Картина, която съдържа текст, графична колекция

Описанието е генерирано автоматичноТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра „Софтуерни и интернет технологии“

**СЕМЕСТРИАЛНА ДОМАШНА РАБОТА**

по дисциплината „Базово програмиране”

на тема: „Информационна система Самолетни полети”

Вариант 350

|  |  |
| --- | --- |
| Изготвил: Васил Тодоров Станчев | Проверил: |
| Специалност: Киберсигурност |  |
| Група: 1 |  |
| Факултетен номер: 23621813 |  |

2024

Съдържание

[I. Задание на проекта 3](#_Toc120821980)

[II. Анализ на решението 4](#_Toc120821981)

[1. Структура за данните в програмата 4](#_Toc120821982)

[2. Реализация на условие A 4](#_Toc120821983)

[3. Реализация на условие B 4](#_Toc120821984)

[4. Реализация на условие C 5](#_Toc120821985)

[5. Реализация на условие D 5](#_Toc120821986)

[6. Реализация на условие E 6](#_Toc120821987)

[7. Реализация на условие F 7](#_Toc120821988)

[8. Реализация на условие … - допълнение първо 7](#_Toc120821989)

[9. Реализация на условие … - допълнение второ 8](#_Toc120821990)

[10. Реализация на допълнение трето 8](#_Toc120821991)

[III. Упътване за употреба 9](#_Toc120821992)

[1. Впишете съответната част от проекта 9](#_Toc120821993)

[2. Впишете съответната част от проекта 9](#_Toc120821994)

[IV. Примерно действие на програмата 10](#_Toc120821995)

[1. Условие A 10](#_Toc120821996)

[2. Условие B 10](#_Toc120821997)

[3. Условие C 10](#_Toc120821998)

[4. Условие D 10](#_Toc120821999)

[5. Условие E 10](#_Toc120822000)

[6. Условие F 11](#_Toc120822001)

[7. Допълнение първо 11](#_Toc120822002)

[8. Допълнение второ 11](#_Toc120822003)

[9. Допълнение трето 11](#_Toc120822004)

Задание на проекта

Да се напише компютърна програма, реализираща информационна система, която поддържа полети на летище. Програмата съхранява и обработва данни заминаващи полети (номер на полета, дестинация, име на пилот, основна цена на полета, пътник първа/втора класа (true/false), дата на полета, име на пътника).

**Базова задача – сложност ниска**

1. Меню за избор на функциите в програмата.

**Функции от програмата са:**

1. Добавяне на полети
   1. Добавяне по един полет. Добавяне на списък с полети. Въвежда се цяло число n и след него n на брой полети.
2. Извеждане на екрана
   1. Извеждане на всички полети в оформен вид
3. Търсене и извеждане на екрана
   1. Извеждане на всички полети с най-ниска цена
   2. Извеждане на полети на даден пилот
4. Подреждане на основния масив с полети. Подреждане на полетите в низходящ ред по азбучен ред на дестинацията.
5. Управление на файл
   1. Извеждане на масива от полети във файл (двоичен)
   2. Въвеждане на масива от полети от файл (двоичен)

**Допълнение първо (+ базова задача)**

1. Създайте подменю, в което се влиза от основното, с нови функции за:
   1. Извеждане на полетите във възходящ ред на цената на полета и класата на полета, без да се променя основния масив
   2. Търсене и извеждане на полети по въведена дата и определена дестинация.

**Допълнение второ (+ базова задача)**

1. Изчисляване на текуща цена на полета
   1. Въвежда се дата на заминаване, извежда се номер на полет с основната цена за първа или втора класа. Ако до полета има повече от 20 дни, цената на билета е 50% от основната цена, ако има по малко от 20 дни - цената е 75% от реалната цена, ако остават по малко от 5 дни, цената на билета е 100% от реалната цена, ако остават по малко от 24 часа цената на билета е с 20% по висока от основната цена. (Приема се че един месец е 30 дни);
   2. При желание е възможна замяна на билет от втора класа с такъв от първа класа. Въвежда се номер на полета и името на пътника.
2. Отказ от полет
   1. По номер на полет и име на пътник, пътникът се отказва от полет, мястото му се освобождава и се възстановяват, 100% от цената ако билета е върнат 20 дни преди полета, 75% ако е върнат по малко от 20 дни преди полета, 50% ако е върнат 5 дни преди заминаване на полета (Извежда се сумата за връщане);

**Допълнение трето (+ базова задача)**

1. Данните в програмата да се попълват автоматично от файл при стартиране и да се записват автоматично във файл при затваряне на програмата.

