Технически университет - Варна

Дисциплина : Обектно Ориентирано Програмиране-1 част

**ДОКУМЕНТАЦИЯ**

**Изработил:** Николай Иванов **Проверил:**

**КСТ гр. 1а курс 2 .....................**

**Фак. № 17621301**

Съдържание

Кратко описание на първият клас......................................4

Кратко описание на втория клас и неговите функции......4

функция 2.1 - less\_than\_2k()................................................4

функция 2.2 - frenchFlights().................................................5

функция 2.3 - push\_in\_vector().............................................5

функция 2.4 - push\_in\_vector2()...........................................5

Кратко описание на трети клас и неговите функции.........5

функция - 3.1 – Func1\_get\_flights\_and\_owner()..................6

функция - 3.2 – Func2\_get\_max\_of\_destination()................6

функция - 3.3 – Func3\_getPlane\_with\_max\_destinations()..6

ф-я 3.4 Func4\_get\_max\_flights\_destination\_by\_company().7

функция -3.5 Func5\_return\_vector\_of\_Cplanes().................7

функция -3.6 - show\_french\_flights()....................................8

3.7 – Работа с файлове.........................................................8

Листинг на програмата с коментари ..................................9

Демонстрация на работата на програмата ......................18

**Куросв Проект N 2**

Програма за обработка на самолети

I. Да се дефинира клас Cplane, съхраняващ информация за марка на самолета и летатилни часове с необходимите конструктори, моетоди и оператори.

II. Да се дефинира клас CAirport, съхраняващ информация за име на авиокомпанията, националност, брой дестинации и самолети, като последните се съхраняват в контейнер map<CPlane , unsigned>. Име на дестинацията и брой полети до нея са съхранени в multimap <String , unsigned>. Освен необходимите методи, да се реализират и следните член функции:

* Изчислява и връща средния брой дестинации на френските авиокомпании;
* Връща списък от самолети (list<CPlane>), с летателни часове по-малко от 2000 и брой дестинации повече от 10;

III. Да се дефинира клас CAirport, съхраняващ информация за името на летището, номер на полет и контейнер съхраняващ CAirtravel, съхраняващ данните за различните авиокомпании и брой полети. Например:

Class CAirport{

string name;

unsigned n\_polets ;

multimap<CAirtravel, unsigned> airtravel;

Да се дефинира конструктор с Параметри за летището и име на файл, съдържащ необходимата информация за едно летище.

Да се добавят следните методи:

* При подаден аргумент - марка самолет, връща броя полети и авиокомпания, собственик на самолета;
* При подаден аргумент име на дестинция, връща името на авиокомпанията, обезпечила най-много полети;
* Връща марка самолет, с най-много полети;
* При подаден аргумент авиокомпания, връща дестинация с най-много полети;
* Връща контейнер от самолети, съдържащ летателни часове >10000 на всички самолети от авиокомпаниите, оилзваще летище София

IV. да се демонстрира работата на класа CAirport, като се дефинира обект от този клас и се предостави възможност за различни справки за съответното летище.

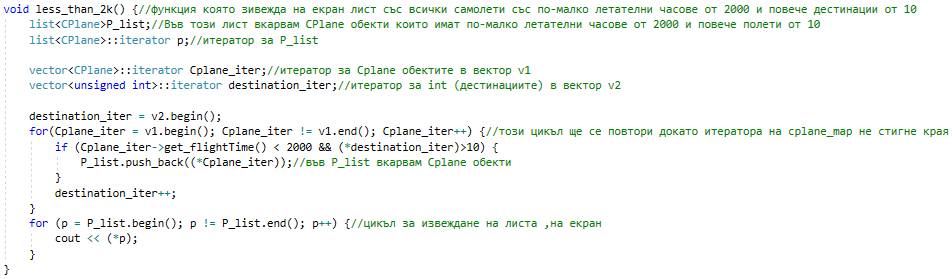
Документиране на курсовия проект: .doc файл със заглавна страница , условие на задачата, кратко описание на класовете и функциите и листинг на програмата с коментари.

Програмата се състои от 3 класа: class CPlane, class CAirtravel и class CAirport

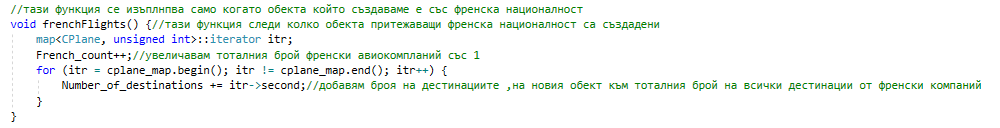
Първия клас (CPlane) се състои от 2 променливи за модел и литателни часове (string planeModel, int flightTime) на самолета, съдержа два конструктора подразбиращ се и копиращ , get и set функции за двете променливи, предефиниране на оператора ‘<‘ и поток изход. Използва се за създаване на самолет обекти ,които ще бъдат подадени като аргумент във втория клас

Втория клас (CAirtravel) се състои от 2 string променливи за име и националност на компанията, map (cplane\_map) който държи обекти от тип CPlane като ‘key’ и unsigned int(брой дестинации)като ‘value’, multimap (multiM\_destination\_container) ,който държи string(име на дестинация) като ‘key’ и int(брой полети към дестинацията) като ‘value’.CAirport има 4 вектора в които влизат разделените данни ‘key’ и ‘value’ от map cplane\_map и multimap multiM\_destination\_containe.Класът има 2 статични променливи от тип int(French\_count, Number\_of\_destinations), те се използват за намиране броя на френските авиокомпании.Предефиниране на оператор ‘<’ , 2 конструктора( експицитен и дефалтен) и 4 void функции:

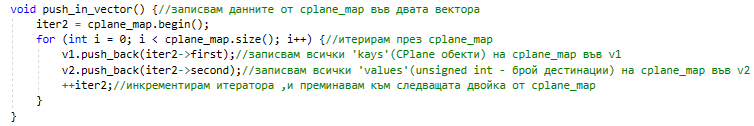
**2.1 -** **void less\_than\_2k()** – функцията се извиква в конструктора на CAirtravel. Тази функция проверява дали във мапа със CPlane обекти има самолети със по-малко от 2000 литателни часа и повече от 10 дестинации.Тези CPlane обекти които изпълняваткритериите , влизат във list контейнер със име P\_list. След проверката ,функцията извежда листа на екран.



