



ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра „СИТ“

СЕМЕСТРИАЛНА ДОМАШНА РАБОТА

по дисциплината „Базово програмиране“

на тема: „Самолетни полети“

Вариант 51

Изготвил: Селин Синан Кабил

Проверил: ас. инж. Велислав

Колесниченко

Специалност: СИТ

Група: 2^б

Факултетен номер: 22621625

2023

Съдържание

I. Задание на проекта	4
II. Анализ на решението	5
1. Структура за данните в програмата	5
2. Реализация на условие А	6
3. Реализация на условие В	9
4. Реализация на условие С	12
5. Реализация на условие D-a	15
6. Реализация на условие D-b	17
6. Реализация на условие Е	20
7. Реализация на условие F-a	22
7. Реализация на условие F-b	23
8. Реализация на условие G-a - допълнение първо	26
8. Реализация на условие G-b - допълнение първо	28
9. Реализация на условие H-a- допълнение второ	31
9. Реализация на условие H-b- допълнение второ	34
9. Реализация на условие I- допълнение второ	36
10. Реализация на допълнение трето	38
III. Упътване за употреба	41
1. Условие А	41
2. Условие В-a	41
3. Условие В-b	42
4. Условие С	42
5. Условие D-a	42
6. Условие D-b	43
7. Условие Е	43
8. Условие F-a	43
9. Условие F-b	43
10. Допълнение първо G-a	43
11. Допълнение първо G-b	44
12. Допълнение второ H -a	44
13. Допълнение второ H -b	44
14. Допълнение второ I	45
15. Допълнение трето	45
IV. Примерно действие на програмата	46
1. Условие А	46

2. Условие В-а	47
3. Условие В-б	47
4. Условие С	49
5. Условие D-a	50
6. Условие D-b	50
7. Условие Е	51
8. Условие F-a	51
9. Условие F-b	52
10. Допълнение първо G-a	52
11. Допълнение първо G-b	53
12. Допълнение второ Н -а	55
13. Допълнение второ Н -b	55
14. Допълнение второ I	56
15. Допълнение трето	56

I. Задание на проекта

Самолетни полети

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

A. Меню за избор на функциите в програмата.

Функции от програмата са:

B. Добавяне на полети

a. Добавяне по един полет.

b. Добавяне на списък с полети. Въвежда се цяло число *n* и след него *n* на брой полети.

C. Извеждане на екрана

a. Извеждане на всички полети в оформен вид

D. Търсене и извеждане на екрана

a. Извеждане на всички полети с най-ниска цена

b. Извеждане на полети на даден пилот

E. Подреждане на основния масив с полети

a. Подреждане на полетите в низходящ ред по азбучен ред на дестинацията.

F. Управление на файл

a. Извеждане на масива от книги във файл (двоичен)

b. Въвеждане на масива от книги от файл (двоичен)

Допълнение първо (+ базова задача)

G. Създайте подменю, в което се влиза от основното, с нови функции за:

a. Извеждане на полетите във възходящ ред на цената на полета и класата на полета, без да се променя основния масив

b. Търсене и извеждане на полети по въведена дата и определена дестинация.

Допълнение второ (+ базова задача)

H. Изчисляване на текуща цена на полета

a. 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);

b. При желание е възможна замяна на билет от втора класа с такъв от първа класа. Въвежда се номер на полета и името на пътника.

I. Отказ от полет

a. 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)

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

II. Анализ на решението

1. Структура за данните в програмата

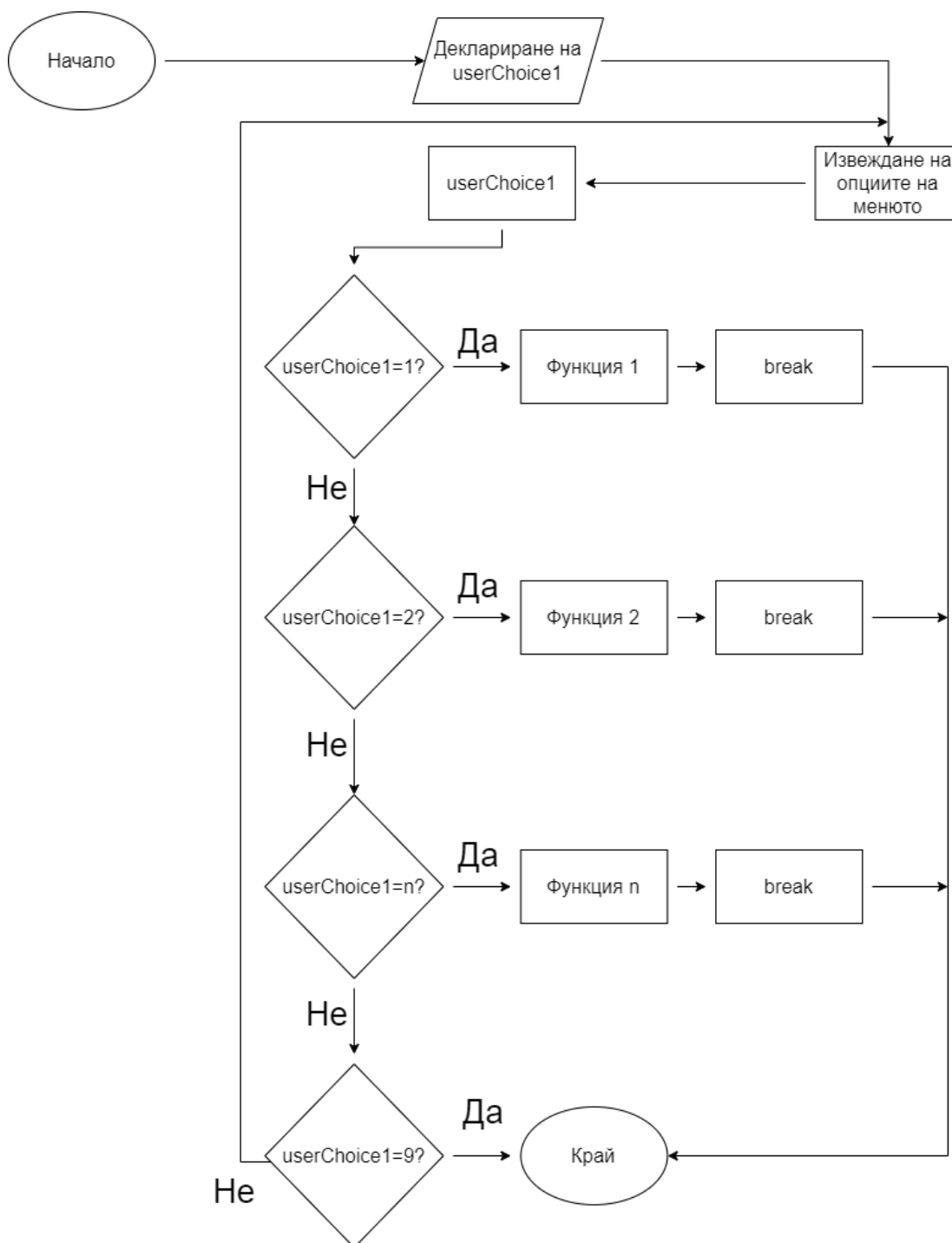
Структура	Обяснение	Примерни стойности
<pre>struct flights { char pilot[100]; char destination[50]; char passenger[100]; double price=0; bool firstClass; int date=0; int number=0; };</pre>	<ul style="list-style-type: none">- pilot-съхранява името и фамилията на пилота- destination-съхранява дестинацията на полета- passenger-съхранява името и фамилията на пътника- price-съхранява цената на полета- firstClass-съхранява информация за класата на пътника - 1(Първа класа) , 0(Втора класа)- date-съхранява датата на полета DDMMYYYY- number-съхранява номера на полета	Иван Георгиев Лондон Гергана Йорданова 299.99 1 12122022 2

2. Реализация на условие A

2.1. Анализ на алгоритъма, който трябва да се реализира

1. Разпечатва се списък с опции, от които потребителят да избира.
2. Потребителят въвежда своя избор.
3. Обработка се изборът на потребителя.
4. В зависимост от избора на потребителя, се изпълнява съответното действие.
5. Алгоритъмът се повтаря докато потребителят избере да излезе от менюто.

2.2. Блок схема на алгоритъма



2.3. Функция, с която е реализиран алгоритъма

```
unsigned short userChoice2 = 0, userChoice = 0;
do {
    cout << "\nИзберете опция от менюто\n";
    cout << "1.Добавяне на един полет\n";
    cout << "2.Добавяне на n полета\n";
    cout << "3.Извеждане на полетите\n";
    cout << "4.Полети с най-ниска цена\n";
    cout << "5.Извеждане на полети на даден пилот\n";
    cout << "6.Извеждане във файл\n";
    cout << "7.Подреждане на полетите в низходящ ред по азбучен ред на
дестинацията\n";
    cout << "8.Подменю\n";
    cout << "9.Изход\n";
    cout << "Избор:";
    cin >> userChoice;
    switch (userChoice) {
    case 1:
        if (n + m < 99)
            enterFlight(flight, 1, m);
        break;
    case 2:
        cout << "\nВъведете брой полети : ";
        cin >> n;
        if (n + m < 99)
            enterFlight(flight, n, m);
        else
            cout << "Невалиден брой операции" << endl;
        break;
    case 3:
        printFlight(flight, size);
        break;
    case 4:
        leastExpensive(flight, size);
        break;
    case 5:
        printPilot(flight, size);
        break;
    case 6:
```

```

        file_out(flight, size, n, m);
        break;
    case 7:
        destinationAlph(flight, size);
        break;
    case 8:
        do {
            cout << "\nИзберете опция от менюто ";
            cout << "\n1.Сортиране във възходящ ред на цената на полета и
класата на полета";
            cout << "\n2.Търсене по въведена дата и определена дестинация";
            cout << "\n3.Цена на билет: ";
            cout << "\n4.Замяна на билет от втора класа";
            cout << "\n5.Отказ от полет";

            cout << "\n6.Назад ";
            cout << "\nИзбор:";
            cin >> userChoice2;
            switch (userChoice2) {
            case 1:
                classPriceAsc(flight, size);
                break;
            case 2:
                searchByDateDestin(flight, size);
                break;
            case 3:
                currentPrice(flight, size);
                break;
            case 4:
                switchClass(flight, size);
                break;
            case 5:
                cancelFlight(flight, size);
                break;
            case 6:
                break;
            default:
                cout << "Въведете валиден номер на операция:" << endl;
                break;
            }
        }

```



```

        } while (userChoice2 != 6);
        break;
    case 9:
        file_out(flight,size, n, m);
        break;
    default:
        cout << "Въведете валиден номер на операция : "<<endl;
        break;
    }
} while (userChoice != 9);
return 0;
}

```

Дефинира се променлива , която ще се използва за съхраняване номера на опцията, избрана от потребителя. unsigned short userChoice = 0. "Unsigned short" е тип integer, който може да съхранява само неотрицателни стойности. В този случай е удачно да се използва, защото опциите в менюто са малки положителни числа и променливата заема по-малко памет. Тази променлива ще се използва за съхраняване на избора на потребителя от менюто.

С do-while цикъл се създава меню. Цикълът продължава да работи, докато потребителят не е избрал опцията за излизане от програмата. Вътре в цикъла се извеждат опциите на менюто, въвежда се номера на желаната опция. Със оператора switch се управлява избора на потребителя, като всеки case на оператора съответства на опция от менюто.

2.3.1. Входни данни на функцията

Главната функция не приема параметри.

2.3.2. Изходни данни на функцията или данни, които се извеждат

Опциите на менюто се извеждат и потребителят въвежда своя избор. Изпълнява се кодът за съответното действие. Ако потребителят избере 9, програмата ще излезе от цикъла и ще затвори програмата с return 0.

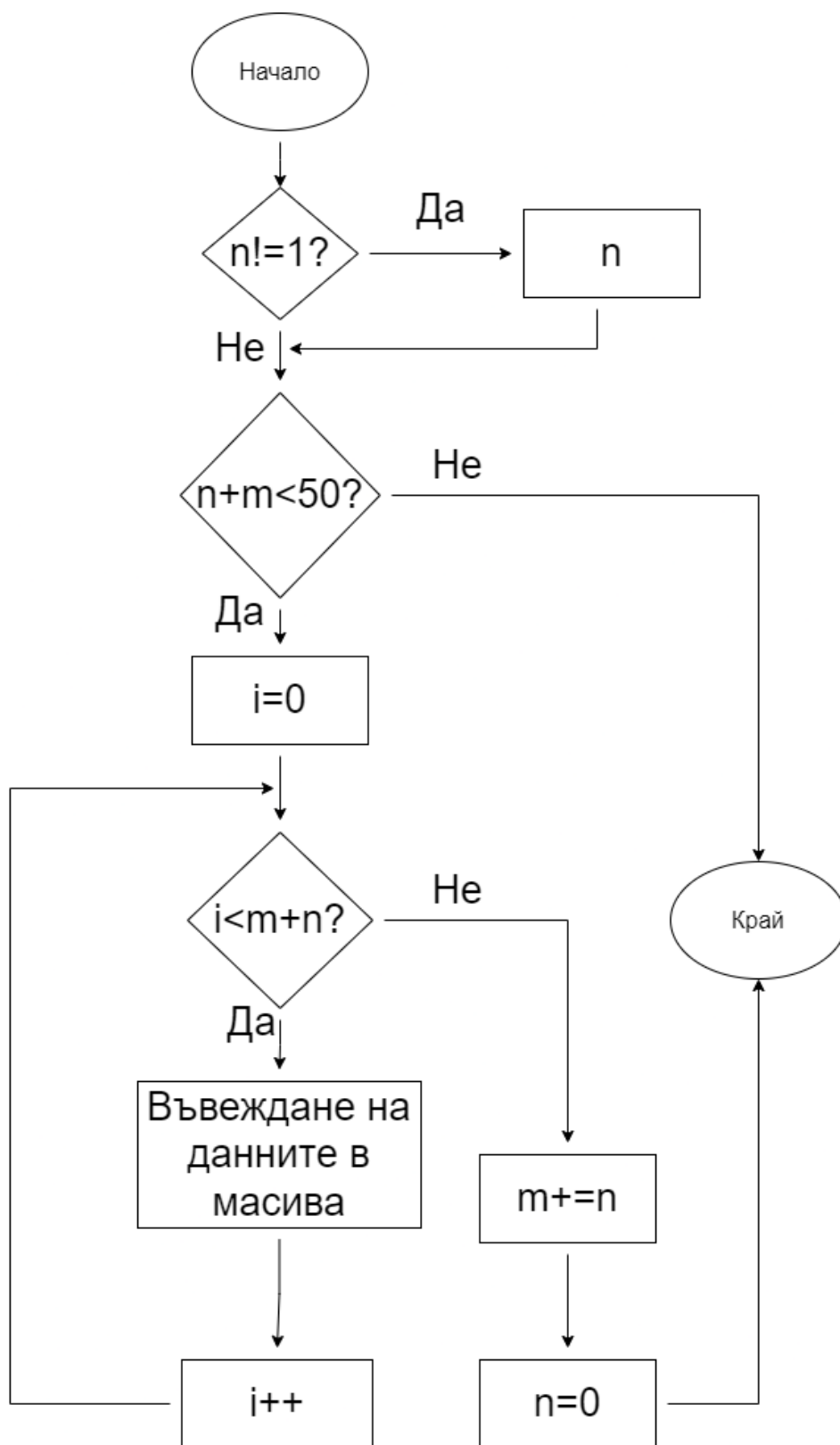
3. Реализация на условие В

3.1. Анализ на алгоритъма, който трябва да се реализира

1. С цикъл for се обхождат всички елементи на масива.
2. Потребителя въвежда подробностите за елемента.
3. Проверява се дали въведения брой елементи не надвишава размера на масива.
4. Съхранява се информация за структурата в масива.

5. Увеличава се променливата на брояча.
6. Повторят се стъпки 2-5, докато потребителят приключи с въвеждането на структури или масивът се запълни.