Анализ на решението

Структура за данните в програмата

|  |  |  |
| --- | --- | --- |
| Структура | Обяснение | Примерени стойности |
| struct Flight {  string id;  string destination;  string nameOfPilot;  string nameOfPassenger;  float basePrice;  bool firstClass = false;  bool secondClass = false;  Date date;  }; | * id – поле за номер на полета * destination – поле за дестинация на полета * nameOfPilot – поле за име на пилота * nameOfPassenger – поле за име на пътник * basePrice – поле за основна цена на полет * firstClass – поле за стойност лъжа/истина за първа класа * secondCLass - поле за стойност лъжа/истина за втора класа * date – поле за дата на полета | 2343  Burgas  Ivan Petrov  Vasil Stanchev  23.253  True  False  {2, 12, 2004} |

Реализация на условие A

Анализ на алгоритъма, който трябва да се реализира

Алгоритъма реализира избора на различните възможности на системата за полети. Този избор се реализира чрез меню в което се въвеждат две променливи за избор. Едната е за главното меню, а другата е за избор 5 в който има допълнително меню за допълнителни възможности. Броя на възможностите е 10 за всяка от 9-те условия и 10 е за изход, като има и “default” случай, който служи ако потребителят реши да въведе друг избор, който не е зададен, и да изведе съобщение.

Блок схема на алгоритъма

#### A screenshot of a cell phone Description automatically generated

Функция, с която е реализиран алгоритъма

bool menu(Flight arr[], int& n)

{

int choice1 = 0, choice2 = 0;

while (choice1 != 10)

{

cout << "\tТова е система за полети.\nИзберете от следните възможности:" << endl;

cout << "1. Добавяне на полети" << endl;

cout << "2. Показване на полети" << endl;

cout << "3. Показване на полетите със най-ниска основна цена" << endl;

cout << "4. Търсене на определен полет спрямо име на пилот" << endl;

cout << "5. Показване на допълнителни възможности" << endl;

cout << "6. Показване на текуща цена спрямо дата на заминаване" << endl;

cout << "7. Промяна на класа на полета спрямо номер на полета и име на пътника" << endl;

cout << "8. Изтриване на полет и показване на сума за връщане" << endl;

cout << "9. Запис във файл" << endl;

cout << "10. Изход от програма" << endl;

cout << "Вкарване на избор:" << endl;

cin >> choice1;

if (cin.fail())

{

cin.clear();

cin.ignore((std::numeric\_limits<std::streamsize>::max)(), '\n');

cin >> choice1;

}

cin.clear();

switch (choice1)

{

case 1:

enterFlights(arr, n);

sortArr(arr, n);

break;

case 2:

showFlights(arr, n);

break;

case 3:

showMinPricedFlights(arr, n);

break;

case 4:

showPilotFlights(arr, n);

break;

case 5:

while (choice2 != 3)

{

cout << "1. Показване на сортираните полети спрямо цена и клас" << endl;

cout << "2. Търсене на полет спрямо дата и дестинация" << endl;

cout << "3. Връщане обратно във главното меню" << endl;

cout << "Вкарване на избор:" << endl;

cin >> choice2;

if (cin.fail())

{

cin.clear();

cin.ignore((std::numeric\_limits<std::streamsize>::max)(), '\n');

cin >> choice1;

}

switch (choice2)

{

case 1:

showSortedAscendingPriceAndClass(arr, n);

break;

case 2:

showDateAndPilotFlights(arr, n);

break;

case 3:

break;

default:

cout << "Въведете правилен избор" << endl;

break;

}

}

break;

case 6:

calculateCurrentFlightPrice(arr, n);

break;

case 7:

changePassengerClass(arr, n);

break;

case 8:

removeFlight(arr, n);

break;

case 9:

writeInFile(arr, n);

cout << "Записано във файл!"<<endl;

break;

case 10:

cout << "Довиждане!" << endl;

return false;

break;

default:

cout << "Въведете правилен избор" << endl;

break;

}

}

return true;

}

Следната функция представя описания алгоритъм, като към него е прибавено проверка за променливите за избор на менюто.

#### Входни данни на функцията

Входните данни са параметрите към функцията, масива съдържащ полетите и променлива съдържаща неговия размер, и опциите свързани към менюто.

#### Изходни данни на функцията или данни, които се извеждат

Функцията ще изведе възможните опции на менюто и следователно след избора на опция ще се покажат изходните данни на съответната опция, сочеща към функия, която изпълнява това действие.

Реализация на условие B

Анализ на алгоритъма, който трябва да се реализира

Алгоритъма реализира добавянето на списък с полети, като броя е определен от потребителя. Върти се цикъл, в който се добавят полетите поле по поле.