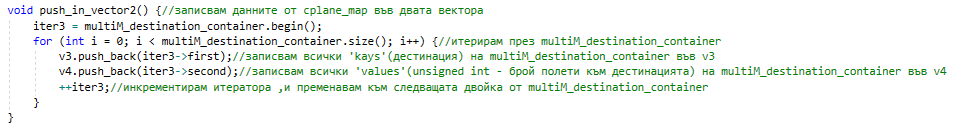
**2.2 -** **void frenchFlights()** – функцията се извиква в конструктора на CAirtravel. Ако обекта който се създава е със френка националност. frenchFlights() инкраментира статичната променлива която е отговорна за броене на френските самолети, и добавя броя полети на авиокомпанията към тоталния борй полети от френски компании.



**2.3 -** **void push\_in\_vector()** - функцията се извиква в конструктора на CAirtravel. Тази функция разделя map ‘cplane\_map’ на ‘key’(CPlane) и ‘value’(int) ,и ги вкарва със съответните вектори ‘v1’(CPlane) и ’v2’(int).Тези 2 вектора се използват във **less\_than\_2k()** и фукнците във клас CAirport, за достъп на капсулованите данни от вървия клас (CPlane).

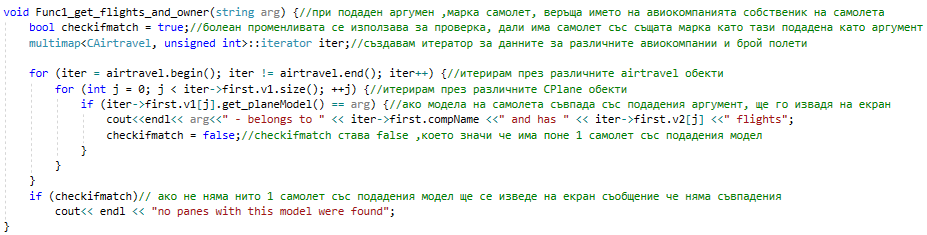


**2.4 -** **void push\_in\_vector2()** - функцията се извиква в конструктора на CAirtravel. Тази функция разделя multimap ‘multiM\_destination\_container’ на ‘key’(String) и ‘value’(int) ,и ги вкарва със съответните вектори ‘v3’(String) и ’v4’(int).Тези 2 вектора се използват фукнците във клас CAirport, за достъп на капсулованите данни от вървия клас (CPlane)

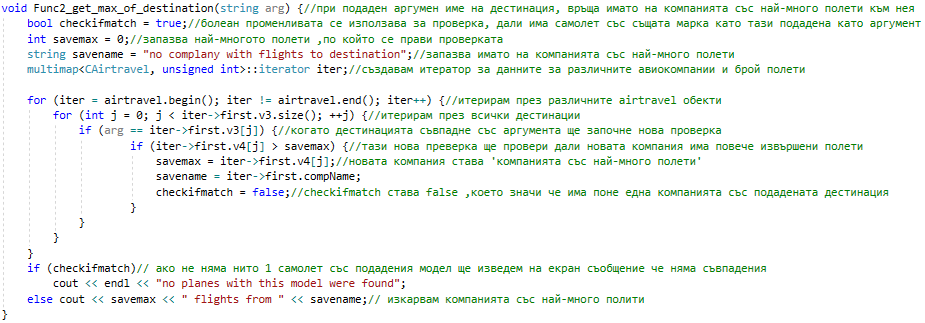


Трети клас (CAirport) се състои от 2 променливи за име на летището и номер на полет (string airportName, unsigned int n\_flights),multimap airtravel съдържащ данните за различните авиокомпании и брой полети, 3 конструктора 1 дефалутен и 2 експлицитни, единия приема string(име на файл) а другия string, int и multimap. Класът има и 6 член функции.

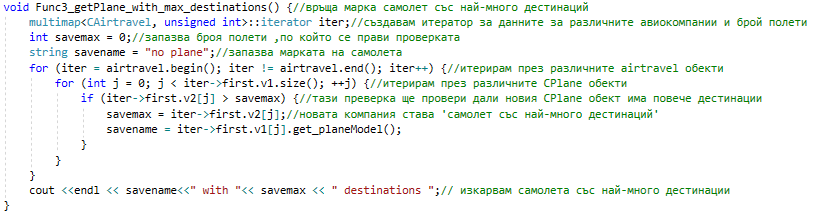
**3.1 –** **void Func1\_get\_flights\_and\_owner(string arg)** - функцията се използва в main(), от потребителя.С помощта на итератори получавам достъп до данните(v1 ,v2 и compName) от предишните класове.Функцията сравнява подадения String аргумент със марките на всички самолети и врща компанията собственик или съобщение ,че самолета не е намерен.



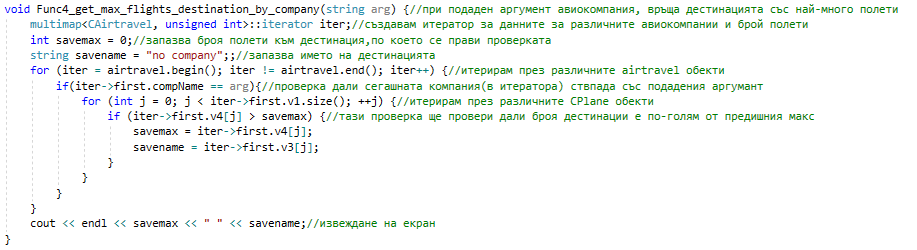
**3.2 –** **void Func2\_get\_max\_of\_destination(string arg)** - функцията се използва в main(), от потребителя. С помощта на итератори получавам достъп до данните(v3 ,v4 и compName) от предишните класове. Функцията сравнява подадения String аргумент със дистенациите.Когато намери дестинация която съвпада с издадения аргумент, запазва името на компанията която извършва полетите във променливата savename ,а броя на полетите в savemax. Ако функцията намери друга компания със повече полети от предишната, новата компания ще заеме мястото на старата. След като функцията проверо всички дестинации, ще извете на екран имато на компанията със най-много полети към избраната дестинация.

