push_back(passenger);
            }
        }
    }
}
```

همان طور که گفتم اطلاعات مسافران به صورت زیر ذخیره شده اند.

```
airline,name,sex,age,marriage,origin,destination
Delta,Danyal Iran Mehr,Male,21,Single,America,Emirates
Delta,Reza Iran Mehr,Male,25,Married,America,Emirates
Delta,Abas Iran Mehr,Male,28,Married,Canada,America
Air France,Zohreh Iran Mehr,Female,30,Married,Germany,Canada
Lufthansa,Ali Alavi,Male,63,Married,Emirates,Canada
```