Блок схема на алгоритъма

#### A screenshot of a computer Description automatically generated

Функция, с която е реализиран алгоритъма

void enterFlights(Flight arr[], int& n)

{

char answerTicketClass;

bool flag = false;

int newElementsCount;

cout << "Вкарайте броя полети, които искате да вкарате:" << endl;

cin >> newElementsCount;

cout << "Вкарайте полетата във полетите:" << endl;

for (int i = n; i < n + newElementsCount; i++)

{

cout << "Вкарайте дестинация:" << endl;

validateString(arr[i].destination);

cout << "Вкарайте номер на полета:" << endl;

getline(cin, arr[i].id);

cout << "Вкарайте име на пилот:" << endl;

validateString(arr[i].nameOfPilot);

cout << "Вкарайте име на пътник:" << endl;

validateString(arr[i].nameOfPassenger);

cout << "Вкарайте основна цена на полета" << endl;

cin >> arr[i].basePrice;

cout << "Вашия билет първа класа ли е? Да или не / Y(y) или N(n)" << endl;

flag = false;

while (flag != true)

{

cin >> answerTicketClass;

cin.ignore((std::numeric\_limits<std::streamsize>::max)(), '\n');

switch (answerTicketClass)

{

case 'Y':

case 'y':

arr[i].firstClass = true;

flag = true;

break;

case 'N':

case 'n':

arr[i].secondClass = true;

flag = true;

break;

default:cout << "Вкарайте валиден вход" << endl;

break;

}

}

cout << "Вкарайте дата на полета: ден, месец и година:" << endl;

validateDate(arr[i].date);

}

n = n + newElementsCount;

}

Следната функция реализира алгоритъма описан по-горе, като добавя валидация за всички полета с изключение за номера на полета. За входа на класата на дадения полет е използван цикъл, с който се проверява дали потребителя иска полетът му да бъде вписан като първа класа или не.

#### Входни данни на функцията

Входните данни биват променлива за броя на полети за регистриране и вкарването на самите данни на полетите.

#### Изходни данни на функцията или данни, които се извеждат

Функцията има само изходни съобщения за информация на потребителя какво трябва да въведе в конзолата.

Реализация на условие C

Анализ на алгоритъма, който трябва да се реализира

Алгоритъмът реализира извеждане на въведените полети. Върти се цикъл, в който се показват полет по полет.

Блок схема на алгоритъма

A black background with white squares and black text

Description automatically generated

Функция, с която е реализиран алгоритъма

void showFlights(Flight arr[], int& n)

{

string classType;

cout << "Лист с полети: " << endl;

cout << "НОМЕР НА ПОЛЕТА | ДЕСТИНАЦИЯ | ПИЛОТ | ПЪТНИК | ОСНОВНА ЦЕНА | КЛАСА | ДАТА" << endl;

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

{

classType = (arr[i].firstClass == true) ? "Първа класа" : "Втора класа";

cout << arr[i].id << " | " << arr[i].destination << " | " << arr[i].nameOfPilot << " | " << arr[i].nameOfPassenger

<< " | " << arr[i].basePrice << " | " << classType << " | " << arr[i].date.day << "."

<< arr[i].date.month << "." << arr[i].date.year << endl;

}

}

Следната функция реализира описания алгоритъм, като форматира изходните данни да са подредени под формата на таблица, за по-лесен изглед за потребителя.

#### Входни данни на функцията

Входните данни са параметрите на функцията: масивът и променливата с неговия размер.

#### Изходни данни на функцията или данни, които се извеждат

Изходните данни са всичките полети, които се изписват на конзолата подредени в табличен вид.

Реализация на условие D

Анализ на алгоритъма, който трябва да се реализира

1. Алгоритъмът реализира намирането на най-малката основна цена в масива. Това се постига чрез въртенете на цикъл в който се намира най-малкия елемент с помощта на променливата “min”, като освен това цикълът пълни нов масив, в който се пазят индексите на най-малката цена, тъй като тя може да бъде присъства в множество полети. След този цикъл се върти втори, който извежда тези полети с помощта на масива за индекси.
2. Алгоритъмът реализира намирането на полети принадлежащи към поданено име на пилот. Върти се цикъл, в който се проверява дали името съвпада с такова в масива, като ако то съвпада се изписва на конзолата.