****

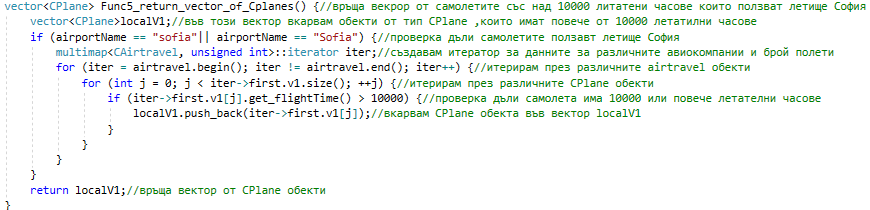
**3.3 – void Func3\_getPlane\_with\_max\_destinations()** - функцията се използва в main(), от потребителя. С помощта на итератори получавам достъп до данните(v1 и v2) от предишните класове. Функцията итерира през всички CPlane обекти и техните борй дестинации.Когато функцията намери CPlane обект със повече дестинации от предишния, му запазва името във променливата savename а броя на дестинациите във savemax. След като функцията провери всички самолети, ще извете на екран марката на самолета със най-много дестинаци и боря дестинации.



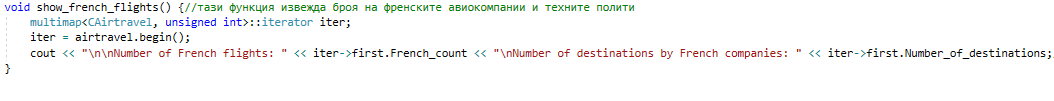
**3.4 – void Func4\_get\_max\_flights\_destination\_by\_company(string arg)** - функцията се използва в main(), от потребителя. С помоща на итератори получавам достъп до данните(v3 , v4 и compName) от предишните класове. Ако подадения аргумент и compName съвпадат, функцията ще итерира през дестинациите на компланията и ща запази тази детсинация към която има най-много полети.След като функцията провери всички детсинации, ще изведе на екран ,най-посещаванат дестинация за избраната компания.



**3.5 – vector<CPlane> Func5\_return\_vector\_of\_Cplanes()** - функцията се използва в main(), от потребителя. С помощта на итератори получавам достъп до данните(v1) от предишните класове.Ако името не летището е Sofia, функцията ще създаде вектор ,от самолети със повече от 10000 литателни часове,и ще го върне там където е повикана функцията (main()).



**3.6 - void show\_french\_flights()** - функцията се използва в main() при създаването на обектите и файл конструктора в клас CAirport. Функцията извежда на екран броя на френските авиокомпании и техния броя полети.



**3.7** – Работа с файлове – Работа с файл се осъществява чрез експлицитния конструктор на CAirport. Във началото на конструктора се декларира променлива rollcontrol, която служи за разпределяне на данните от файла. Във променливата String line ще се запише щя ред данни от файлла .В зависимост от числото във rollcontrol, реда със данни ще отиде във болк от кода, предназначен за обработка на ред данни с определена структура.

Структура на файла – за да може да се използва файл от програмата ,той трябва да има спецефична структура:

Име на летището и номер на полета

Марка , брой часове и дестинации на самолета

дестинция и борй полети към нея

име на авиокомпанията , националност и брой полети

произволен стринг или чар, който се изполаза да покаче на програмата че дайла има още данни.

Пример:

sofia 34

Concorde 20500 33 Tu-160 700 15 Yak-40 5490 20 PC-6 15000 76

Moscow 57 Dalas 23 Melborne 30 Stockholm 77

IGavion French 56

n

KR860 1340 22 C919 700 2 porter 54500 20 falcon-x7 15000 33

London 77 Moscow 13 Prague 49 Melborne 88

Avdef French 23

Когато програмата стигне до ‘n’(ред 5), rollControl ще се стане 1 и програмата ще се повтори отново(без да записва нови данни за Име на летището и номер на полета)

Листинг на програмата с коментари

#include <iostream>

#include <string>

#include <map>

#include <iterator>

#include <list>

#include <fstream>//импортирам fstream за работа със файл

#include <vector>

#include <sstream>//импортитам sstream ,за обработрка на данните от файла

using namespace std;

class CPlane{//Създавам класът CPlane

string planeModel;//модел на самолета

int flightTime;//брой литателни часове

public:

//Constreuctors

CPlane(); //Дефалтен конструктор

//Експлицитен конструктор, приемащ аргументи за променливите -planeModel и flightTime

CPlane(string PM, int FT) {

planeModel = PM;

flightTime = FT;

}

//Setters & Getters

void set\_planeModel(string x){//Сетър за planeModel

planeModel = x;

}

string get\_planeModel() const {//Гетър за planeModel

return planeModel;

}

void set\_flightTime(int x) {//Сетър за flightTime

flightTime = x;

}

int get\_flightTime() const {//Гетър за flightTime

return flightTime;

}

//Оperators

bool operator<(const CPlane& obj) const { //Предефиниране на оператора за по-малко '<'

return flightTime > obj.flightTime;

}

//Оstream

friend ostream &operator <<(ostream& out, CPlane& obj); // Декларирам приятелска функция за потоков изход

};

ostream &operator <<(ostream& out, CPlane& obj) {//тялото на приятелската функция за потоков изход

return out <<endl<<"Plane model :"<< obj.planeModel <<" / Flight-time: "<< obj.flightTime;

}

class CAirtravel{//Създавам класът CAirtravel

public:

string compName;//име на компанията

string nationality;//националност на компанията

//map държи CPlane обекти като 'Key' и unsigned int(брой дестинаций) като 'Value'

map<CPlane, unsigned int>cplane\_map;

map<CPlane, unsigned int>::iterator iter2;//Итератор ,който итерира cplane\_map

//За да получа достъп до данните от клас CPlane използвам 2 вектора - v1 и v2.