3.2. Блок схема на алгоритъма



3.3. Функция, с която е реализиран алгоритъма

```
void enterFlight(flights flight[], int n, int& m);
void enterFlight(flights flight[], int n, int& m) {
    if(n!=1){
        cout << "\nВъведете брой полети : ";
        cin >> n;}
    if (n + m < 50){
        for (int i = m; i < m + n; i++) {
            cout << "\nВъведете полет:\n\nНомер: ";
            cin >> flight[i].number;
            cout << "Дата: ";
            cin >> flight[i].date;
            cout << "Цена: ";
            cin >> flight[i].price;
            cout << "Пилот: ";
            cin.ignore();
            cin.getline(flight[i].pilot,100);
            cout << "Първа(1)/Втора(0) класа: ";
            cin >> flight[i].firstClass;
            cin.ignore();
            cout << "Име на пътник: ";
            cin.getline(flight[i].passenger, 100);
            cout << "Дестинация: ";
            cin.getline(flight[i].destination, 50);

        }m += n;
        n = 0;}
    else
        cout << "Невалиден брой операции" << endl;
}
```

Потребителят въвежда броя на полетите, които желае да въведе, ако ще се въвеждат повече от 1 полет. След това функцията проверява дали общият брой полети ($n + m$) е по-малък от 50, което е максималният брой полети, които могат да бъдат съхранени в масива. Ако общият брой полети е по-малък от 50, функцията влиза в цикъл за събиране на информация за всеки полет. Цикълът започва от стойността на "m" и повтаря "n" пъти(или до $(m+n)$ -тия индекс), като събира информация за всеки полет и я съхранява в масива. В края на цикъла стойността

на "m" се актуализира, за да отрази новия въведен брой полети. Ако общият брой полети е по-голям от 50, функцията показва съобщение за грешка.

За всяка итерация алгоритъмът въвежда стойността на номера, датата, цената, името на пилота, класата, пътника и дестинацията на полета.

3.3.1. Входни данни на функцията

Функцията приема три параметъра: масив от полети от тип flights, цяло число "n", представляващо броя на полетите, които трябва да бъдат въведени, и адреса на цяло число "m", представляващо броя на вече въведените полети. "m" се подава по адрес, за да се запази общият брой въведени данни за следващото въвеждане и при проверката ($m+n < 50$) да не се превиши общият размер на масива.

3.3.2. Изходни данни на функцията или данни, които се извеждат

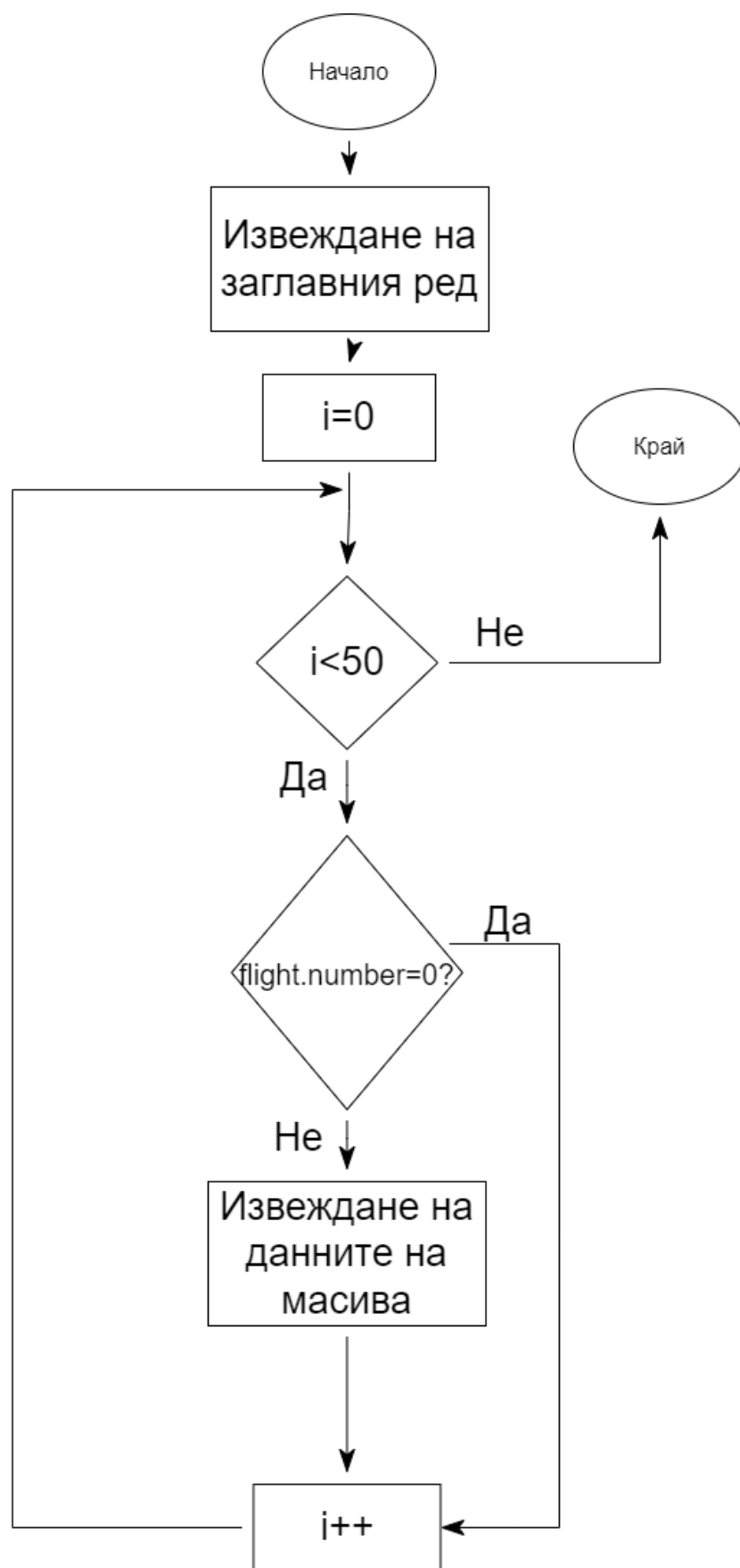
Функцията е от тип void(не връща резултат), тъй като масивът е адресът на първият елемент и всички данни се запазват.

4. Реализация на условие C

4.1. Анализ на алгоритъма, който трябва да се реализира

1. Извежда се заглавният ред на елементите на масива.
2. Декларира се променлива брояч, за да се следи броя на елементите в масива.
3. Извежда се информация за всеки елемент на масива, въведен от клавиатурата
4. Увеличава се променливата на брояча.
5. Повторят се стъпки 2-5, докато масивът приключи.

4.2. Блок схема на алгоритъма



4.3. Функция, с която е реализиран алгоритъма

```
void printFlight(flights flight[], int size);
void printFlight(flights flight[], int size) {
    cout << setfill(' ') << setw(10) << left << "Номер" << setw(15) << "Дата" <<
    setw(12) << left << "Цена" << setw(25) << left << "Пилот";
    cout << setfill(' ') << setw(15) << left << "Дестинация" << setw(18) << left <<
    "Име на пътник" << setw(15) << left << "Първа/Втора класа " << endl;

    for (int i = 0; i < size; i++) {
        if (flight[i].number == 0)
            continue;
        cout << setfill(' ') << setw(10) << left << flight[i].number;
        cout << setw(15) << left << flight[i].date;
        cout << setw(12) << left << flight[i].price;
        cout << setfill(' ') << setw(25) << left
            << flight[i].pilot << setw(15) << left << flight[i].destination << setw(25) <<
            left << flight[i].passenger << setw(15) << left << flight[i].firstClass << endl;
    }
}
```

Функцията първо отпечатва заглавния ред на елементите, като използва функцията `cout` и манипулаторите `setfill`, `setw` и `left`. След това преминава през елементите на масива и за всеки елемент отпечатва подробностите за полета, използвайки функцията `cout` и същите манипулатори. Ако номерът на полета на текущия елемент е нула, цикълът пропуска елемента и продължава към следващия.

Подробностите за всеки полет се отпечатват в табличен формат, като всяко поле заема определен брой места в изхода. Манипулаторите `setfill` и `setw` се използват за контролиране на изхода, като `setfill` определя знака, който да се използва като запълване, когато ширината на полето е по-голяма от данните, които се отпечатват, а `setw` определя ширината на полето. Манипулаторът `left` подравнява изхода отляво на полето.

4.3.1. Входни данни на функцията

Функцията приема масив от полети от тип `flights` и размера на масива като параметри. Размера на масива показва колко пъти ще се повтори цикълът.

4.3.2. Изходни данни на функцията или данни, които се извеждат

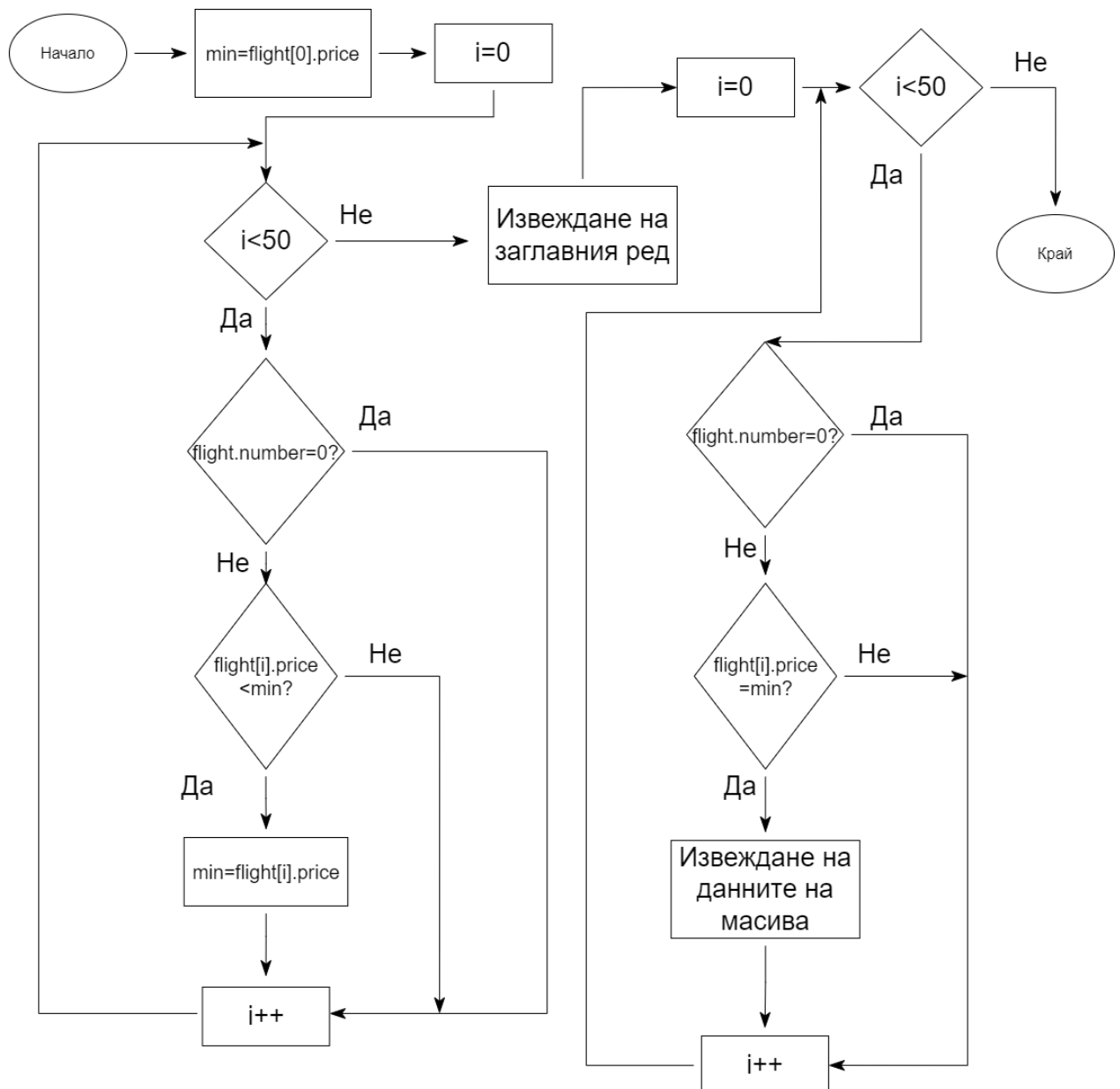
Извежда всички елементи на масива в табличен вид.

5. Реализация на условие D-a

5.1. Анализ на алгоритъма, който трябва да се реализира

1. Инициализира се променлива min с първия елемент в масива.
2. Останалите елементи в масива се обхождат.
3. Всеки елемент се сравнява с текущия минимален елемент. Ако е по-малък, се актуализира минималния елемент, за да бъде текущият елемент.
4. След итерация през целия масив, минималният елемент ще бъде съхранен в променливата min.
5. Масивът се обхожда за втори път и се проверява дали минималният елемент се повтаря.
6. Ако е така масивът се извежда.

5.2. Блок схема на алгоритъма



5.3. Функция с която е реализиран алгоритъма

```
void leastExpensive(flights flight[], int size) ;
void leastExpensive(flights flight[], int size) {
    double min = flight[0].price;
    for (int i = 0; i < size; i++){
        if (flight[i].number == 0)
            continue;
        else if (flight[i].price < min)
            min = flight[i].price;
    }

    cout << setfill(' ') << setw(10) << left << "Номер" << setw(15) << "Дата" <<
    setw(12) << left << "Цена" << setw(25) << left << "Пилот";
    cout << setfill(' ') << setw(15) << left << "Дестинация" << setw(18) << left <<
    "Име на пътник" << setw(15) << left << "Първа/Втора класа " << endl;

    for (int i = 0; i < size; i++) {
        if (flight[i].price == min) {
            cout << setfill(' ') << setw(10) << left << flight[i].number << setw(15) << left
            << flight[i].date << setw(12) << left << flight[i].price;
            cout << setfill(' ') << setw(25) << left << flight[i].pilot << setw(15) << left <<
            flight[i].destination << setw(25) << left << flight[i].passenger << setw(15) << left <<
            flight[i].firstClass << endl;
        }

        else if (flight[i].number == 0)
            continue;
    }
}
```

Това е функция, която отпечатва подробностите за най-евтиния полет в даден масив от структури на полети.

Функцията първо инициализира променлива min с цената на първия елемент от масива. След това преминава през елементите на масива и сравнява цената на всеки елемент със стойността на min. Ако цената на текущия елемент е по-малка от min, min се актуализира до новата цена.

След това функцията отпечатва заглавния ред на елементите, като използва функцията cout и манипулаторите setfill, setw и left. Отново преминава през елементите на масива и за всеки елемент проверява дали номерът на полета на

текущия елемент е нула. Ако е така, цикълът пропуска елемента и продължава към следващия.

След това проверява дали цената на елемента е равна на стойността на min. Ако е така, подробностите за полета се отпечатват с помощта на функцията cout и същите манипулатори както преди.

5.3.1. Входни данни на функцията

Функцията приема масив от полети от тип flights и размера на масива като параметри. Размера на масива показва колко пъти ще се повтори цикълът.