Блок схема на алгоритъма

1. A black background with white rectangles

   Description automatically generated
2. A screenshot of a computer

   Description automatically generated

Функция с която е реализиран алгоритъма

1. Следната функция реализира алгоритъма, форматира изходните данни да са подредени под формата на таблица, за по-лесен изглед за потребителя.

void showMinPricedFlights(Flight arr[], int& n)

{

Flight min = arr[0];

string classType;

int indexes[20];

int count = 0;

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

{

if (arr[i].basePrice < min.basePrice)

{

min = arr[i];

count = 0;

}

if (arr[i].basePrice == min.basePrice)

{

indexes[count++] = i;

}

}

cout << "Лист с полети: " << endl;

cout << "НОМЕР НА ПОЛЕТА | ДЕСТИНАЦИЯ | ПИЛОТ | ПЪТНИК | ОСНОВНА ЦЕНА | КЛАСА | ДАТА" << endl;

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

{

classType = (arr[indexes[i]].firstClass == true) ? "Първа класа" : "Втора класа";

cout << arr[indexes[i]].id << " | " << arr[indexes[i]].destination << " | " << arr[indexes[i]].nameOfPilot << " | " << arr[indexes[i]].nameOfPassenger

<< " | " << arr[indexes[i]].basePrice << " | " << classType << " | " << arr[indexes[i]].date.day << "."

<< arr[indexes[i]].date.month << "." << arr[indexes[i]].date.year << endl;

}

}

1. Следната функция реализира алгоритъма, като добавя проверка при условие, че не бъдат намерени полети свързани с пилота.

void showPilotFlights(Flight arr[], int& n)

{

string name;

string classType;

bool flag = false;

cout << "Вкарайте име на пилот:" << endl;

validateString(name);

cout << "Лист с полети свързан с даденото име на пилота: " << endl;

cout << "НОМЕР НА ПОЛЕТА | ДЕСТИНАЦИЯ | ПИЛОТ | ПЪТНИК | ОСНОВНА ЦЕНА | КЛАСА | ДАТА" << endl;

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

{

classType = (arr[i].firstClass == true) ? "Първа класа" : "Втора класа";

if (arr[i].nameOfPilot == name)

{

cout << arr[i].id << " | " << arr[i].destination << " | " << arr[i].nameOfPilot << " | " << arr[i].nameOfPassenger

<< " | " << arr[i].basePrice << " | " << classType << " | " << arr[i].date.day << "."

<< arr[i].date.month << "." << arr[i].date.year << endl;

flag = true;

}

}

if (flag == false)

{

cout << "Няма свързани полети с дадения пилот" << endl;

}

}

#### Входни данни на функцията

И двете функции взимат входни данни масива и неговия размер като параметри, като подточка b) взима променлива за името на пилота

#### Изходни данни на функцията или данни, които се извеждат

И двете функции извеждат масиви с полети, които се изписват на конзолата подредени в табличен вид.

Реализация на условие E

Анализ на алгоритъма, който трябва да се реализира

Алгоритъмът реализира сортиране на масив чрез метода на мехурчето, като го подрежда в низходящ ред по азбучен ред на дестинацията.

Блок схема на алгоритъма

A screenshot of a computer

Description automatically generated

Функция с която е реализиран алгоритъма

void sortArr(Flight arr[], int& n)

{

bool swap;

char first, second;

for (int i = 0; i < n - i; i++)

{

swap = false;

for (int j = 0; j < n - i - 1; j++)

{

first = (char)arr[j].destination[0];

second = (char)arr[j + 1].destination[0];

if (compareCharacters(first, second))

{

Flight temp = arr[j];

arr[j] = arr[j+1];

arr[j + 1] = temp;

swap = true;

}

}

if (swap == false)

break;

}

}

bool compareCharacters(char a, char b) {

return tolower(a) < tolower(b);

}

Следните функции реализират алгоритъма, като към главната функция е прибавена проверка за завъртане с цел оптимизация на алгоритъма. Втората функция прави проверка дали ASCII кода на първата буква е по-малкък от следващата. Тези букви се държат в две променливи, които взимат първата буква на текущия елемент и на следващия.

#### Входни данни на функцията

На главната функция входните данни са масива и неговия размер, а пък на втората са две букви.

#### Изходни данни на функцията или данни, които се извеждат

Главната функция не извежда нищо, тъй като тя само сортира. Втората връща истина или лъжа спрямо това дали първия ASCII код е по-малък от следващия.

Реализация на условие F

Анализ на алгоритъма, който трябва да се реализира

а) Алгоритъмът реализира записът на масив във двоичен файл. Върти се цикъл, в който се записват поле по поле полетите във файла, като за стринговете се запазва и техния размер, разположен точно преди тях. Полетата от масива се кастват като масив от символи.

b) Алгоритъмът реализира четенето на масив от двоичен файл. Върти се цикъл, в който се чете от файл като за стринговете се взима първо техния размер и се отделия място в C стринг, и после този С стринг се превръща в такъв от класа. Четенето за останалите поле е нормално, като само самия поток се каства като масив от символи.