//Разделям 'Key' и 'Value' на map cplane\_map и записвам разделените стойности във 2 вектора v1 и v2.

vector<CPlane>v1;//Във вектор v1 записвам 'Key' от тип cplane\_map (CPlane обекти)

//Във вектор v2 записвам 'Value' от cplane\_map (unsigned int - брой дестинаций)

vector<unsigned int>v2;

multimap<string, unsigned int>multiM\_destination\_container;//multimap държи string (дестинация) като 'Key' и unsigned int(брой полети към дестинацията) като 'Value'

multimap<string, unsigned int>::iterator iter3;//Итератор ,който итерира multiM\_destination\_container

//за по-лесен достъп. разделям 'Key' и 'Value' от multimap multiM\_destination\_container и записвам разделените стойности във 2 вектора, v3 и v4.

vector<string>v3;//Във вектор v1 записвам 'Key' от тип string (дестинация)

vector<unsigned int>v4;//Във вектор v2 записвам 'Value' от cplane\_map (unsigned int - брой полети към дестинацията)

static int French\_count;//static int следи броя на френските компании

static int Number\_of\_destinations;//броя на дестинациите на всички френски компании

CAirtravel();//Дефалтен конструктор.

//Експлицитен конструктор, приемащ аргументи за променливите -compName(име на компанията), nationality(националност на компанията),map - cplane\_map съдържащ(CPlane обекти и брой дестинаций) и multimap - multiM\_destination\_container съдържащ(дестинация и брой полети към нея)

CAirtravel(string compName\_Arg, string nationality\_Arg, map<CPlane, unsigned int>cplane\_map\_Arg,multimap<string, unsigned int> mp\_Arg) {

compName = compName\_Arg;

nationality = nationality\_Arg;

cplane\_map = cplane\_map\_Arg;

multiM\_destination\_container = mp\_Arg;

push\_in\_vector();//<- тази функция разделя 'Key' и 'Value' от map cplane\_map и записва разделените стойности във вектор v1 и v2.

push\_in\_vector2();//<- тази функция разделя 'Key' и 'Value' от multimap multiM\_destination\_container и записва разделените стойности във вектор v3 и v4.

if (nationality == "French" || nationality == "french") {

//проверявам дали новосъздадения обек ,притижава френска националност

frenchFlights();//изпълнява функцията frenchflights() и по този начин инкрементира static променливата "French\_count" и добавя броя на полетите от тази френска компания към static променливата "Number\_of\_destinations" ,която държи проя полети за всички френски компании

}

less\_than\_2k();

}

//тази функция се изъплнпва само когато обекта който създаваме е със френска националност

void frenchFlights() {//тази функция следи колко обекта притежаващи френска националност са създадени

map<CPlane, unsigned int>::iterator itr;

French\_count++;//увеличавам тоталния брой френски авиокомпланий със 1

for (itr = cplane\_map.begin(); itr != cplane\_map.end(); itr++) {

Number\_of\_destinations += itr->second;//добавям броя на дестинациите ,на новия обект към тоталния брой на всички дестинации от френски компаний

}

}

void push\_in\_vector() {//записвам данните от cplane\_map във двата вектора

iter2 = cplane\_map.begin();

for (int i = 0; i < cplane\_map.size(); i++) {//итерирам през cplane\_map

v1.push\_back(iter2->first);//записвам всички 'kays'(CPlane обекти) на cplane\_map във v1

v2.push\_back(iter2->second);//записвам всички 'values'(unsigned int - брой дестинации) на cplane\_map във v2

++iter2;//инкрементирам итератора ,и преминавам към следващата двойка от cplane\_map

}

}

void push\_in\_vector2() {//записвам данните от cplane\_map във двата вектора

iter3 = multiM\_destination\_container.begin();

for (int i = 0; i < multiM\_destination\_container.size(); i++) {//итерирам през multiM\_destination\_container

v3.push\_back(iter3->first);//записвам всички 'kays'(дестинация) на multiM\_destination\_container във v3

v4.push\_back(iter3->second);//записвам всички 'values'(unsigned int - брой полети към дестинацията) на multiM\_destination\_container във v4

++iter3;//инкрементирам итератора ,и пременавам към следващата двойка от multiM\_destination\_container

}

}

void less\_than\_2k() {//функция която зивежда на екран лист със всички самолети със по-малко летателни часове от 2000 и повече дестинации от 10

list<CPlane>P\_list;//Във този лист вкарвам CPlane обекти които имат по-малко летателни часове от 2000 и повече полети от 10

list<CPlane>::iterator p;//итератор за P\_list

vector<CPlane>::iterator Cplane\_iter;//итератор за Cplane обектите в вектор v1

vector<unsigned int>::iterator destination\_iter;//итератор за int (дестинациите) в вектор v2

destination\_iter = v2.begin();

for(Cplane\_iter = v1.begin(); Cplane\_iter != v1.end(); Cplane\_iter++) {//този цикъл ще се повтори докато итератора на cplane\_map не стигне края

if (Cplane\_iter->get\_flightTime() < 2000 && (\*destination\_iter)>10) {

P\_list.push\_back((\*Cplane\_iter));//във P\_list вкарвам Cplane обекти

}

destination\_iter++;

}

for (p = P\_list.begin(); p != P\_list.end(); p++) {//цикъл за извеждане на листа ,на екран

cout << (\*p);

}

}

bool operator<(const CAirtravel& obj) const {//Предефиниране на оператора за по-малко '<'

return compName > obj.compName;

}

};

class CAirport {//Създавам класът CAirport

//информация за летището

string airportName;//име на летището

unsigned int n\_flights;//номер на полет

multimap<CAirtravel, unsigned int>airtravel;//контейнер съдържащ данните за различните авиокомпании и брой полети

public:

CAirport();//Дефалтен конструктор.

//Експлицитен конструктор

CAirport(string airportName\_arg, unsigned int n\_flights\_arg, multimap<CAirtravel, unsigned int>airtravel\_arg) {//приема(име на летището, номер на полет ,контейнер съдържащ данните за различните авиокомпании и брой полети

airportName = airportName\_arg;

n\_flights = n\_flights\_arg;

airtravel = airtravel\_arg;

show\_french\_flights();//изкарвам френските полети на екран

}

CAirport(string filename) {//Експлицитен конструктор за файл

ifstream ObjFile(filename);//създавам файлов овект(ObjFile)

string airportName\_arg;// във airportName\_arg записвам име на летището (от файл)

unsigned int n\_flights\_arg;// във n\_flights\_arg записвам номер на полет на летището (от файл)

//запълвам тези 2 контейнера със данни от файла ,и ги вкарвам във CAirtravel обект

multimap<string, unsigned int>destinations\_container1;

map<CPlane, unsigned int>class1\_container1;

multimap<CAirtravel, unsigned int>Container\_for\_CAirport;//подавам Container\_for\_CAirport във конструктора на Airport

int rollControl = 0;//използвам rollControl за да покажа на програмата кой ред от текстовия документ със какво е пълен

//когато програмата прочете 1 ред от файла, rollControl се инкрементира

//в зависимост от числото във rollControl, програмата ще изпълни различен if statement

while (ObjFile >> airportName\_arg >> n\_flights\_arg) {//записвам името на летището и номера му във airportName\_arg и n\_flights\_arg

if (ObjFile) {//условието във if statementa ще се изпълнява докато има данни във файла

string line;//във line се записват данните от файла, ред по ред

while (getline(ObjFile, line)) {//този цикъл се повтаря докато показателя на файла не стигне края

rollControl++;//rollcontrol се инкрементира

istringstream iss(line);//създавам istringstream обект (iss) който копира данните от line(1 ред от файла)

if (rollControl == 2) {//ако rollControl е 2 ,то filepointera е на ред който съдържа данните за - марка на самолета , литателни часове и брой дестинации на самолета

while (iss) {//цикъла се повтаря докато показателя на iss не стигне края на реда

string subs, dubs, mubs;//дъздавам 3 променливи които се подават като аргумент за създаване и съхраняване на обект от тип CPlane във class1\_container1

//вкарвам стринговете от iss във новосъздадените променливи

iss >> subs;

iss >> dubs;

iss >> mubs;

if (subs == "") {//ако има празни променливи, ще прискочим тази итерация в цикъла

continue;

}

class1\_container1.insert(pair<CPlane, unsigned int>(CPlane(subs, stoi(dubs)), stoi(mubs)));

}

}

if (rollControl == 3) {//ако rollControl е 3 ,то filepointera е на ред който съдържа данните за - име на дестинацията и брой полети към нея

while (iss) {//цъкъла се повтаря докато pointer на iss не стигне края си

string subs, dubs; // дъздавам 2 променливи които се дъхраняват във map destinations\_container1

//вкарвам стринговете от iss във новосъздадените променливи

iss >> subs;

iss >> dubs;

if (subs == "") {//ако има празни променливи, ще прискочим тази итерация в цикъла

continue;

}

destinations\_container1.insert(pair<string, unsigned int>(subs, stoi(dubs)));

}

}

if (rollControl == 4) {//ако rollControl е 4 ,то показателя на файла е на ред който съдържа данните за -име, националност и брой дестинации на компанията

while (iss) {//цъкъла се повтаря докато pointer на iss не стигне края си

string subs, dubs, mubs;

//вкарвам стринговете от iss във новосъздадените променливи

iss >> subs;

iss >> dubs;

iss >> mubs;

if(!(subs==""|| subs == ""|| mubs == ""))//ако има празни променливи, ще прискочим тази итерация в цикъла

Container\_for\_CAirport.insert(pair<CAirtravel, unsigned int>(CAirtravel(subs, dubs, class1\_container1, destinations\_container1), stoi(mubs)));

}

}

if (rollControl == 5) {// ако rollControl е 5, ще преминем към създаването на нов Airtravel обект

//изтривам старите данни от контейнерите

destinations\_container1.clear();

class1\_container1.clear();

rollControl = 1;//rollControl става 1 и цикъла отново започва да взема от файла, докато не свършат

}

}

}

}

//присвояване на данни

airportName = airportName\_arg;

n\_flights = n\_flights\_arg;

airtravel = Container\_for\_CAirport;

show\_french\_flights();//изкарвам френските полети на екран

}

void Func1\_get\_flights\_and\_owner(string arg) {//при подаден аргумен ,марка самолет, веръща името на авиокомпанията собственик на самолета

bool checkifmatch = true;//болеан променливата се използава за проверка, дали има самолет със същата марка като тази подадена като аргумент

multimap<CAirtravel, unsigned int>::iterator iter;//създавам итератор за данните за различните авиокомпании и брой полети

for (iter = airtravel.begin(); iter != airtravel.end(); iter++) {//итерирам през различните airtravel обекти

for (int j = 0; j < iter->first.v1.size(); ++j) {//итерирам през различните CPlane обекти

if (iter->first.v1[j].get\_planeModel() == arg) {//ако модела на самолета съвпада със подадения аргумент, ще го извадя на екран

cout<<endl<< arg<<" - belongs to " << iter->first.compName <<" and has " << iter->first.v2[j] <<" flights";

checkifmatch = false;//checkifmatch става false ,което значи че има поне 1 самолет със подадения модел

}

}

}

if (checkifmatch)// ако не няма нито 1 самолет със подадения модел ще се изведе на екран съобщение че няма съвпадения

cout<< endl << "no panes with this model were found";