Блок схема на алгоритъма

1. A screenshot of a phone

   Description automatically generated
2. A screenshot of a phone

   Description automatically generated

Функция с която е реализиран алгоритъма

a)

void writeInFile(Flight arr[], int& n)

{

ofstream outfile("myfile.bin", ios\_base::binary|ios\_base::beg);

if (outfile.fail())

{

cout << "Файлът не може да бъде отворен" << endl;

}

else

{

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

{

int sizeId = arr[i].id.size();

const char\* strId = arr[i].id.c\_str();

int sizeDestination = arr[i].destination.size();

const char\* strDestination = arr[i].destination.c\_str();

int sizePilot = arr[i].nameOfPilot.size();

const char\* strPilot = arr[i].nameOfPilot.c\_str();

int sizePassenger = arr[i].nameOfPassenger.size();

const char\* strPassenger = arr[i].nameOfPassenger.c\_str();

outfile.write((char\*)&sizeId, sizeof(int));

outfile.write(strId, sizeId + 1);

outfile.write((char\*)&sizeDestination, sizeof(int));

outfile.write(strDestination, sizeDestination + 1);

outfile.write((char\*)&sizePilot, sizeof(int));

outfile.write(strPilot, sizePilot + 1);

outfile.write((char\*)&sizePassenger, sizeof(int));

outfile.write(strPassenger, sizePassenger + 1);

outfile.write((char\*)&arr[i].basePrice, sizeof(float));

outfile.write((char\*)&arr[i].firstClass, sizeof(bool));

outfile.write((char\*)&arr[i].secondClass, sizeof(bool));

outfile.write((char\*)&arr[i].date.day, sizeof(int));

outfile.write((char\*)&arr[i].date.month, sizeof(int));

outfile.write((char\*)&arr[i].date.year, sizeof(int));

outfile.flush();

}

}

outfile.close();

}

b)

void openFromFile(Flight arr[], int& n)

{

ifstream infile("myfile.bin", ios\_base::binary | ios\_base::beg);

if (infile.fail())

{

cout << "Файлът не може да бъде отворен" << endl;

}

else

{

int sizeId, sizeDestination, sizePilot, sizePassenger;

char\* id;

char\* destination;

char\* namePilot;

char\* namePassenger;

while (infile.peek() != EOF)

{

infile.read((char\*)&sizeId, sizeof(int));

id = (char\*)calloc(sizeId + 1, 1);

infile.read(id, sizeId + 1);

arr[n].id = id;

infile.read((char\*)&sizeDestination, sizeof(int));

destination = (char\*)calloc(sizeDestination + 1, 1);

infile.read(destination, sizeDestination + 1);

arr[n].destination = destination;

infile.read((char\*)&sizePilot, sizeof(int));

namePilot = (char\*)calloc(sizePilot + 1, 1);

infile.read(namePilot, sizePilot + 1);

arr[n].nameOfPilot = namePilot;

infile.read((char\*)&sizePassenger, sizeof(int));

namePassenger = (char\*)calloc(sizePassenger + 1, 1);

infile.read(namePassenger, sizePassenger + 1);

arr[n].nameOfPassenger = namePassenger;

infile.read((char\*)&arr[n].basePrice, sizeof(float));

infile.read((char\*)&arr[n].firstClass, sizeof(bool));

infile.read((char\*)&arr[n].secondClass, sizeof(bool));

infile.read((char\*)&arr[n].date.day, sizeof(int));

infile.read((char\*)&arr[n].date.month, sizeof(int));

infile.read((char\*)&arr[n].date.year, sizeof(int));

n++;

}

}

infile.close();

}

И двете функции реализират алгоритмите, като е добавено само проверка дали самия файл може да бъде отворен.

#### Входни данни на функцията

И за двете функции входните данни са масив и неговия размер, параметрите на функциите, като за четенето от файл се използват данните от него, които биват пренесени в масива.

#### Изходни данни на функцията или данни, които се извеждат

И двете функции не връщат резултат, като за изходни данни на записа във файл се възприема файлът, в който ще се пазят данните.