}

void Func2\_get\_max\_of\_destination(string arg) {//при подаден аргумен име на дестинация, връща имато на компанията със най-много полети към нея

bool checkifmatch = true;//болеан променливата се използава за проверка, дали има самолет със същата марка като тази подадена като аргумент

int savemax = 0;//запазва най-многото полети ,по който се прави проверката

string savename = "no complany with flights to destination";//запазва имато на компанията със най-много полети

multimap<CAirtravel, unsigned int>::iterator iter;//създавам итератор за данните за различните авиокомпании и брой полети

for (iter = airtravel.begin(); iter != airtravel.end(); iter++) {//итерирам през различните airtravel обекти

for (int j = 0; j < iter->first.v3.size(); ++j) {//итерирам през всички дестинации

if (arg == iter->first.v3[j]) {//когато дестинацията съвпадне със аргумента ще започне нова проверка

if (iter->first.v4[j] > savemax) {//тази нова преверка ще провери дали новата компания има повече извършени полети

savemax = iter->first.v4[j];//новата компания става 'компанията със най-много полети'

savename = iter->first.compName;

checkifmatch = false;//checkifmatch става false ,което значи че има поне една компанията със подадената дестинация

}

}

}

}

if (checkifmatch)// ако не няма нито 1 самолет със подадения модел ще изведем на екран съобщение че няма съвпадения

cout << endl << "no planes with this model were found";

else cout << savemax << " flights from " << savename;// изкарвам компанията със най-много полити

}

void Func3\_getPlane\_with\_max\_destinations() {//връща марка самолет със най-много дестинаций

multimap<CAirtravel, unsigned int>::iterator iter;//създавам итератор за данните за различните авиокомпании и брой полети

int savemax = 0;//запазва броя полети ,по който се прави проверката

string savename = "no plane";//запазва марката на самолета

for (iter = airtravel.begin(); iter != airtravel.end(); iter++) {//итерирам през различните airtravel обекти

for (int j = 0; j < iter->first.v1.size(); ++j) {//итерирам през различните CPlane обекти

if (iter->first.v2[j] > savemax) {//тази преверка ще провери дали новия CPlane обект има повече дестинации

savemax = iter->first.v2[j];//новата компания става 'самолет със най-много дестинаций'

savename = iter->first.v1[j].get\_planeModel();

}

}

}

cout <<endl << savename<<" with "<< savemax << " destinations ";// изкарвам самолета със най-много дестинации

}

//при подаден аргумент авиокомпания, връща дестинацията със най-много полети

void Func4\_get\_max\_flights\_destination\_by\_company(string arg) {

multimap<CAirtravel, unsigned int>::iterator iter;//създавам итератор за данните за различните авиокомпании и брой полети

int savemax = 0;//запазва броя полети към дестинация,по което се прави проверката

string savename = "no company";;//запазва името на дестинацията

for (iter = airtravel.begin(); iter != airtravel.end(); iter++) {//итерирам през различните airtravel обекти

if(iter->first.compName == arg){//проверка дали сегашната компания(в итератора) ствпада със подадения аргумант

for (int j = 0; j < iter->first.v1.size(); ++j) {//итерирам през различните CPlane обекти

if (iter->first.v4[j] > savemax) {//тази проверка ще провери дали броя дестинации е по-голям от предишния макс

savemax = iter->first.v4[j];

savename = iter->first.v3[j];

}

}

}

}

cout << endl << savemax << " " << savename;//извеждане на екран

}

vector<CPlane> Func5\_return\_vector\_of\_Cplanes() {//връща векрор от самолетите със над 10000 литатени часове които ползват летище София

vector<CPlane>localV1;//във този вектор вкарвам обекти от тип CPlane ,които имат повече от 10000 летатилни часове

if (airportName == "sofia"|| airportName == "Sofia") {//проверка дъли самолетите ползавт летище София

multimap<CAirtravel, unsigned int>::iterator iter;//създавам итератор за данните за различните авиокомпании и брой полети

for (iter = airtravel.begin(); iter != airtravel.end(); iter++) {//итерирам през различните airtravel обекти

for (int j = 0; j < iter->first.v1.size(); ++j) {//итерирам през различните CPlane обекти

if (iter->first.v1[j].get\_flightTime() > 10000) {//проверка дъли самолета има 10000 или повече летателни часове

localV1.push\_back(iter->first.v1[j]);//вкарвам CPlane обекта във вектор localV1

}

}

}

}

return localV1;//връща вектор от CPlane обекти

}

void show\_french\_flights() {//тази функция извежда броя на френските авиокомпании и техните полити

multimap<CAirtravel, unsigned int>::iterator iter;

iter = airtravel.begin();

cout << "\n\nNumber of French flights: " << iter->first.French\_count << "\nNumber of destinations by French companies: " << iter->first.Number\_of\_destinations;//изкарвам броя на всички френски компаний и броя на техните дестинации

}

};

//част от декларацията на static променливите отговорни за тоталния брой на френските авиокомпании

int CAirtravel::French\_count = 0;

int CAirtravel::Number\_of\_destinations = 0;

int main() {

int izbor=0;//променливата izbor позволява на потребителя да избира различни оцпий

do {//докато потребителя не избере нещо различно от 0 ,този цикъл ще се повтаря

cout << "Please enter the corresponding number\n1. Use hardcoded data\n2. Use file data\nInput any other number to exit program\n";

cin >> izbor;

} while (izbor == 0);

if (izbor == 1) {

//създавам контейнери за hardcoded данни

map<CPlane, unsigned int>class1\_container1;//конейнер за Atravelobj1

map<CPlane, unsigned int>class1\_container2;//конейнер за Atravelobj2

map<CPlane, unsigned int>class1\_container3;//конейнер за Atravelobj3

multimap<string, unsigned int>destinations\_container1;//конейнер за Atravelobj3

multimap<string, unsigned int>destinations\_container2;//конейнер за Atravelobj3

multimap<string, unsigned int>destinations\_container3;//конейнер за Atravelobj3

multimap<CAirtravel, unsigned int>Container\_for\_CAirport;

// МАРКА | ЛИТАТЕЛНИ ЧАСОВЕ

//данни за Atravelobj1

CPlane Cobj1("Boeing 777", 500);

CPlane Cobj2("Airbus a380", 1000);

CPlane Cobj3("Boeing 737", 15000);

CPlane Cobj4("Concorde", 2500);

//данни за Atravelobй2

CPlane Cobj5("falcon 7x", 11000);

CPlane Cobj6("Beriev Be30", 2500);

CPlane Cobj7("Pilatus porter", 21000);

CPlane Cobj8("Ilyushin il96", 800);

//данни за Atravelobj3

CPlane Cobj9("Sukhoi KR860", 7900);

CPlane Cobj10("Comac C939", 4500);

//данни за Atravelobj1

class1\_container1.insert(pair<CPlane, unsigned int>(Cobj1, 9));

class1\_container1.insert(pair<CPlane, unsigned int>(Cobj2, 36));

class1\_container1.insert(pair<CPlane, unsigned int>(Cobj3, 76));

class1\_container1.insert(pair<CPlane, unsigned int>(Cobj4, 31));

//данни за Atravelobj2

class1\_container2.insert(pair<CPlane, unsigned int>(Cobj5, 19));

class1\_container2.insert(pair<CPlane, unsigned int>(Cobj6, 45));

class1\_container2.insert(pair<CPlane, unsigned int>(Cobj7, 89));

class1\_container2.insert(pair<CPlane, unsigned int>(Cobj8, 47));

//данни за Atravelobj3

class1\_container3.insert(pair<CPlane, unsigned int>(Cobj9, 7));

class1\_container3.insert(pair<CPlane, unsigned int>(Cobj10, 12));

// ДЕСТИНАЦИЯ | БРОЙ ПОЛЕТИ

//данни за Atravelobj1

destinations\_container1.insert(pair<string, unsigned int>("Tokyo", 47));

destinations\_container1.insert(pair<string, unsigned int>("Varna", 51));

destinations\_container1.insert(pair<string, unsigned int>("London", 121));

destinations\_container1.insert(pair<string, unsigned int>("New York", 64));

//данни за Atravelobj2

destinations\_container2.insert(pair<string, unsigned int>("Tokyo", 19));

destinations\_container2.insert(pair<string, unsigned int>("London", 7));

destinations\_container2.insert(pair<string, unsigned int>("Mexico city", 2));

destinations\_container2.insert(pair<string, unsigned int>("Shenzhen", 22));

//данни за Atravelobj3

destinations\_container3.insert(pair<string, unsigned int>("Rome", 32));

destinations\_container3.insert(pair<string, unsigned int>("Shenzhen", 90));

cout << "\nlist of palnes with less than 2000 flight hours and more than 10 destinations:";

//създавам 3 CAirtravel обекта, които послсе вакарвам във контейнер Container\_for\_CAirport

CAirtravel Atravelobj1("French Bee", "French", class1\_container1, destinations\_container1);

CAirtravel Atravelobj2("Lufthansa", "German", class1\_container2, destinations\_container2);

CAirtravel Atravelobj3("Air France", "French", class1\_container3, destinations\_container3);

//PLANE ОБЕКТ | БРОЙ ПОЛЕТИ

Container\_for\_CAirport.insert(pair<CAirtravel, unsigned int>(Atravelobj1, 338));

Container\_for\_CAirport.insert(pair<CAirtravel, unsigned int>(Atravelobj2, 76));

Container\_for\_CAirport.insert(pair<CAirtravel, unsigned int>(Atravelobj3, 122));

CAirport AirportObj("Sofia", 467, Container\_for\_CAirport);

string StringArg;//StringArg се използва за подаване на аргументи от тип String към член функциите на клас CAirport

vector<CPlane>vectorF5;//съзадвам вектор и итератор, за работа със член функция Func5\_return\_vector\_of\_Cplanes()

vector<CPlane>::iterator VF5iter;

while (true){//докато потребителя не въведе 6 прогармата ще продължи

cout << "\n\nPlease enter the coresponding number:\n1. return owner and number of flights of a plane\n2. return the company with the most flights to a destination\n3. return the plane with the most destinations\n4. return the destination with the most flights by a given company\n5. return list of planes that have more than 10000 in airport Sofia\n6. exit program\n";

cin >> izbor;

switch (izbor)//този switch позвулява на потребителя да избира член-функция

{

case 1://при подаден аргумен марка самолет веръща имато на авиокомпанията собственик на самолета

cout << "Enter a plane model: ";

cin.ignore(256, '\n');

getline(cin, StringArg);//взимам текаста от целия ред

AirportObj.Func1\_get\_flights\_and\_owner(StringArg);

break;

case 2://при подаден аргумен име на дестинация, връща името на компанията със най-много полети до нея

cout << "Enter a destination: ";

cin.ignore(256, '\n');

getline(cin, StringArg);//взимам текста от целия ред

AirportObj.Func2\_get\_max\_of\_destination(StringArg);

break;

case 3://връща марка самолет със най-много дестинации

AirportObj.Func3\_getPlane\_with\_max\_destinations();

break;

case 4://при подаден аргумент виокомпания, връща дестинацията със най-много полети

cout << "Enter a company: ";

cin.ignore(256, '\n');

getline(cin, StringArg);//взимам текста от целия ред

AirportObj.Func4\_get\_max\_flights\_destination\_by\_company(StringArg);

break;

case 5://връща векрор от самолетите със над 10000 литатени часове които ползват летище София

vectorF5 = AirportObj.Func5\_return\_vector\_of\_Cplanes();

for (VF5iter = vectorF5.begin(); VF5iter != vectorF5.end(); VF5iter++) {//изкарвам на екран всички самолети със повече от 10000 литатени часове които ползват летище София

cout << (\*VF5iter);

}

break;

case 6://спира програмата

exit(1);

break;

default:

cout << "\nInvalid input";

break;

}

}

}

else if (izbor == 2) {

string StringArg;//StringArg се използва за подаване на аргументи от тип String към член функциите на клас Cairport и за избиране на файл при работа с файл

vector<CPlane>vectorF5;//съзадвам вектор и итератор, за работа със член-функция Func5\_return\_vector\_of\_Cplanes()

vector<CPlane>::iterator VF5iter;

cout << "please enter the name of the file, you wish to use:\n";

cin >> StringArg;