Реализация на условие … - допълнение първо

Анализ на алгоритъма, който трябва да се реализира

1. Алгоритъмът реализира извеждане на полетите във възходящ ред на цената на полета и класата на полета, като не засяга главния масив, а създава копие. Това се случва чрез две сортирания чрез метода на мехурчето, като първо се сортира по цена и след това по класа. Като на края сортирания масив се извежда на екрана.
2. Алгоритъмът реализира търсенето и извеждането на полети по въведена дата и определена дестинация. Върти се цикъл, в който по въведена дата и дестинация се проверява дали има съответствия в масива, като ако има те се извеждат на екрана.

Блок схема на алгоритъма

1. A black background with white squares

   Description automatically generated
2. A black background with white squares

   Description automatically generated

Функция с която е реализиран алгоритъма

а) void showSortedAscendingPriceAndClass(Flight arr[], int& n)

{

Flight newArr[ARRSIZE\_MAX];

bool swap;

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

{

newArr[i] = arr[i];

}

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

{

swap = false;

for (int j = 0; j < n - i - 1; j++)

{

if (newArr[j].basePrice > newArr[j + 1].basePrice)

{

Flight temp = newArr[j];

newArr[j] = newArr[j + 1];

newArr[j + 1] = temp;

swap = true;

}

}

if (swap == false)

break;

}

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

{

swap = false;

for (int j = 0; j < n - i - 1; j++)

{

if ((newArr[j].firstClass == false) && (newArr[j + 1].firstClass == true))

{

Flight temp = newArr[j];

newArr[j] = newArr[j + 1];

newArr[j + 1] = temp;

swap = true;

}

}

if (swap == false)

break;

}

showFlights(newArr, n);

}

b)

void showDateAndPilotFlights(Flight arr[], int& n)

{

Date searchedDate;

string searchedDestination;

bool flag = false;

string classType;

cout << "Вкарайте дата: ден, месец и дата" << endl;

validateDate(searchedDate);

cout << "Вкарайте дестинация" << endl;

validateString(searchedDestination);

cout << "Лист с полетите на съответната дата и дестинация:" << endl;

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

{

classType = (arr[i].firstClass == true) ? "Първа класа" : "Втора класа";

if ((searchedDate.day == arr[i].date.day) && (searchedDate.month == arr[i].date.month) && (searchedDate.year == arr[i].date.year))

{

if (searchedDestination == arr[i].destination)

{

cout << arr[i].id << " | " << arr[i].destination << " | " << arr[i].nameOfPilot << " | " << arr[i].nameOfPassenger

<< " | " << arr[i].basePrice << " | " << classType << " | " << arr[i].date.day << "."

<< arr[i].date.month << "." << arr[i].date.year << endl;

flag = true;

}

}

}

if (flag == false)

{

cout << "Няма свързани полети на съответната дата и дестинация" << endl;

}

}

И двете функции реализират алгоритмите. Към първата е добавена оптимизация за алгоритма за сортиране, а на втората е добавена проверка при условие, че не бъдат намерени полети свързани с датата и дестинацията.

#### Входни данни на функцията

И за двете функции входните данни са масив и неговия размер, параметрите на функциите, като за втората има променливи за дата и дестинация.

#### Изходни данни на функцията или данни, които се извеждат

И двете функции извеждат масиви с полети, които се изписват на конзолата подредени в табличен вид.

Реализация на условие … - допълнение второ

Анализ на алгоритъма, който трябва да се реализира

1. Алгоритъмът реализира изчислението на текуща цена спрямо дата на заминаване, като цената бива изчислявана по различни критерии според това колко дни остават до полета. Върти се цикъл, в който за всеки полет се изчислява текущата цена, като ако датата е след дадения полет, то се изписва, че той е пропуснат. Изкарват се данните на масива в табличен вид.
2. Алгоритъмът реализира промяната на пътник втора класа с такава с първа. Това се случва спрямо номер на полета и името на пътника. Върти се цикъл и ако има съвпадение на номера и името на пътника, му се задава въпрос за промяна на името и според отговора се определя дали ще има промяна или не.

Блок схема на алгоритъма

1. A screenshot of a computer screen

   Description automatically generated
2. A screenshot of a computer

   Description automatically generated

Функция с която е реализиран алгоритъма

a)

void calculateCurrentFlightPrice(Flight arr[], int& n)

{

Date departure;

string classType;

int departureDays = 0, flightDepartureDays = 0,difference;

float currentPrice = 0;

cout << "Вкарайте дата: ден, месец и година" << endl;

validateDate(departure);

cout << "Лист с полети с основната цена и новата спрямо посочената дата:" << endl;

cout << "НОМЕР НА ПОЛЕТА | ДЕСТИНАЦИЯ | ПИЛОТ | ПЪТНИК | ОСНОВНА ЦЕНА | НОВА ЦЕНА | КЛАСА | ДАТА НА ПОЛЕТА | ДАТА НА ПЪТНИКА" << endl;