CAirport AirportObj(StringArg);//продавам стринг със имато на кайла към конструктора на AirportObj

while (true) {//докато потребителя не въведе 6 прогармата ще продължи

cout << "\n\nPlease enter the coresponding number:\n1. return owner and number of flights of a plane\n2. return the company with the most flights to a destination\n3. return the plane with the most destinations\n4. return the destination with the most flights by a given company\n5. return list of planes that have more than 10000 in airport Sofia\n6. exit program\n";

cin >> izbor;

switch (izbor)//този switch позвулява на потребителя да избира член-функция

{

case 1://при подаден аргумен марка самолет веръща имато на авиокомпанията собственик на самолета

cout << "Enter a plane model: ";

cin.ignore(256, '\n');

getline(cin, StringArg);//взимам текаста от целия ред

AirportObj.Func1\_get\_flights\_and\_owner(StringArg);

break;

case 2://при подаден аргумен име на дестинация, връща името на компанията със най-много полети до нея

cout << "Enter a destination: ";

cin.ignore(256, '\n');

getline(cin, StringArg);//взимам текста от целия ред

AirportObj.Func2\_get\_max\_of\_destination(StringArg);

break;

case 3://връща марка самолет със най-много дестинации

AirportObj.Func3\_getPlane\_with\_max\_destinations();

break;

case 4://при подаден аргумент виокомпания, връща дестинацията със най-много полети

cout << "Enter a company: ";

cin.ignore(256, '\n');

getline(cin, StringArg);//взимам текста от целия ред

AirportObj.Func4\_get\_max\_flights\_destination\_by\_company(StringArg);

break;

case 5://връща векрор от самолетите със над 10000 литатени часове които ползват летище София

vectorF5 = AirportObj.Func5\_return\_vector\_of\_Cplanes();

for (VF5iter = vectorF5.begin(); VF5iter != vectorF5.end(); VF5iter++) {//изкарвам на екран всички самолети със повече от 10000 литатени часове които ползват летище София

cout << (\*VF5iter);

}

break;

case 6://спира програмата

exit(1);

break;

default:

cout << "\nInvalid input";

break;

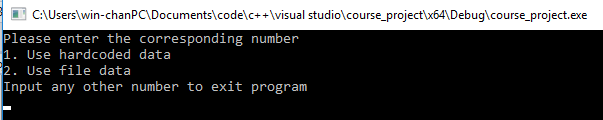
}}}

system("pause");

return 0;

}

Демонстрация на работата на програмата



Когато програмата стартира, ще даде избор на потребителя да избере дали да използва вече въведените данни или данни файл.

Ако потребителя въведе 1, то програмата ще използва вече въведените данни.

Ако въведе 2 ще им бъде позволено да въведът име на файл.

Програмат идва със 2 вече готови файла „fail.txt“и “fail2.txt”

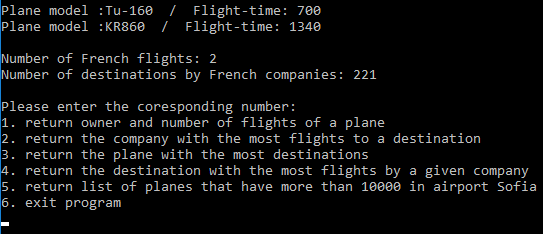


При въвеждане име на файл, потребителят трябва да използва правилна капитализация и да добави окончанието на файла (.txt).

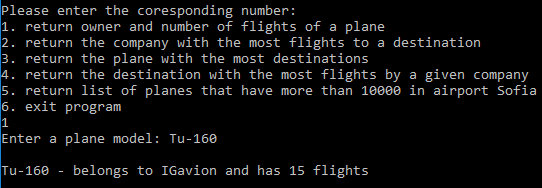
След въвеждане на файл или избиране да вече въведените данни, програмата ще изведе: Броя на всички самолети със по-малко летателни часове от 2000 и повече дестинации от 10 (less\_than\_2k() от CAirtravel).

Броя на всички френски авиокомпании и техния комбиниран брой полети (show\_french\_flights() от CAirport).

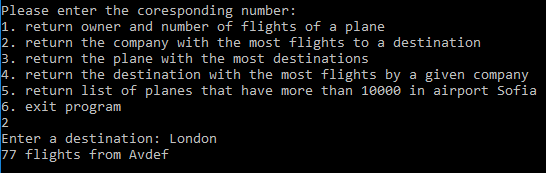
Потребителя получава опция за справка.При избирането да направи такава ще има опцията да избере измежду пет различни справки или при избирането на последната опция да спре програмата.



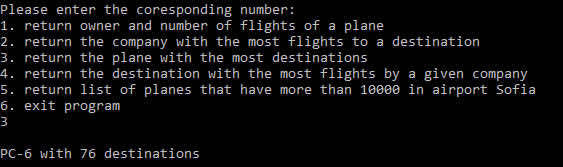
При избиране на справка 1 , потребителят ще трябва да въведе марка на самолет. Ако самолета съществува програмта ще изведе собственика.



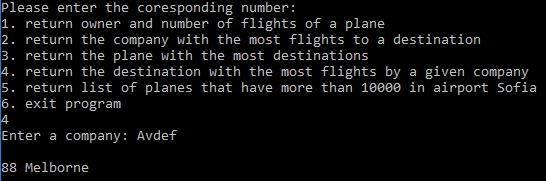
При избиране на справка 2 , потребителя ще трябва да въведе име на дестинация. Ако дестинацията съществува програмта ще изведе името на компанията със най-много полети до нея.



При избиране на справка 3 , програма извежда марката и броя полети на самолета със най-много полети.



При избиране на справка 4 , потребителя ще трябва да въведе име на компания. Ако компанията съществува, програмта ще изведе броя полети и името на дестинацията със най-много полети, за избраната програма.



При избиране на справка 5 , програмта ще изведе всички самолети със повече от 10000 литателни часова ,които използват летище София