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

{

classType = (arr[i].firstClass == true) ? "Първа класа" : "Втора класа";

departureDays = departure.day + (departure.month \* 30) + ((departure.year - 1) \* 360);

flightDepartureDays = arr[i].date.day + (arr[i].date.month \* 30) + ((arr[i].date.year - 1) \* 360);

difference = flightDepartureDays - departureDays;

if (difference > 20)

{

currentPrice = 0.5 \* arr[i].basePrice;

}

else if (difference <= 20 && difference > 5)

{

currentPrice = 0.75 \* arr[i].basePrice;

}

else if (difference <= 5 && difference > 1)

{

currentPrice = arr[i].basePrice;

}

else if (difference == 1)

{

currentPrice = (0.2 \* arr[i].basePrice) + arr[i].basePrice;

}

else

{

cout << "Полета е пропуснат" << endl;

continue;

}

cout << arr[i].id << " | " << arr[i].destination << " | " << arr[i].nameOfPilot << " | " << arr[i].nameOfPassenger

<< " | " << arr[i].basePrice << " | " << currentPrice << " | " << classType << " | " << arr[i].date.day << "."

<< arr[i].date.month << "." << arr[i].date.year << " | " << departure.day << "." << departure.month << "."

<< departure.year << endl;

}

}

b)

void changePassengerClass(Flight arr[], int& n)

{

char choice;

bool flag1 = false, flag2 = false;

string searchedId, searchedName;

cout << "Вкарайте име на пътника:" << endl;

validateString(searchedName);

cout << "Вкарайте номер на полета:" << endl;

getline(cin, searchedId);

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

{

if (searchedId == arr[i].id && searchedName == arr[i].nameOfPassenger)

{

cout << "Искате ли да промените билета си на първа класа?" << endl;

cout << "Отговор Да/Не / Y(y)/N(n):" << endl;

while (flag1 != true)

{

cin >> choice;

cin.ignore((std::numeric\_limits<std::streamsize>::max)(), '\n');

switch (choice)

{

case 'Y':

case 'y':

arr[i].firstClass = true;

flag1 = true;

break;

case 'N':

case 'n':

arr[i].secondClass = true;

flag1 = true;

break;

default:cout << "Въведете валиден вход" << endl;

break;

}

}

flag2 = true;

}

}

if (flag2 == false)

cout << "Данните не съответстват" << endl;

else

showFlights(arr, n);

}

И двете функции реализират алгоритмите. При втората функция е добавено и проверка при условие, че пътника не е открит.

#### Входни данни на функцията

И за двете функции входните данни са масив и неговия размер, параметрите на функциите. При първата се вкарва също и дата, а при втората номер на полет и име на пътник.

#### Изходни данни на функцията или данни, които се извеждат

И за двете функции на конзолата се изкарва масивът с полети подреден в табличен вид. В първата функция се изкарва също, ако дадения полет е пропуснат.

Реализация на допълнение I(Отказ от полет)

Анализ на алгоритъма, който трябва да се реализира

Алгоритъмът реализира отказът на полет и неговата върната цена. Отказът се определя спрямо номер на полет и име на пътник. Върти се цикъл, в който спрямо съвпадение на данните се изчисляват оставащите дни до полета и спрямо тях се изчеслява цената за връщане. След това се върти втори цикъл, в който се изтрива въпросния полет като на негово място осталите полети след него минават с един назад. След края на цикъла се прави проверка дали данните са били открити, дали датата е съвпаднала, и ако всичко е било наред се извежда сумата за връщане.

Блок схема на алгоритъма

A black and white screen shot of a diagram

Description automatically generated

Функция с която е реализиран алгоритъма

void removeFlight(Flight arr[], int& n)

{

string searchedId, searchedName;

bool flag1 = false, flag2 = false;

cout << "Вкарайте име на пътника:" << endl;

validateString(searchedName);

cout << "Вкарайте номер на полета:" << endl;

getline(cin, searchedId);

cout << "Вкарайте дата: ден, месец и година" << endl;

Date returnDate;

validateDate(returnDate);

int returnDays = 0, flightDepartureDays = 0, difference;

float returnMoney = 0;

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

{

returnDays = returnDate.day + (returnDate.month \* 30) + ((returnDate.year - 1) \* 360);

flightDepartureDays = arr[i].date.day + (arr[i].date.month \* 30) + ((arr[i].date.year - 1) \* 360);

difference = flightDepartureDays - returnDays;

if (searchedId == arr[i].id && searchedName == arr[i].nameOfPassenger)

{

if (difference >= 20)

{

returnMoney = arr[i].basePrice;

flag2 = true;

}

else if (difference < 20 && difference > 5)

{

returnMoney = 0.75 \* arr[i].basePrice;

flag2 = true;

}

else if (difference < 5 && difference > 0)

{

returnMoney = 0.5 \* arr[i].basePrice;

flag2 = true;

}

n = n - 1;

for (int j = i; j < n; j++)

{

arr[j] = arr[j + 1];

}

flag1 = true;

break;

}

}

if (flag1 == false)

cout << "Данните не съответстват" << endl;

else if (flag1 == true && flag2 == false)

cout << "Полетът е пропуснат" << endl;

else

cout << "Сума за връщане: " << returnMoney << endl;

}

Следната функция реализира алгоритъмът.

#### Входни данни на функцията

Входните данни са параметрите на функцията, масив и неговата дължина, и променливи отнасящи се към името на пътника, номера на полета и дата.

#### Изходни данни на функцията или данни, които се извеждат

Изходните данни са сумата за връщане, изписана на конзолата.

Допълнение трето

Реализацията на третото допълнение е изцяло в главна функция main():

int main()

{

SetConsoleOutputCP(CP\_UTF8);

SetConsoleCP(CP\_UTF8);

Flight arr[ARRSIZE\_MAX];

int n;

n = 0;

**openFromFile**(arr, n);

sortArr(arr, n);

menu(arr, n);

**writeInFile(**arr, n);

}

Упътване за употреба

За всяко поле има създадена валидация. Функции, които отговарят за валидация на стринг и дата са:

void validateString(string& validated)

{

cin.clear();

cin.ignore((std::numeric\_limits<std::streamsize>::max)(), '\n');

getline(cin, validated);

while (cin.fail())

{

cout << "Невалиден вход, вкарайте текст";

cin.clear();

cin.ignore((std::numeric\_limits<std::streamsize>::max)(), '\n');

getline(cin, validated);

}

}

void validateDate(Date& date)

{

bool flag = false;

while (flag != true)

{

cin >> date.day >> date.month >> date.year;

if ((date.day > 0 && date.day < 31) && (date.month > 0 && date.month < 13) && date.year > 1900)

{

flag = true;

}

else

cout << "Въведете правилна дата" << endl;

}

}

Впишете съответната част от проекта

Очаквани входни данни

При искане за дата, тя трябва да бъде въведена в следния формат: ден, месец и година.

Очакван резултат от изпълнението на конкретния фрагмент от проекта

Ако бъде направен за превишаване на цифрата на съответното поле, ще бъде изкарано съобщение за грешка. Това ще продължи, докато потребителя не въведе правилния формат.

Впишете съответната част от проекта

Очаквани входни данни

При полето за номер на полет е позволено писането на цифри. По-принцип за всички останали стрингови полета, това е забранено.

Очакван резултат от изпълнението на конкретния фрагмент от проекта

Ако бъде направен опит за писане на число в останалите полета за стрингове, програмата ще върне съобщение за грешка.

Примерно действие на програмата

Условие A

Снимка на изгледа с примерни входни данни

A screen shot of a computer

Description automatically generated

Снимка на изгледа с примерни изходни данни

A screenshot of a computer

Description automatically generated

Условие B

Снимка на изгледа с примерни входни данни

A screen shot of a computer

Description automatically generated

Условие C

Снимка на изгледа с примерни изходни данни

A screenshot of a computer screen

Description automatically generated

Условие D

Снимка на изгледа с примерни входни данни

b)A black background with white text

Description automatically generated

Снимка на изгледа с примерни изходни данни

a)

b)A black background with white text

Description automatically generated

Условие E

Снимка на изгледа с примерни изходни данни

A screenshot of a computer screen

Description automatically generated

Условие F

Снимка на изгледа с примерни изходни данни

A screenshot of a computer

Description automatically generated

Допълнение първо

Снимка на изгледа с примерни входни данни

A screenshot of a computer

Description automatically generated

b) A screen shot of a black screen

Description automatically generated

Снимка на изгледа с примерни изходни данни

1. A computer screen with white text

   Description automatically generated
2. 

Допълнение второ

Снимка на изгледа с примерни входни данни

1. 
2. A black screen with white text

   Description automatically generated

Снимка на изгледа с примерни изходни данни

1. A black screen with white text

   Description automatically generated
2. A screenshot of a computer screen

   Description automatically generated

Допълнение I (Отказ на полет)

Снимка на изгледа с примерни входни данни

A black screen with white text

Description automatically generated

Снимка на изгледа с примерни изходни данни