5.3.2. Изходни данни на функцията или данни, които се извеждат

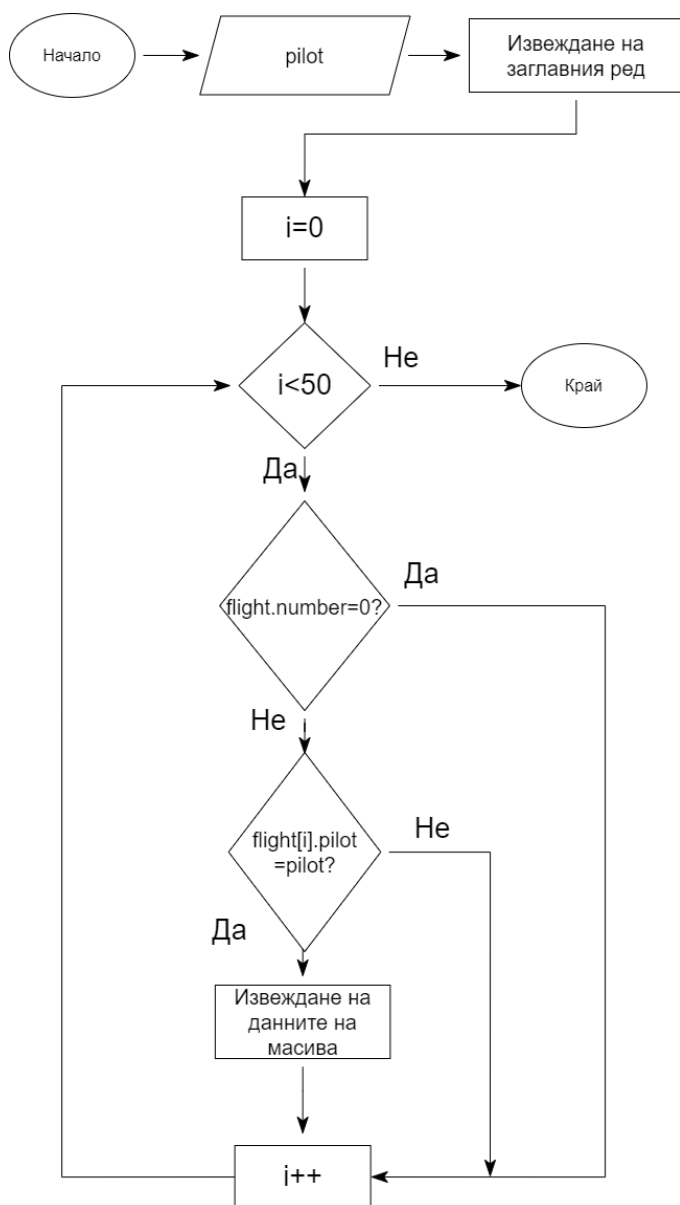
Извежда всички полети с най-ниска цена в табличен вид.

6. Реализация на условие D-b

6.1. Анализ на алгоритъма, който трябва да се реализира

1. Извежда се заглавният ред на елементите на масива.
2. Декларира се променлива за име пилот и се въвежда от клавиатурата.
3. С for цикъл се проверява дали всеки елемента съвпада с името на пилота, въведено от клавиатурата.
4. Увеличава се променливата на брояча.
5. Повторят се стъпки 2-5, докато масивът приключи или ако даден елемент е 0(не е въведен).

6.2. Блок схема на алгоритъта



6.3. Функция с която е реализиран алгоритъта

```
void printPilot(flights flight[], int size);
```

```
void printPilot(flights flight[], int size) {  
    char pilot[100];
```

```
    cout << "Въведете име на пилот : ";
```

```
    cin.ignore();
```

```
    cin.getline(pilot, 100);
```

```
    cout << setfill(' ') << setw(10) << left << "Номер" << setw(15) << "Дата" <<  
    setw(12) << left << "Цена" << setw(25) << left << "Пилот";
```

```
    cout << setfill(' ') << setw(15) << left << "Дестинация" << setw(18) << left <<  
    "Име на пътник" << setw(15) << left << "Първа/Втора класа " << endl;
```

```

for (int i = 0; i < size; i++) {
    if (flight[i].number == 0)
        break;
    else if (strcmp(flight[i].pilot, pilot)==0) {
        cout << setfill(' ') << setw(10) << left << flight[i].number;
        cout << setw(15) << left << flight[i].date;
        cout << setw(12) << left << flight[i].price;
        cout << setfill(' ') << setw(25) << left << flight[i].pilot << setw(15) << left <<
flight[i].destination << setw(25) << left << flight[i].passenger << setw(15) << left <<
flight[i].firstClass << endl;

    }

}
}
}

```

Потребителят въвежда името на пилот и след това функцията търси в масива от полети за полети с пилот, чието име съвпада с въведеното от потребителя. Ако се намери съвпадение, функцията отпечатва подробностите за полета (номер, дата, цена, пилот, дестинация, пътник и класа). За всеки елемент проверява дали номерът на полета на текущия елемент е нула. Ако е така, цикълът пропуска елемента и продължава към следващия.

Функцията използва функцията `strcmp`, за да сравни името на пилота в масива с името, въведено от потребителя, тъй като е `c-string`. Функцията също така проверява номера на всеки полет, за да види дали е равно на 0, което показва края на масива. Входни данни на функцията

Функцията приема масив от полети от тип `flights` и размера на масива като параметри. Размера на масива показва колко пъти ще се повтори цикълът.

6.3.1. Входни данни на функцията

Функцията приема масив от полети от тип `flights` и размера на масива като параметри. Размера на масива показва колко пъти ще се повтори цикълът.

6.3.2. Изходни данни на функцията или данни, които се извеждат

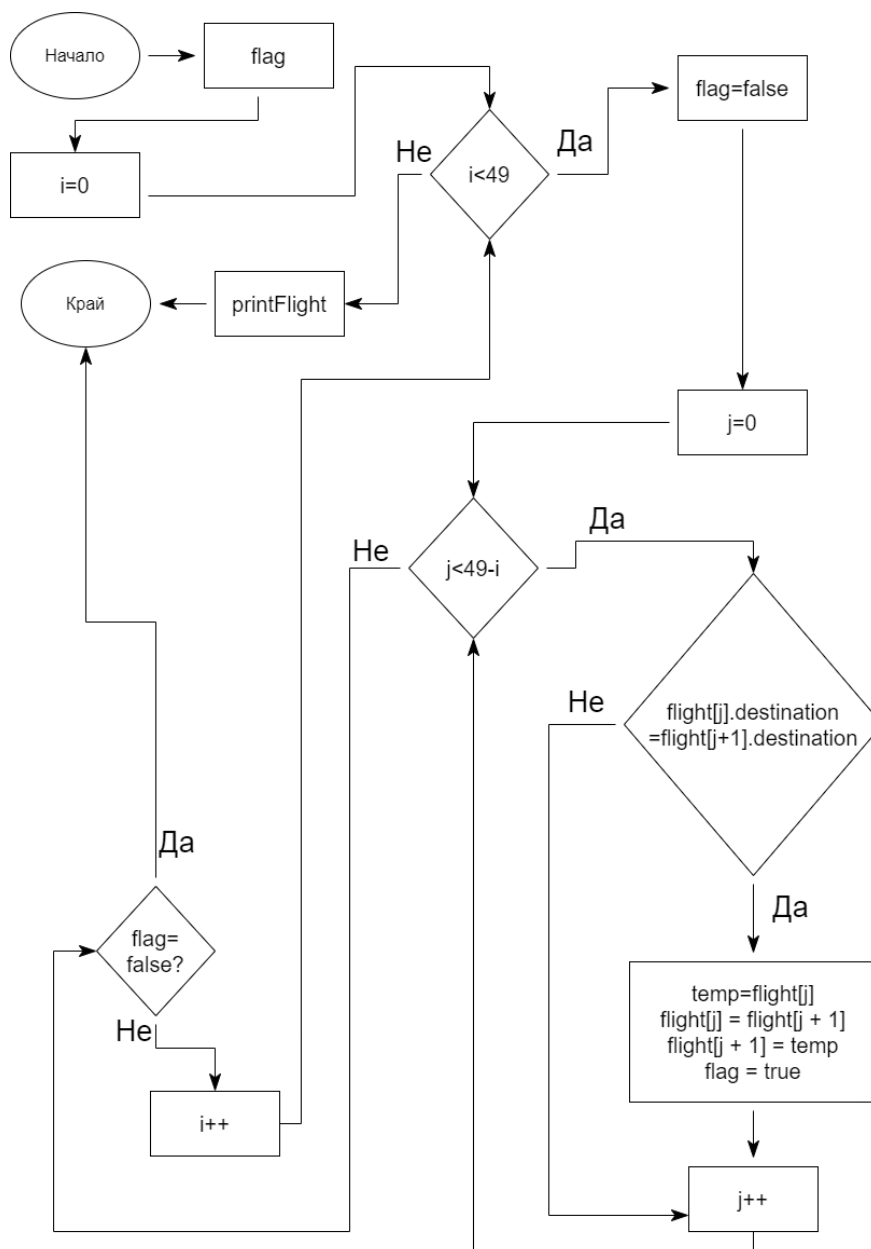
Извежда всички полети с въведения пилот в табличен вид.

7. Реализация на условие Е

7.1. Анализ на алгоритъма, който трябва да се реализира

1. Обхожда се масива с помощта на 2 for цикъла (Метод на мехурчето).
2. Сравняват се дестинациите на съседни полети. Ако дестинацията на първия полет е преди дестинацията на втория полет по азбучен ред, се разменят полетите.
3. Повтаря се стъпка 2, докато масивът бъде сортиран.
4. Извиква се функцията printFlight, за да се отпечатаат подробностите за полетите.

7.2. Блок схема на алгоритъма



7.3. Функция с която е реализиран алгоритъма

```
void destinationAlph(flights flight[], int size);  
void destinationAlph(flights flight[], int size) {  
    bool flag;  
  
    for (int i = 0; i < size - 1; i++) {  
        flag = false;  
        for (int j = 0; j < size - 1 - i; j++) {  
            if (strcmp(flight[j].destination, flight[j + 1].destination)==-1) {  
                flights temp = flight[j];  
                flight[j] = flight[j + 1];  
                flight[j + 1] = temp;  
                flag = true;  
            }  
        }  
        if(flag==false)  
            break;  
    }  
    printFlight(flight, size);  
}
```

Функцията използва алгоритъма за bubble sort, за да сортира масива. Започва с итерация през масива с помощта на for цикъл и за всяка итерация сравнява дестинациите на съседни полети. Ако дестинацията на първия полет е преди дестинацията на втория полет по азбучен ред, полетите се разменят. Функцията продължава да итерира и разменя, докато масивът бъде сортиран. Променливата flag е true ако има поне един swap във вътрешният цикъл, ако flag остане false цикълът се прекъсва, за да се лимитират ненужни итерации.

След като сортирането приключи, функцията извиква функцията printFlight, за да отпечата подробностите за полетите.

7.3.1. Входни данни на функцията

Функцията приема масив от полети от тип flights и размера на масива като параметри. Размера на масива показва колко пъти ще се повтори цикълът.

7.3.2. Изходни данни на функцията или данни, които се извеждат

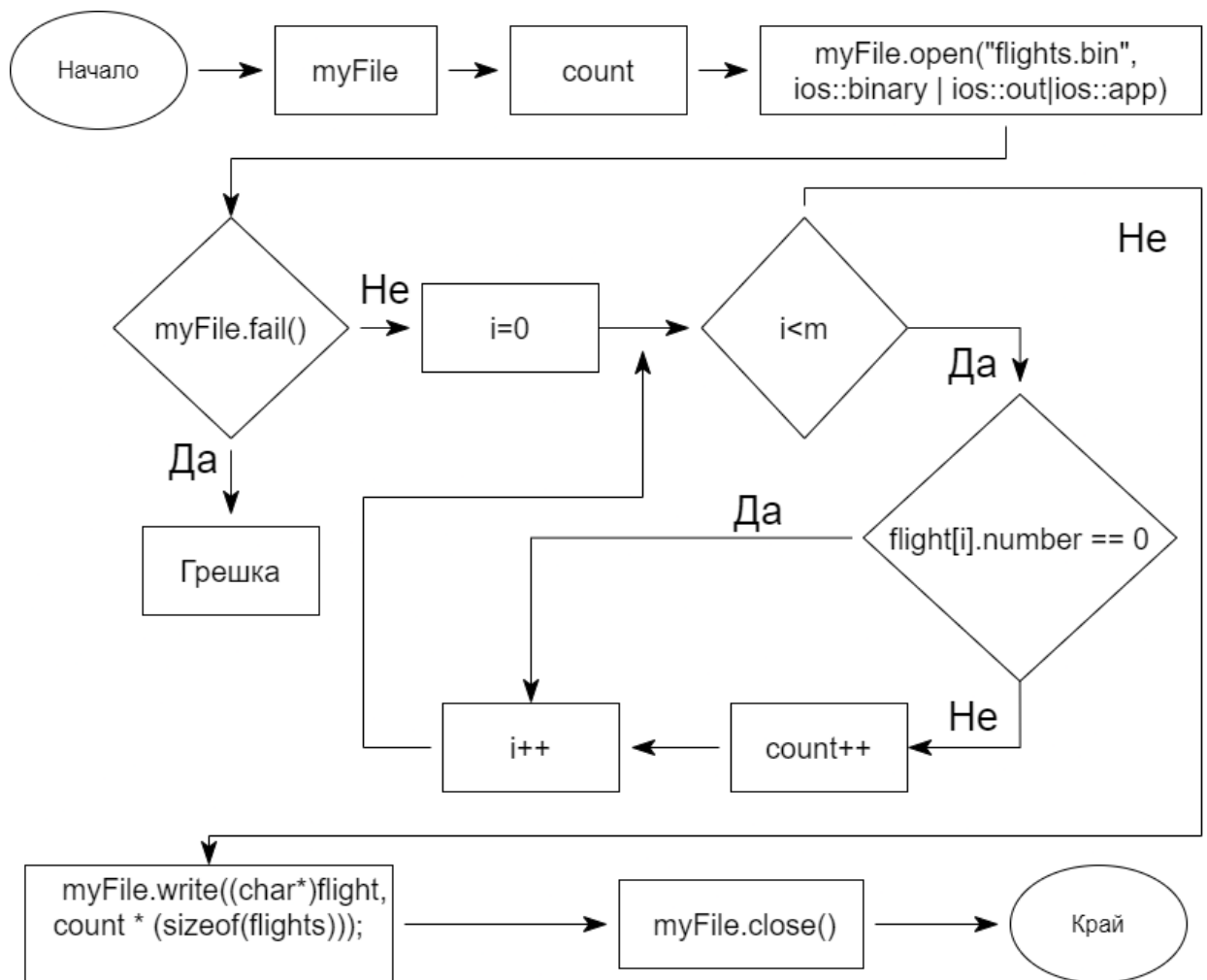
Извежда всички полети в низходящ ред по азбучен ред на дестинацията в табличен вид.

8. Реализация на условие F-a

8.1. Анализ на алгоритъма, който трябва да се реализира

1. Отваря се двоичният файл в двоичен режим, и статус append.
2. С for цикъл, започващ от 0 до m(последният записан елемент), се преброяват всички въведени записи и се съхраняват в променлива.
3. Данните се записват във файла
4. Затваря се файлът.

8.2. Блок схема на алгоритъма



8.3. Функция с която е реализиран алгоритъма

```
void file_out(flights flight[], int size, int& m);
void file_out(flights flight[], int size, int& m) {
    fstream myFile;
    int count = 0;
    myFile.open("flights.bin", ios::binary | ios::out|ios::app);
    if (myFile.fail())
        cout << "Грешка" << endl;
```

```

    for (int i = 0; i < m; i++) {
        if (flight[i].number == 0)
            continue;
        else count++;
    }
    myFile.write((char*)flight, count * (sizeof(flights)));
    myFile.close();
}

```

Функцията започва с отваряне на двоичния файл в двоичен режим и статус append с помощта на класа fstream. Ако файлът не успее да се отвори, се отпечатва съобщение за грешка.

След това използва цикъл, за да преброи броя на полетите в масива, които нямат номер 0. Това показва броя на полетите за запис във файла.

След това функцията използва функцията за запис(write), за да запише полетите във файла като двоични данни. Функцията write записва блок от данни във файла. Първият аргумент е указател към данните за запис, а вторият аргумент е броят байтове за запис. В този случай функцията записва брой полети, където count е броят полети въведени в масива, а размерът на всеки полет е sizeof(flights) байтове.

Накрая функцията затваря файла.

8.3.1. Входни данни на функцията

Функцията приема масив от полети от тип flights и размера на масива като параметри и адреса на цяло число "m", представляващо броя на вече въведените полети. "m" се подава по адрес, за да се запази общият брой въведени данни за следващото въвеждане. Размера на масива показва колко пъти ще се повтори цикълът.

8.3.2. Изходни данни на функцията или данни, които се извеждат

Тази функция не връща резултат и не извежда данни. Тя записва подробностите за полетите от масива в двоичния файл. Полетите ще бъдат добавени в края на файла, ако той вече съществува, или ще бъде създаден нов файл, ако не съществува.

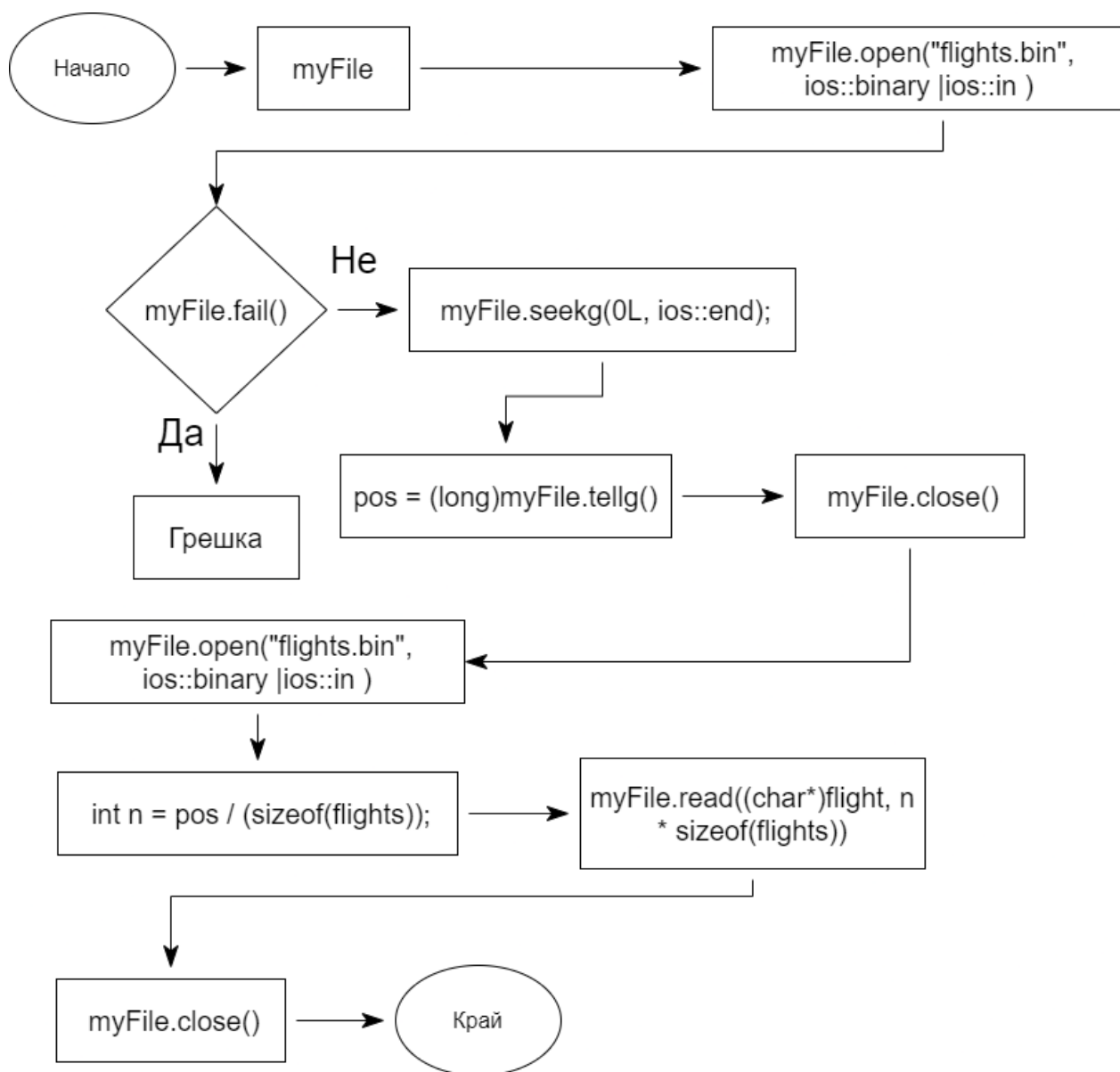
9. Реализация на условие F-b

9.1. Анализ на алгоритъма, който трябва да се реализира

1. Отваря се двоичният файл в двоичен режим, и статус append.
2. С for цикъл, започващ от 0 до m(последният записан елемент), се преброяват всички въведени записи и се съхраняват в променлива.

3. Данните се записват във файла
4. Затваря се изходния файл.

9.2. Блок схема на алгоритъма



9.3. Функция с която е реализиран алгоритъма

```

void file_in(flights flight[])
void file_in(flights flight[]) {
    fstream myFile;
    myFile.open("flights.bin", ios::binary | ios::in );
    if (myFile.fail())
        cout << "Грешка" << endl;
    myFile.seekg(0L, ios::end);
    long pos = (long)myFile.tellg();
    myFile.close();
}
  
```



```

myFile.open("flights.bin", ios::binary | ios::in);
int n = pos / (sizeof(flights));
myFile.read((char*)flight, n * sizeof(flights));
myFile.close();
}

```

Функцията първо отваря файла в двоичен режим "in", използвайки fstream променливата myFile. Ако файлът не успее да се отвори, се отпечатва съобщение за грешка.

След това функцията използва функцията seekg(), за да премести указателя на файла до края на файла, и функцията tellg(), за да получи текущата позиция на указателя на файла (т.е. размера на файла в байтове) . Затваря файла.

Функцията отваря отново файла в двоичен режим in и изчислява броя на структурите за полети, които могат да бъдат съхранени във файла, като раздели размера на файла (в байтове) на размера на една структура за полети (също в байтове).

След това , функцията използва функцията read(), за да прочете съдържанието на файла в масива на полета, използвайки typeid към char*, за да укаже, че данните трябва да се четат като текстови данни. След това затваря файла.

9.3.1. Входни данни на функцията

Функцията приема масив от полети от тип flights като параметър

9.3.2. Изходни данни на функцията или данни, които се извеждат

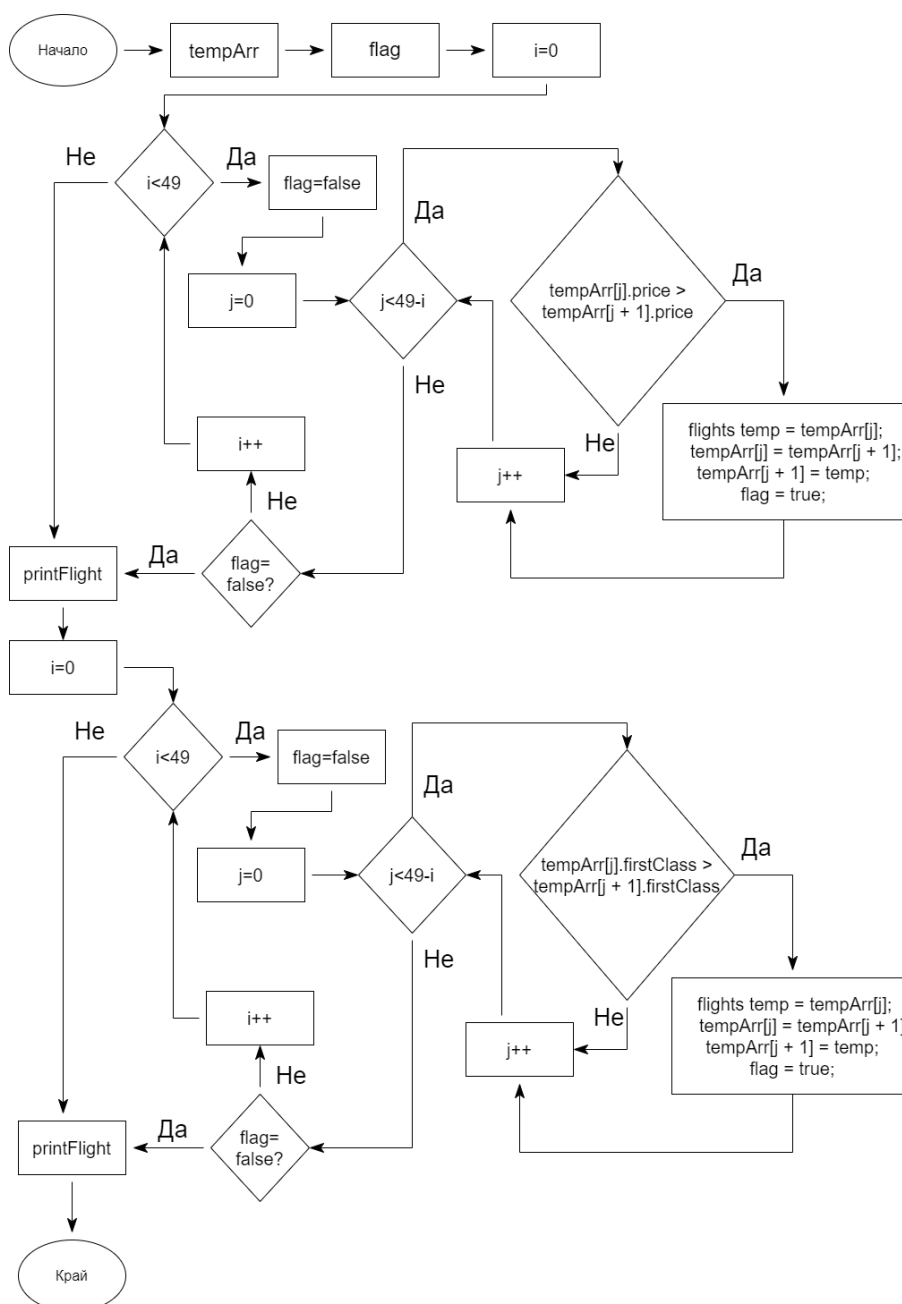
Тази функция не връща резултат и не извежда данни. Тя чете двоичен файл, наречен "flights.bin", и съхранява съдържанието му в масив от структури за полети.

10. Реализация на условие G-a - допълнение първо

10.1. Анализ на алгоритъма, който трябва да се реализира

1. Създава се копие на основния масив.
2. Обхожда се масива с for цикъл.
3. Сравняват се всяка двойка съседни елементи.
4. Ако първият елемент е по-голям от втория елемент, се разменят.
5. Този процес се повтаря , докато масивът не бъде сортиран.
6. Стъпки 2-5 се повтарят за сортирането по втория признак.
7. Извежда се резултатът от сортирането.

10.2. Блок схема на алгоритъма



10.3. Функция с която е реализиран алгоритъма

```
void classPriceAsc(const flights flight[], int size);
void classPriceAsc(const flights flight[], int size) {
    flights tempArr[50];
    bool flag;

    for (int i = 0; i < size; i++) {
        tempArr[i]=flight[i];
    }

    cout << "\nПолети във възходящ ред на цената:\n";

    for (int i = 0; i < size - 1; i++) {
        flag = false;
        for (int j = 0; j < size - 1 - i; j++) {
            if (tempArr[j].price > tempArr[j + 1].price) {
                flights temp = tempArr[j];
                tempArr[j] = tempArr[j + 1];
                tempArr[j + 1] = temp;
                flag = true;
            }
        }
        if (flag == false) break;
    }

    printFlight(tempArr, size);

    cout << "\n Полети във възходящ ред на класа на полета \n";

    for (int i = 0; i < size - 1; i++) {
        flag = false;
        for (int j = 0; j < size - 1 - i; j++) {
            if (tempArr[j].firstClass > tempArr[j + 1].firstClass) {
                flights temp = tempArr[j];
                tempArr[j] = tempArr[j + 1];
                tempArr[j + 1]= temp;
                flag = true;
            }
        }
    }
}
```

```

        if (flag == false)
            break;
    }
    printFlight(tempArr, size);
}

```

Функцията първо създава копие на входния масив, наречен tempArr, и инициализира булев флаг, наречен flag. След това функцията сортира два пъти масива tempArr използвайки метода на мехурчето. Първата сортировка сортира полетите във възходящ ред по цена. Преминава през масива и сравнява цената на всяка двойка съседни полети. Ако цената на първия полет е по-висока от цената на втория полет, полетите се разменят. Цикълът продължава, докато масивът не бъде сортиран във възходящ ред по цена. Втората сортировка сортира полетите във възходящ ред по първа класа. Работи по същия начин като се сравнява първата класа на всяка двойка съседни полети вместо цената. След всяко сортиране функцията отпечатва сортирания масив от полети с помощта на функцията printFlight.

10.3.1. Входни данни на функцията

Функцията приема масив от полети от тип flights и размера на масива като параметри. Размера на масива показва колко пъти ще се повтори цикълът.

10.3.2. Изходни данни на функцията или данни, които се извеждат

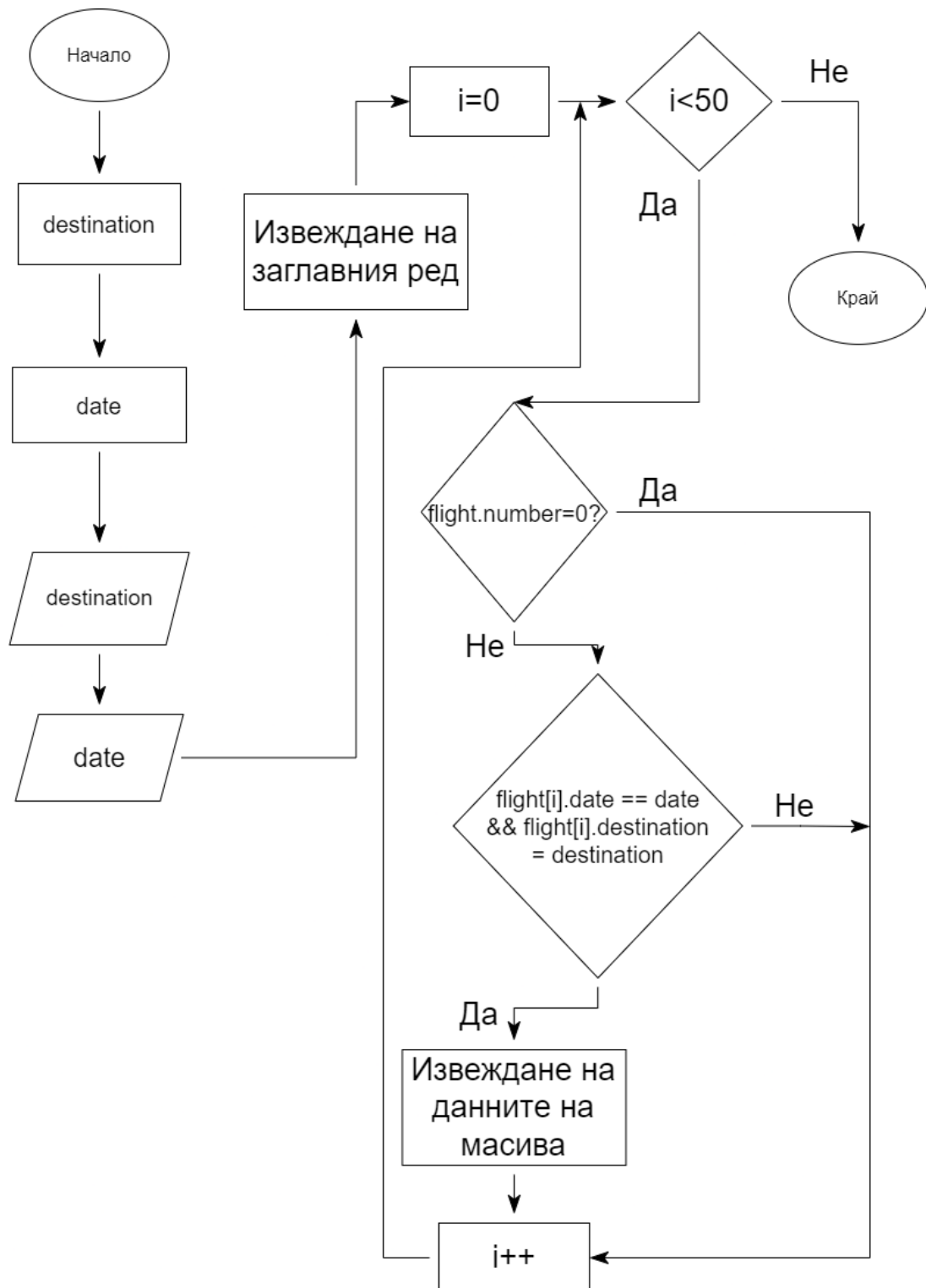
Извежда всички елементи на масива в табличен вид сортирани по съответните признаци.

11. Реализация на условие G-b - допълнение първо

11.1. Анализ на алгоритъма, който трябва да се реализира

1. Въвежда се дата и дестинация;
2. Преминава се през всички елементи на масива с for цикъл
3. За всеки елемент се проверява дали е равен на датата и дестинацията, въведени от клавиатурата.
4. Ако е така, се извеждат елементите на масива
5. Увеличава се брояча на цикъла проверката се повтаря, докато се достигне края на масива.
6. Ако края на масива е достигнат без намиране на целевата стойност, функцията приключва.

11.2. Блок схема на алгоритъма



11.3. Функция с която е реализиран алгоритъма

```

void searchByDateDestin(flights flight[], int size) ;
void searchByDateDestin(flights flight[], int size) {
    char destination[50];
    int date;

```

```

cout << "\nВъведете дата:";
cin >> date;
cin.ignore();
cout << "Въведете дестинация:";
cin.getline(destination, 100);
    cout << setfill(' ') << setw(10) << left << "Номер" << setw(15) << "Дата" <<
setw(12) << left << "Цена" << setw(25) << left << "Пилот";
    cout << setfill(' ') << setw(15) << left << "Дестинация" << setw(18) << left <<
"Име на пътник" << setw(15) << left << "Първа/Втора класа " << endl;

```

```

for (int i = 0; i < size; i++) {
    if (flight[i].number == 0)
        continue;
    else if ((flight[i].date == date) && (strcmp(flight[i].destination, destination)==0))
    {
        cout << setfill(' ') << setw(10) << left << flight[i].number << setw(15) << left
<< flight[i].date << setw(12) << left << flight[i].price;
        cout << setfill(' ') << setw(25) << left << flight[i].pilot << setw(15) << left <<
flight[i].destination << setw(25) << left << flight[i].passenger << setw(15) << left <<
flight[i].firstClass << endl;
    }
}
}

```

Потребителят да въвежда дата и дестинация и след това функцията преминава през масива от полети, за да намери всички полети, които съответстват както на датата, така и на дестинацията. Ако се намери съвпадение, функцията отпечата подробностите за полета (номер, дата, цена, пилот, дестинация, име на пътника и първа/втора класа). Ако не бъде намерено съвпадение, функцията приключва.

11.3.1. Входни данни на функцията

Функцията приема масив от полети от тип `flights` и размера на масива като параметри. Размера на масива показва колко пъти ще се повтори цикълът.

11.3.2. Изходни данни на функцията или данни, които се извеждат

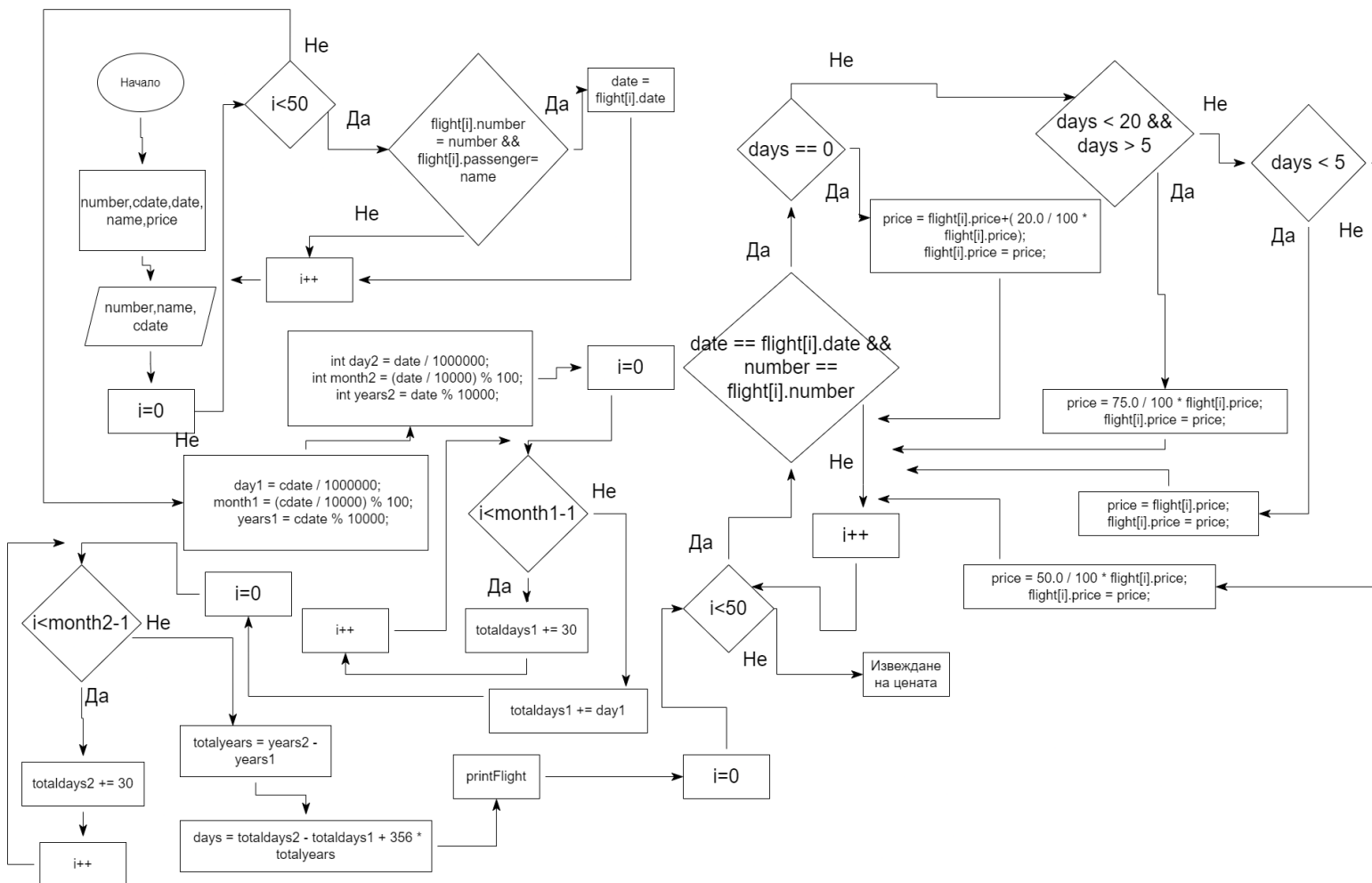
Извежда всички елементи на масива, отговарящи на датата и дестинацията в табличен вид.

12. Реализация на условие Н -а - допълнение второ

12.1. Анализ на алгоритъма, който трябва да се реализира

1. Въвежда се номер на полет, име на пътник и текущата дата. Стойността им се съхранява в променливи.
2. Обхожда се масива, за да се намери полет, съответстващ на въведеното име и номер. Ако открие такъв полет, датата на полета се съхранява в променлива, наречена date.
3. Извлича се деня, месеца и годината от cdate и date.
4. Изчислява се общия брой дни между cdate и датата.
5. Извежда се информация за полетите
6. Масивът се обхожда наново , за да се намери полет, съответстващ на въведеното име и номер. Ако намери такъв полет, той коригира цената на полета въз основа на броя дни между текущата датата и датата на полета.
7. Извежда се коригираната цена.

12.2. Блок схема на алгоритъма



12.3. Функция с която е реализиран алгоритъма

```
void currentPrice(flights flight[], int size);
void currentPrice(flights flight[], int size) {
    int number, cdate = 0, date = 0;
    char name[100];
    double price = 0;
    cout << "\nВъведете номер :";
    cin >> number;
    cout << "\nВъведете име на пътник:";
    cin.ignore();
    cin.getline(name, 100);
    cout << "\nВъведете днешна дата:";
    cin >> cdate;

    for (int i = 0; i < size; i++) {
        if (flight[i].number == number && (strcmp(flight[i].passenger, name) == 0)) {
            date = flight[i].date;
        }
    }

    int day1 = cdate / 1000000;
    int month1 = (cdate / 10000) % 100;
    int years1 = cdate % 10000;

    int day2 = date / 1000000;
    int month2 = (date / 10000) % 100;
    int years2 = date % 10000;

    int totaldays1 = 0;
    for (int i = 0; i < month1 - 1; i++) {
        totaldays1 += 30;
    }
    totaldays1 += day1;

    int totaldays2 = 0;
    for (int i = 0; i < month2 - 1; i++) {
        totaldays2 += 30;
    }
    totaldays2 += day2;
```



```

int totalyears = years2 - years1;

int days = totaldays2 - totaldays1 + 356 * totalyears;
printFlight(flight, size);

for (int i = 0; i < size; i++) {
    if (date == flight[i].date && number == flight[i].number) {
        if (days == 0){
            price = flight[i].price+( 20.0 / 100 * flight[i].price);
            flight[i].price = price;
        }
        else if (days < 20 && days > 5){
            price = 75.0 / 100 * flight[i].price;
            flight[i].price = price;
        }
        else if (days < 5){
            price = flight[i].price;
            flight[i].price = price;
        }
        else{
            price = 50.0 / 100 * flight[i].price;
            flight[i].price = price;
        }
    }
} cout << "\nЦената на билета е " << price<<"лв.";

}

```

Въвежда се номера на полета и името на пътника, а след това текущата дата. След това функцията преминава през списъка с полети и намира датата на полета с дадения номер и име на пътника.

Първо се извлича деня, месеца и годината на всяка дата от променливите cdate(текуща дата) и date(дата на полета). След това изчислява общия брой дни през първия месец и всички предходни месеци чрез добавяне на 30 дни за всеки месец (приемайки, че всеки месец има 30 дни). След това добавя броя на дните в текущия месец, за да получи общия брой дни от началото на годината до текущата дата. Прави същото за втората дата. И накрая, функцията изчислява общия брой години между двете дати, като извади годината на първата дата от годината на

втората дата и умножи това по 356 (приемайки, че всяка година има 356 дни). След това изчислява общия брой дни между двете дати, като извади общия брой дни на текущата дата от общия брой дни на датата на полета и добави броя на дните в допълнителните години.

След това функцията отново преминава през списъка с полети и намира полета с дадения номер и дата. Изчислява цената на билета въз основа на броя дни между текущата дата и датата на полета. Ако до полета има по-малко от 20 и повече от 5 дена, цената се намалява с 25%. Ако дните са по-малко от 5, цената остава същата. При брой дни над 20, цената се намалява с 50%. Ако датите съвпадат, цената се увеличава с 20%. След това окончателната изчислена цена се отпечатва на конзолата.

12.3.1. Входни данни на функцията

Функцията приема масив от полети от тип `flights` и размера на масива като параметри. Размера на масива показва колко пъти ще се повтори цикълът.

12.3.2. Изходни данни на функцията или данни, които се извеждат

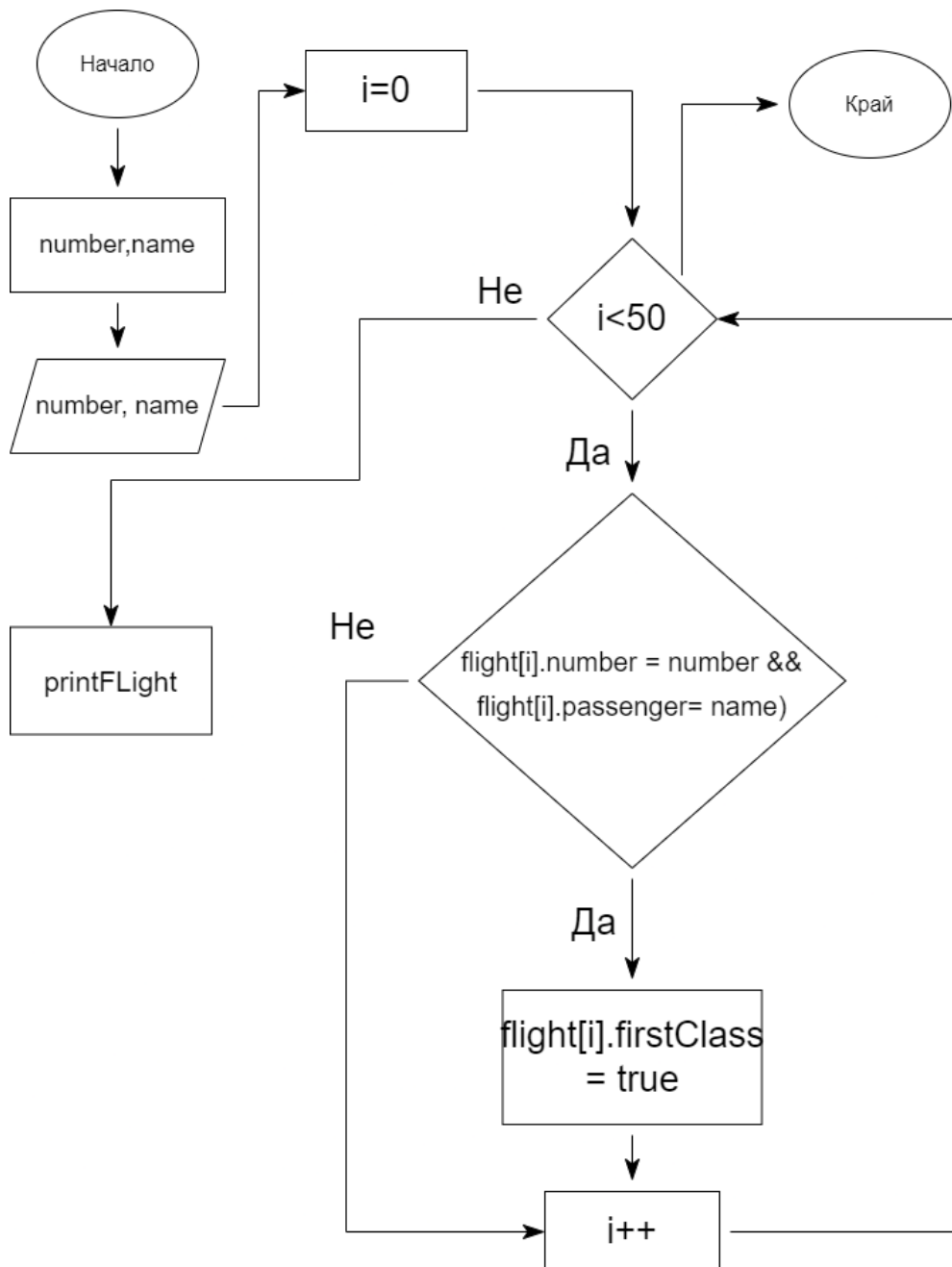
Извежда всички елементи на масива в табличен вид и цената на полета.

13. Реализация на условие H -b - допълнение второ

13.1. Анализ на алгоритъма, който трябва да се реализира

1. Въвежда се номер на полет и име на пътник;
2. Функцията търси полет с посочения номер и име.
3. Ако намери такъв полет, полето `firstClass` на този полет се задава на `true`.
4. Извежда се информация за полетите

13.2. Блок схема на алгоритъма



13.3. Функция с която е реализиран алгоритъма

```
void switchClass(flights flight[], int size) {
    int number;
    char name[100];
    cout << "\nВъведете номер :";
    cin >> number;
    cout << "\nВъведете име на пътник:";
    cin.ignore();
    cin.getline(name, 100);
    for (int i = 0; i < size; i++)
```

```

        if (flight[i].number == number && (strcmp(flight[i].passenger, name) == 0))
            flight[i].firstClass = true;
    printFlight(flight, size);
}

```

Въвежда се номер на полет и името на пътника от клавиатурата. След това функцията обхожда масива от полети с помощта на цикъл и сравнява номера на всеки полет с въведения номер на полет и полето на пътника с въведеното име с помощта на функцията `strcmp()`. Ако бъде намерено съвпадение, полето `firstClass` на съпадащия полет се задава на `true`. И накрая, функцията извиква функцията `printFlight()`, за да отпечата актуализирания списък с полети.

13.3.1. Входни данни на функцията

Функцията приема масив от полети от тип `flights` и размера на масива като параметри. Размера на масива показва колко пъти ще се повтори цикълът.

13.3.2. Изходни данни на функцията или данни, които се извеждат

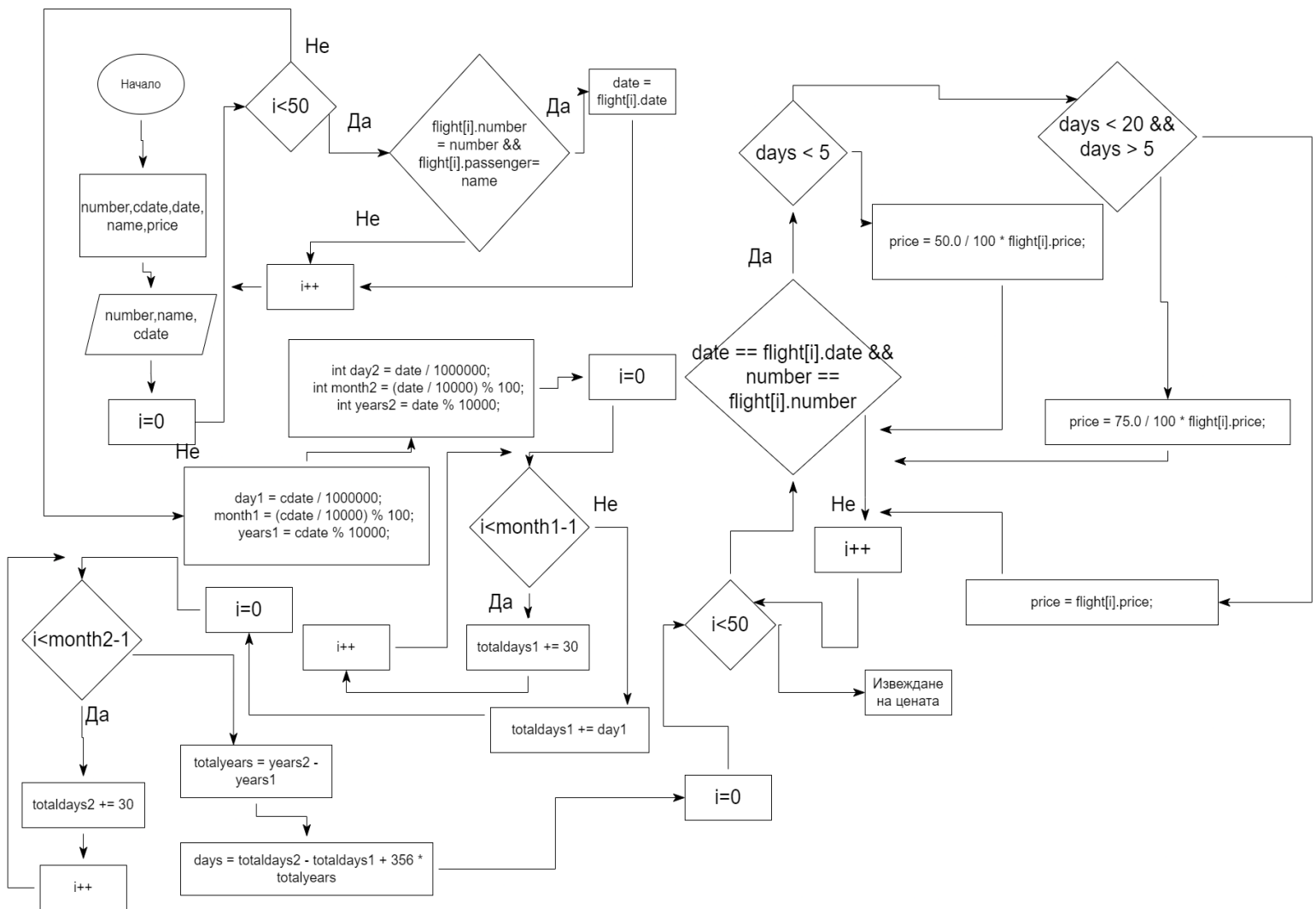
Извежда всички елементи на масива в табличен вид.

14. Реализация на условие I - допълнение второ

14.1. Анализ на алгоритъма, който трябва да се реализира

1. Въвежда се номер на полет, име на пътник и текущата дата. Стойността им се съхранява в променливи.
2. Обхожда се масива, за да се намери полет, съответстващ на въведеното име и номер. Ако открие такъв полет, датата на полета се съхранява в променлива, наречена `date`.
3. Извлича се деня, месеца и годината от `cdate` и `date`.
4. Изчислява се общия брой дни между `cdate` и `date`.
5. Масивът се обхожда наново, за да се намери полет, съответстващ на въведеното име и номер. Ако намери такъв полет, той коригира цената на полета въз основа на броя дни между текущата датата и датата на полета.
6. Извежда се коригираната сума за връщане.

14.2. Блок схема на алгоритъма



14.3. Функция с която е реализиран алгоритъма

Въвежда се номера на полета, името на пътника, текущата дата. След това функцията обхожда масива от полети с помощта на цикъл и търси съвпадение между въведения номер на полета и името на пътника и полетата за номер и пътник за всеки полет с помощта на функцията strcmp(). Ако бъде намерено съвпадение, датата на съвпадащия полет се съхранява в отделна променлива. След това функцията изчислява броя на дните между текущата дата и датата на полета, като използва поредица от изчисления, които преобразуват датите в общия брой дни от началото на годината. Накрая функцията изчислява сумата, която трябва да бъде възстановена на пътника въз основа на броя дни преди полета. Ако броят на дните е по-малък от 5, възстановя се е 50% от цената на полета. Ако броят на дните е между 5 и 20, възстановя се 75% от цената на полета. В противен случай се възстановява пълната цена на полета. След това функцията отпечатва сумата за възстановяване.

14.3.1. Входни данни на функцията

Функцията приема масив от полети от тип `flights` и размера на масива като параметри, Размера на масива показва колко пъти ще се повтори цикълът.

14.3.2. Изходни данни на функцията или данни, които се извеждат

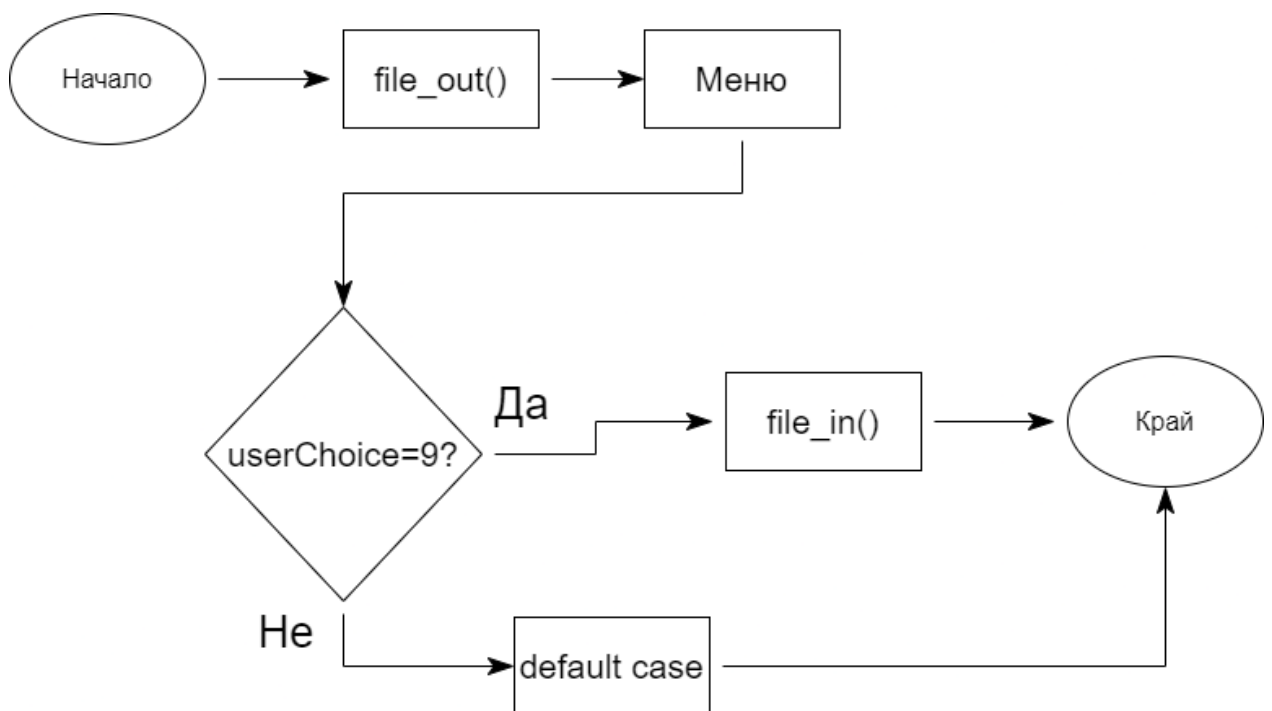
Извежда всички елементи на масива в табличен вид и сумата за връщане.

15. Реализация на допълнение трето

15.1. Анализ на алгоритъма, който трябва да се реализира

1. Главната функция извиква функцията за въвеждане на масив от файл.
2. Извежда менюто и обработва избора на потребителя.
3. Ако потребителят избере да излезе от програмата (опция 9) се изпълнява функцията за извеждане във файл.

15.2. Блок схема на алгоритъма



15.3. Функция с която е реализиран алгоритъма

```
int main()
{
    setlocale(LC_ALL, "BG");
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    const int size = 50;
    flights flight[size];
    int n = 0, m = 0;
```

```

unsigned short userChoice2 = 0, userChoice = 0;
file_in(flight);
do {
    cout << "\nИзберете опция от менюто\n";
    cout << "1.Добавяне на един полет\n";
    cout << "2.Добавяне на n полета\n";
    cout << "3.Извеждане на полетите\n";
    cout << "4.Полети с най-ниска цена\n";
    cout << "5.Извеждане на полети на даден пилот\n";
    cout << "6.Извеждане във файл\n";
    cout << "7.Подреждане на полетите в низходящ ред по азбучен ред на
дестинацията\n";
    cout << "8.Подменю\n";
    cout << "9.Изход\n";
    cout << "Избор:";
    cin >> userChoice;
    switch (userChoice) {
    case 1:
        enterFlight(flight, 1, m);
        break;
    case 2:
        enterFlight(flight, n, m);
        break;
    case 3:
        printFlight(flight, size);
        break;
    case 4:
        leastExpensive(flight, size);
        break;
    case 5:
        printPilot(flight, size);
        break;
    case 6:
        file_out(flight, size, m);
        break;
    case 7:
        destinationAlph(flight, size);
        break;
    case 8:
        do {

```

```

cout << "\nИзберете опция от менюто ";
    cout << "\n1.Сортиране във възходящ ред на цената на полета и
класата на полета";
    cout << "\n2.Търсене по въведена дата и определена дестинация";
    cout << "\n3.Цена на билет: ";
    cout << "\n4.Замяна на билет от втора класа";
    cout << "\n5.Отказ от полет";

    cout << "\n6.Назад ";
    cout << "\nИзбор:";
    cin >> userChoice2;
    switch (userChoice2) {
    case 1:
        classPriceAsc(flight, size);
        break;
    case 2:
        searchByDateDestin(flight, size);
        break;
    case 3:
        currentPrice(flight, size);
        break;
    case 4:
        switchClass(flight, size);
        break;
    case 5:
        cancelFlight(flight, size);
        break;
    case 6:
        file_out(flight, size, m);
        break;
    default:
        cout << "Въведете валиден номер на операция:" << endl;
        break;
    }
} while (userChoice2 != 6);
break;
case 9:
    file_out(flight,size,m);
    break;
default:

```



```

        cout << "Въведете валиден номер на операция : "<<endl;
        break;
    }
} while (userChoice != 9);
return 0;
}

```

Функцията за извеждане на данни от файл се изпълнява преди менюто. А функцията за въвеждане на данни във файл се изпълнява когато потребителят избере опцията за изход от програмата (номер 9).

15.3.1. Входни данни на функцията

Главната функция не приема параметри.

15.3.2. Изходни данни на функцията или данни, които се извеждат

Опциите на менюто се извеждат и потребителят въвежда своя избор. Изпълнява се кодът за съответното действие.

III. Упътване за употреба

1. Условие А

1.1. Очаквани входни данни

Не се въвеждат входни данни.

1.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Извеждат се опциите на менюто.

2. Условие В-а

2.1. Очаквани входни данни

Избор:1

Номер: 7

Дата: 27012021

Цена: 130

Пилот: Петър Петров

Първа(1)/Втора(0) класа: 0

Име на пътник: Анна Василева

Дестинация: Берлин

2.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Информацията за полета се запазва в масива.

3. Условие B-b

3.1. Очаквани входни данни

Избор:2

2

Номер: 8

Дата: 13112022

Цена: 250

Пилот: Петър Петров

Първа(1)/Втора(0) класа: 0

Име на пътник: Николай Христов

Дестинация: Лондон

Номер: 9

Дата: 28122022

Цена: 500

Пилот: Петър Петров

Първа(1)/Втора(0) класа: 1

Име на пътник: Анна Василева

Дестинация: Дубай

3.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Информацията за полета се запазва в масива. Ако се въведе невалиден брой полети се извежда съобщение за грешка.

4. Условие C

4.1. Очаквани входни данни

Избор:3

4.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Извежда всички елементи на масива в табличен вид.

5. Условие D-a

5.1. Очаквани входни данни

Избор:4

5.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Извежда всички елементи с най- ниска цена в табличен вид.

6. Условие D-b

6.1. Очаквани входни данни

Избор:5

Въведете име на пилот : Милена Стоянова

6.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Извежда всички полети на въведения пилот в табличен вид.

7. Условие E

7.1. Очаквани входни данни

Избор:7

7.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Извежда всички полети в табличен вид, подредени в низходящ ред по азбучен ред на дестинацията.

8. Условие F-a

8.1. Очаквани входни данни

Избор:9

8.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Данните на масива се записват в двоичен файл.

9. Условие F-b

9.1. Очаквани входни данни

Не се въвеждат входни данни

9.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Въвежда се елементите на масива от двоичен файл.

10. Условие G-a

10.1. Очаквани входни данни

Избор:8

Избор:1

10.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Извежда всички полети във възходящ ред на цената на полета и класата на полета

11. Условие G-b

11.1. Очаквани входни данни

Избор:2

Въведете дата:10102022

Въведете дестинация:Истанбул

11.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Извежда всички полети, отговарящи на въведената дата и дестинация.

12. Условие H-a

12.1. Очаквани входни данни

Избор:8

Избор:3

Въведете номер :3

Въведете име на пътник:Мартин Петров

Въведете днешна дата:1122022

12.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Извежда всички полети в табличен вид и цената на полета.

13. Условие H-b

13.1. Очаквани входни данни

Избор:8

Избор:4

Въведете номер :4

Въведете име на пътник:Мария Костова

13.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Извежда всички полети в табличен вид и като заменя билета на въведения пътник с такъв от първа класа .

14. Условие I

14.1. Очаквани входни данни

Избор:8

Избор:5

Въведете номер :6

Въведете име на пътник:Гергана Йорданова

Въведете днешна дата:15032021

14.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Извежда се сумата за връщане.

15. Условие J

15.1. Очаквани входни данни

Избор:9

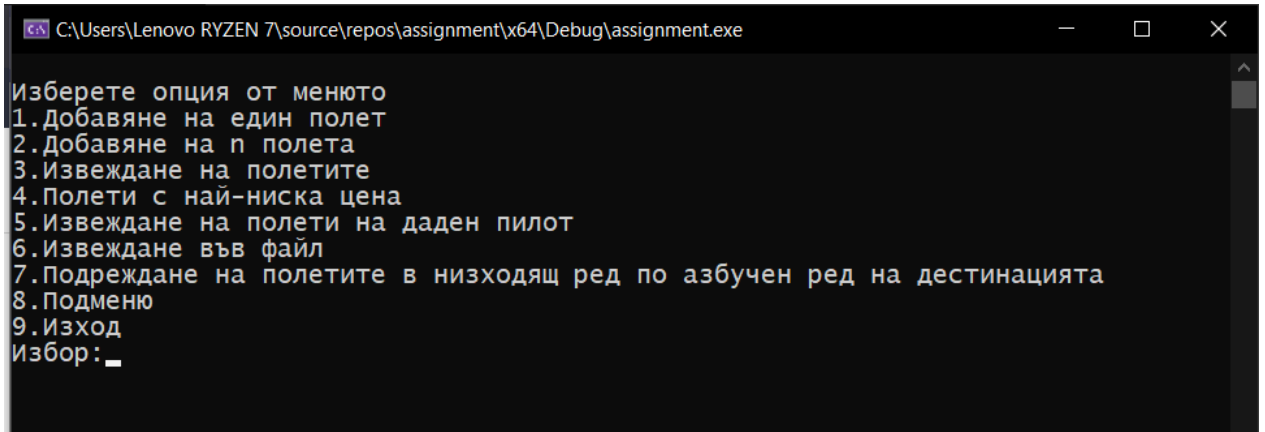
15.2. Очакван резултат от изпълнението на конкретния фрагмент от проекта

Данните на масива се запазват в двоичен файл.

IV. Примерно действие на програмата

1. Условие А

1.1. Снимка на изгледа с примерни входни данни

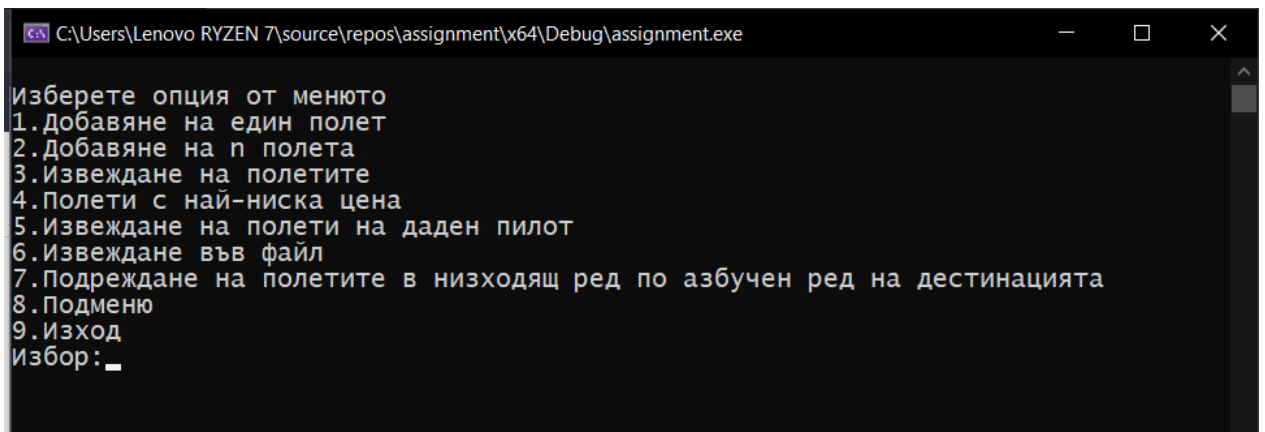


A screenshot of a Windows application window titled "C:\Users\Lenovo RYZEN 7\source\repos\assignment\x64\Debug\assignment.exe". The window has a dark background and displays a menu in Bulgarian. The menu items are: "Изберете опция от менюто", "1.Добавяне на един полет", "2.Добавяне на n полета", "3.Извеждане на полетите", "4.Полети с най-ниска цена", "5.Извеждане на полети на даден пилот", "6.Извеждане във файл", "7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията", "8.Подменю", "9.Изход", and "Избор: _". The cursor is positioned at the end of the "Избор:" line.

```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\x64\Debug\assignment.exe

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор: _
```

1.2. Снимка на изгледа с примерни изходни данни



A screenshot of the same application window as above, showing the menu. The menu items are identical to the previous screenshot. The cursor is positioned at the end of the "Избор:" line.

```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\x64\Debug\assignment.exe

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор: _
```

2. Условие В-а

2.1. Снимка на изгледа с примерни входни данни

```
Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:1

Въведете полет:

Номер: 7
Дата: 18052021
Цена: 185.9
Пилот: Петър Петров
Първа(1)/Втора(0) класа: 1
Име на пътник: Анна Василева
Дестинация: Дубай
```

2.2. Снимка на изгледа с примерни изходни данни

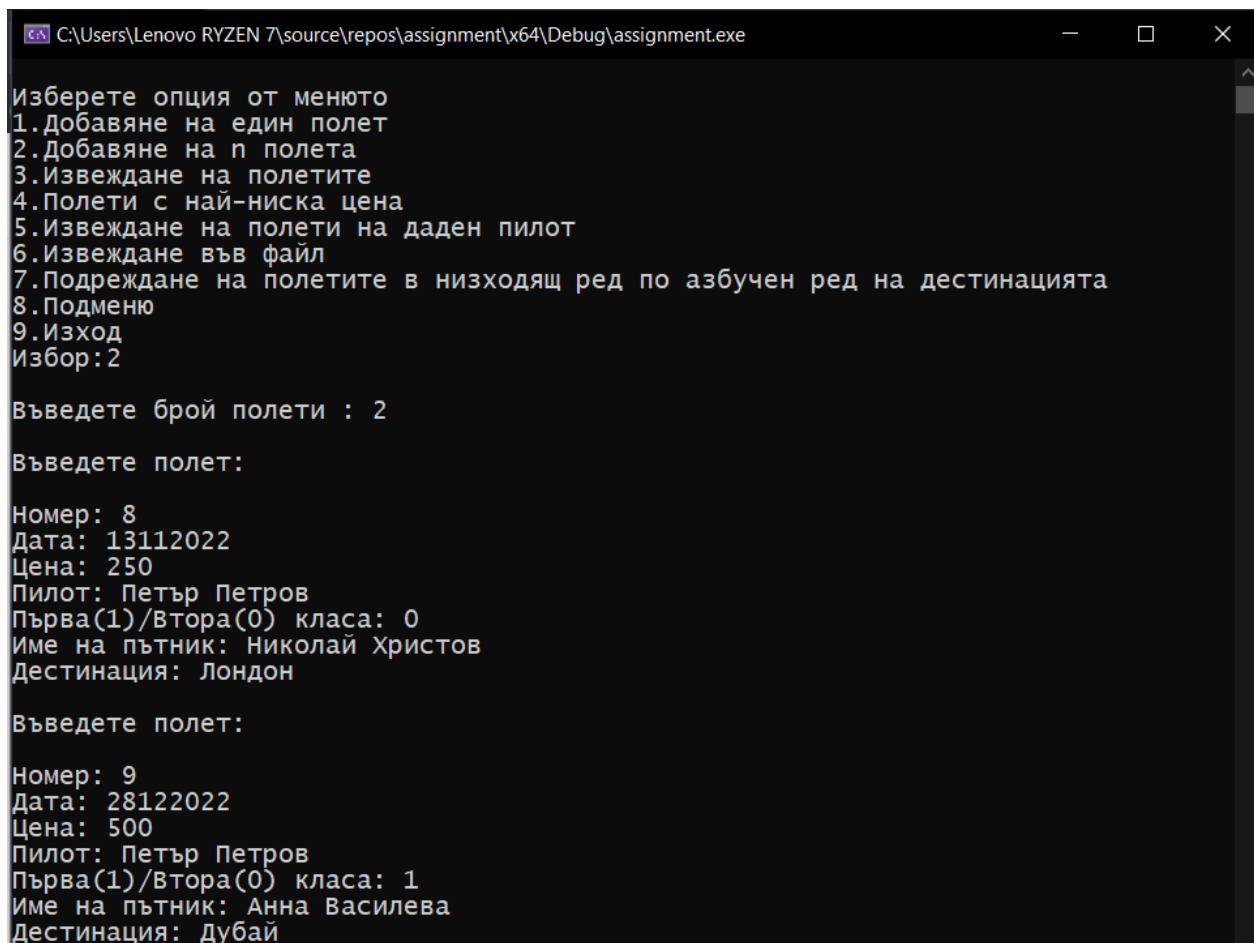
```
Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:1

Въведете полет:

Номер: 7
Дата: 18052021
Цена: 185.9
Пилот: Петър Петров
Първа(1)/Втора(0) класа: 1
Име на пътник: Анна Василева
Дестинация: Дубай
```

3. Условие B-b

3.1. Снимка на изгледа с примерни входни данни



```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\x64\Debug\assignment.exe

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:2

Въведете брой полети : 2

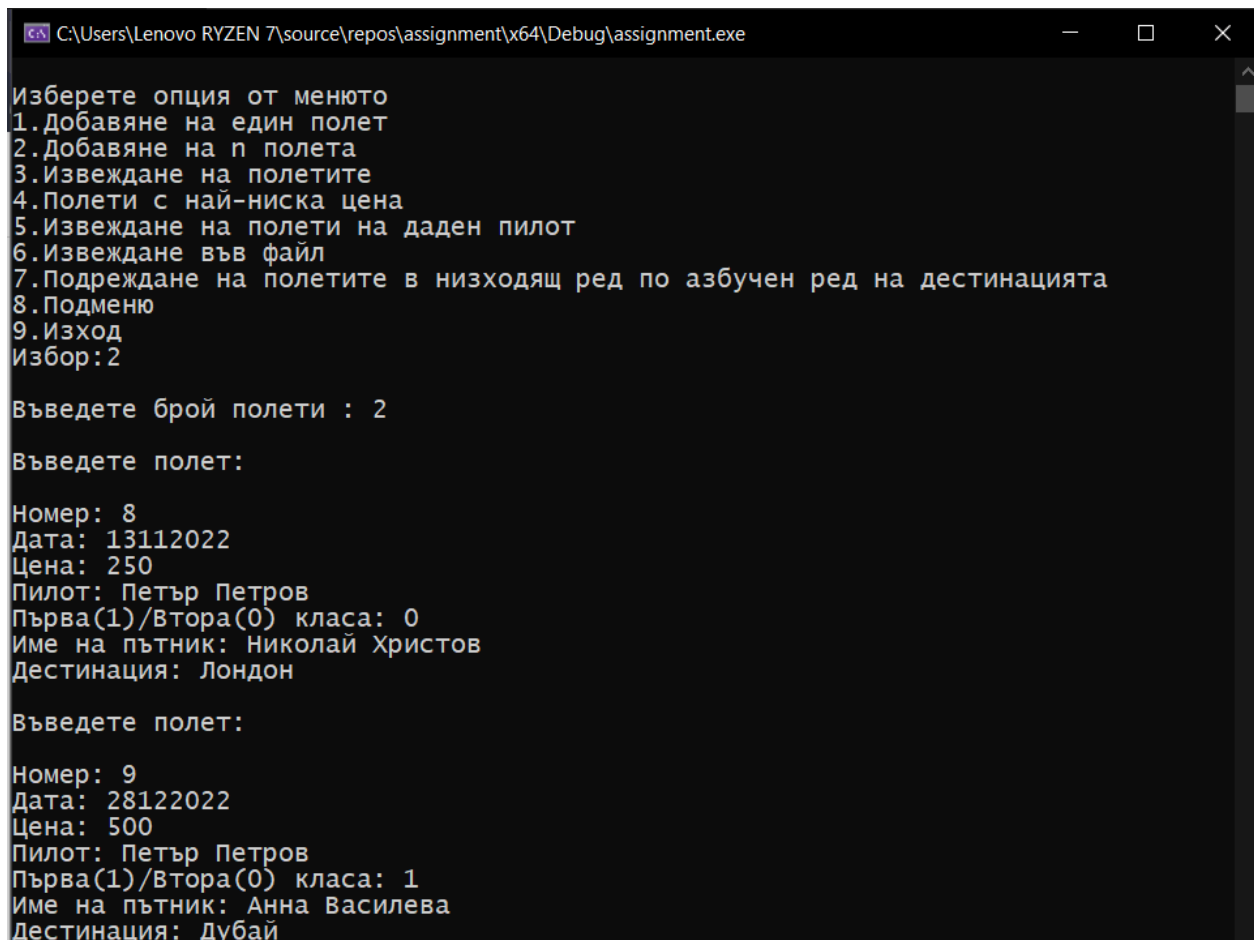
Въведете полет:

Номер: 8
Дата: 13112022
Цена: 250
Пилот: Петър Петров
Първа(1)/Втора(0) класа: 0
Име на пътник: Николай Христов
Дестинация: Лондон

Въведете полет:

Номер: 9
Дата: 28122022
Цена: 500
Пилот: Петър Петров
Първа(1)/Втора(0) класа: 1
Име на пътник: Анна Василева
Дестинация: Дубай
```


3.2. Снимка на изгледа с примерни изходни данни



```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\x64\Debug\assignment.exe

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подмену
9.Изход
Избор:2

Въведете брой полети : 2

Въведете полет:

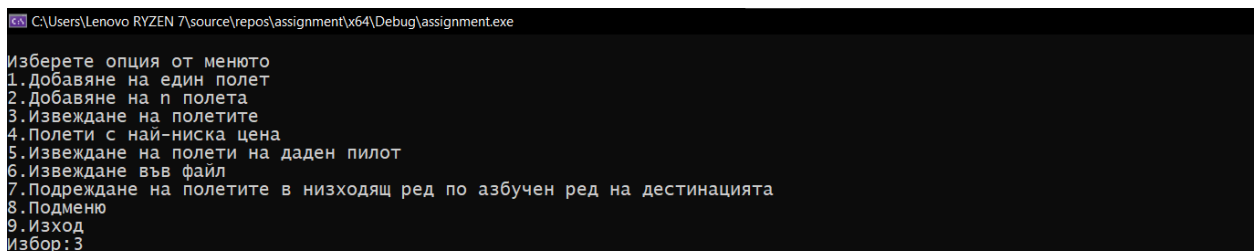
Номер: 8
Дата: 13112022
Цена: 250
Пилот: Петър Петров
Първа(1)/Втора(0) класа: 0
Име на пътник: Николай Христов
Дестинация: Лондон

Въведете полет:

Номер: 9
Дата: 28122022
Цена: 500
Пилот: Петър Петров
Първа(1)/Втора(0) класа: 1
Име на пътник: Анна Василева
Дестинация: Дубай
```

4. Условие C

4.1. Снимка на изгледа с примерни входни данни



```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\x64\Debug\assignment.exe

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подмену
9.Изход
Избор:3
```

4.2. Снимка на изгледа с примерни изходни данни

```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\src\Debug\assignment.exe

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:3



| Номер | Дата     | Цена   | Пилот             | Дестинация | Име на пътник      | Първа/Втора класа |
|-------|----------|--------|-------------------|------------|--------------------|-------------------|
| 1     | 11122022 | 199.99 | Георги Иванов     | Лондон     | Виктория Михайлова | 1                 |
| 2     | 15122022 | 250    | Георги Иванов     | Париж      | Димитър Христов    | 1                 |
| 3     | 30122022 | 100    | Милена Стоянова   | Мадрид     | Мартин Петров      | 0                 |
| 4     | 10102022 | 100    | Милена Стоянова   | Истанбул   | Мария Костова      | 0                 |
| 5     | 5022021  | 300    | Валентин Георгиев | Барселона  | Александър Иванов  | 1                 |
| 6     | 15032021 | 150    | Валентин Георгиев | Ню Йорк    | Гергана Йорданова  | 0                 |


```

5. Условие D-a

5.1. Снимка на изгледа с примерни входни данни

```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\src\Debug\assignment.exe

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:4
```

5.2. Снимка на изгледа с примерни изходни данни

```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\src\Debug\assignment.exe

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:4



| Номер | Дата     | Цена | Пилот           | Дестинация | Име на пътник | Първа/Втора класа |
|-------|----------|------|-----------------|------------|---------------|-------------------|
| 3     | 30122022 | 100  | Милена Стоянова | Мадрид     | Мартин Петров | 0                 |
| 4     | 10102022 | 100  | Милена Стоянова | Истанбул   | Мария Костова | 0                 |


```

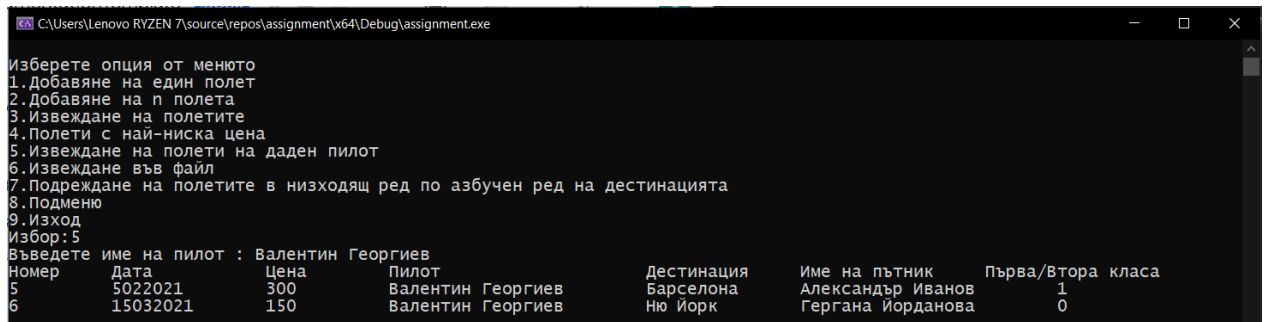
6. Условие D-b

6.1. Снимка на изгледа с примерни входни данни

```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\src\Debug\assignment.exe

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:5
```

6.2. Снимка на изгледа с примерни изходни данни



```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\assignment.exe

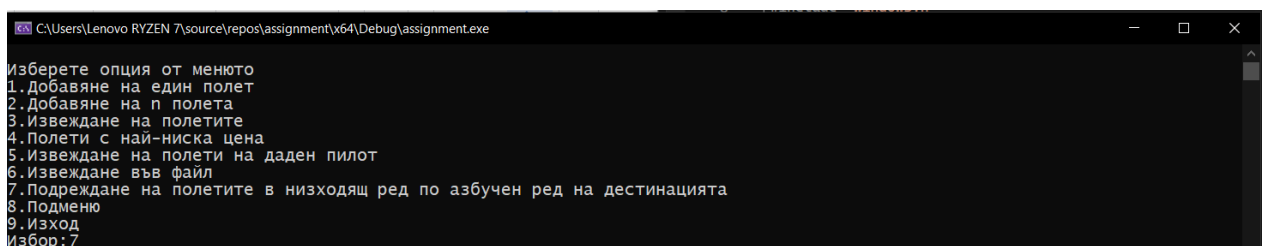
Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на п полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:5
Въведете име на пилот : Валентин Георгиев


| Номер | Дата     | Цена | Пилот             | Дестинация | Име на пътник     | Първа/Втора класа |
|-------|----------|------|-------------------|------------|-------------------|-------------------|
| 5     | 5022021  | 300  | Валентин Георгиев | Барселона  | Александър Иванов | 1                 |
| 6     | 15032021 | 150  | Валентин Георгиев | Ню Йорк    | Гергана Йорданова | 0                 |


```

7. Условие E

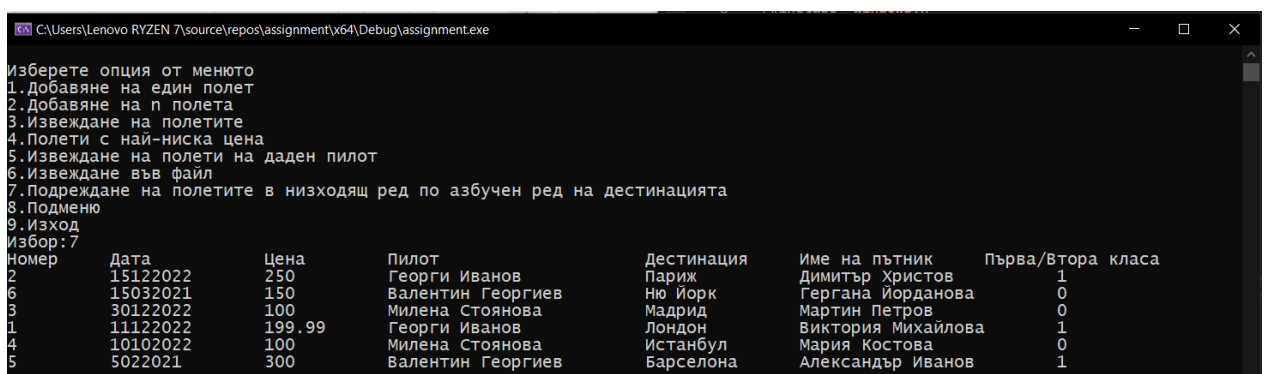
7.1. Снимка на изгледа с примерни входни данни



```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\assignment.exe

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на п полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:7
```

7.2. Снимка на изгледа с примерни изходни данни



```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\assignment.exe

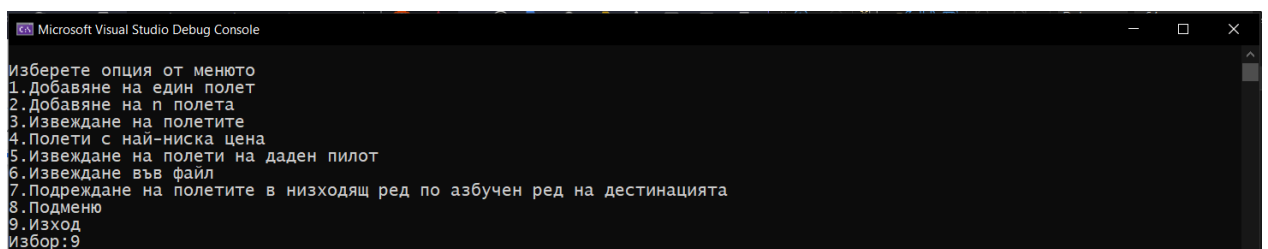
Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на п полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:7


| Номер | Дата     | Цена   | Пилот             | Дестинация | Име на пътник      | Първа/Втора класа |
|-------|----------|--------|-------------------|------------|--------------------|-------------------|
| 2     | 15122022 | 250    | Георги Иванов     | Париж      | Димитър Христов    | 1                 |
| 6     | 15032021 | 150    | Валентин Георгиев | Ню Йорк    | Гергана Йорданова  | 0                 |
| 3     | 30122022 | 100    | Милена Стоянова   | Мадрид     | Мартин Петров      | 0                 |
| 1     | 11122022 | 199.99 | Георги Иванов     | Лондон     | Виктория Михайлова | 1                 |
| 4     | 10102022 | 100    | Милена Стоянова   | Истанбул   | Мария Костова      | 0                 |
| 5     | 5022021  | 300    | Валентин Георгиев | Барселона  | Александър Иванов  | 1                 |


```

8. Условие F-a

8.1. Снимка на изгледа с примерни входни данни



```
Microsoft Visual Studio Debug Console

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на п полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:9
```

8.2. Снимка на изгледа с примерни изходни данни

Данните на масива се записват в двоичен файл.

9. Условие F-b

9.1. Снимка на изгледа с примерни входни данни

Не се въвеждат входни данни

9.2. Снимка на изгледа с примерни изходни данни

Въвежда се елементите на масива от двоичен файл.

10. Допълнение първо - G-a

10.1. Снимка на изгледа с примерни входни данни

```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\assignment.exe

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:8

Изберете опция от менюто
1.Сортиране във възходящ ред на цената на полета и класата на полета
2.Търсене по въведена дата и определена дестинация
3.Цена на билет:
4.Замяна на билет от втора класа
5.Отказ от полет
6.Назад
Избор:1
```

10.2. Снимка на изгледа с примерни изходни данни

```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\assignment.exe

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:8

Изберете опция от менюто
1.Сортиране във възходящ ред на цената на полета и класата на полета
2.Търсене по въведена дата и определена дестинация
3.Цена на билет:
4.Замяна на билет от втора класа
5.Отказ от полет
6.Назад
Избор:1

Полети във възходящ ред на цената:
номер    Дата    Цена    Пилот    Дестинация    Име на пътник    Първа/Втора класа
3         30122022    100    Милена Стоянова    Мадрид    Мартин Петров    0
4         10102022    100    Милена Стоянова    Истанбул    Мария Костова    0
6         15032021    150    Валентин Георгиев    Ню Йорк    Гергана Йорданова    0
1         11122022    199.99    Георги Иванов    Лондон    Виктория Михайлова    1
2         15122022    250    Георги Иванов    Париж    Димитър Христов    1
5         5022021    300    Валентин Георгиев    Барселона    Александър Иванов    1

Полети във възходящ ред на класа на полета
номер    Дата    Цена    Пилот    Дестинация    Име на пътник    Първа/Втора класа
3         30122022    100    Милена Стоянова    Мадрид    Мартин Петров    0
4         10102022    100    Милена Стоянова    Истанбул    Мария Костова    0
6         15032021    150    Валентин Георгиев    Ню Йорк    Гергана Йорданова    0
1         11122022    199.99    Георги Иванов    Лондон    Виктория Михайлова    1
2         15122022    250    Георги Иванов    Париж    Димитър Христов    1
5         5022021    300    Валентин Георгиев    Барселона    Александър Иванов    1
```

11. Допълнение първо -G-b

11.1. Снимка на изгледа с примерни входни данни

```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\assignment.exe
6      15032021      150      Валентин Георгиев      Нью Йорк      Гертана Йорданова      0

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на п полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подмену
9.Изход
Избор:8

Изберете опция от менюто
1.Сортиране във възходящ ред на цената на полета и класата на полета
2.Търсене по въведена дата и определена дестинация
3.Цена на билет:
4.Замяна на билет от втора класа
5.Отказ от полет
6.Назад
Избор:2
```

11.2. Снимка на изгледа с примерни изходни данни

```
C:\Users\Lenovo RYZEN 7\source\repos\assignment\assignment.exe
6      15032021      150      Валентин Георгиев      Нью Йорк      Гертана Йорданова      0

Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на п полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подмену
9.Изход
Избор:8

Изберете опция от менюто
1.Сортиране във възходящ ред на цената на полета и класата на полета
2.Търсене по въведена дата и определена дестинация
3.Цена на билет:
4.Замяна на билет от втора класа
5.Отказ от полет
6.Назад
Избор:2

Въведете дата:5022021
Въведете дестинация:Барселона
номер    дата    Цена    Пилот    Дестинация    Име на пътник    Първа/Втора класа
5        5022021    300    Валентин Георгиев    Барселона    Александър Иванов    1
```

12. Допълнение второ - Н-а

12.1. Снимка на изгледа с примерни входни данни

```
Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:8

Изберете опция от менюто
1.Сортиране във възходящ ред на цената на полета и класата на полета
2.Търсене по въведена дата и определена дестинация
3.Цена на билет:
4.Замяна на билет от втора класа
5.Отказ от полет
6.Назад
Избор:3

Въведете номер :4

Въведете име на пътник:Мария Костова

Въведете днешна дата:1102022
```

12.2. Снимка на изгледа с примерни изходни данни

```
Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:8

Изберете опция от менюто
1.Сортиране във възходящ ред на цената на полета и класата на полета
2.Търсене по въведена дата и определена дестинация
3.Цена на билет:
4.Замяна на билет от втора класа
5.Отказ от полет
6.Назад
Избор:3

Въведете номер :4

Въведете име на пътник:Мария Костова

Въведете днешна дата:1102022
```

Номер	Дата	Цена	Пилот	Дестинация	Име на пътник	Първа/Втора класа
1	11122022	199.99	Георги Иванов	Лондон	Виктория Михайлова	1
2	15122022	250	Георги Иванов	Париж	Димитър Христов	1
3	30122022	100	Милена Стоянова	Мадрид	Мартин Петров	0
4	10102022	100	Милена Стоянова	Истанбул	Мария Костова	0
5	5022021	300	Валентин Георгиев	Барселона	Александър Иванов	1
6	15032021	150	Валентин Георгиев	Ню Йорк	Гергана Йорданова	0

Цената на билета е 75лв.

13. Допълнение второ - H-b

13.1. Снимка на изгледа с примерни входни данни

```
Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:8

Изберете опция от менюто
1.Сортиране във възходящ ред на цената на полета и класата на полета
2.Търсене по въведена дата и определена дестинация
3.Цена на билет:
4.Замяна на билет от втора класа
5.Отказ от полет
6.Назад
Избор:4

Въведете номер :6

Въведете име на пътник:Гергана Йорданова
Име на пътник: Пилот Цена Билет Дестинация
```

13.2. Снимка на изгледа с примерни изходни данни

```
Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:8

Изберете опция от менюто
1.Сортиране във възходящ ред на цената на полета и класата на полета
2.Търсене по въведена дата и определена дестинация
3.Цена на билет:
4.Замяна на билет от втора класа
5.Отказ от полет
6.Назад
Избор:4

Въведете номер :6

Въведете име на пътник:Гергана Йорданова
Име на пътник: Пилот Цена Билет Дестинация Име на пътник Първа/Втора класа
1 11122022 199.99 Георги Иванов Лондон Виктория Михайлова 1
2 15122022 250 Георги Иванов Париж Димитър Христов 1
3 30122022 100 Милена Стоянова Мадрид Мартин Петров 0
4 10102022 100 Милена Стоянова Истанбул Мария Костова 0
5 5022021 300 Валентин Георгиев Барселона Александър Иванов 1
6 15032021 150 Валентин Георгиев Ню Йорк Гергана Йорданова 1
```

14. Допълнение второ - I-a

14.1. Снимка на изгледа с примерни входни данни

```
Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:8

Изберете опция от менюто
1.Сортиране във възходящ ред на цената на полета и класата на полета
2.Търсене по въведена дата и определена дестинация
3.Цена на билет:
4.Замяна на билет от втора класа
5.Отказ от полет
6.Назад
Избор:5

Въведете номер :6

Въведете име на пътник:Гергана Йорданова

Въведете днешна дата:15032021
0
```

14.2. Снимка на изгледа с примерни изходни данни

```
Изберете опция от менюто
1.Добавяне на един полет
2.Добавяне на n полета
3.Извеждане на полетите
4.Полети с най-ниска цена
5.Извеждане на полети на даден пилот
6.Извеждане във файл
7.Подреждане на полетите в низходящ ред по азбучен ред на дестинацията
8.Подменю
9.Изход
Избор:8

Изберете опция от менюто
1.Сортиране във възходящ ред на цената на полета и класата на полета
2.Търсене по въведена дата и определена дестинация
3.Цена на билет:
4.Замяна на билет от втора класа
5.Отказ от полет
6.Назад
Избор:5

Въведете номер :6

Въведете име на пътник:Гергана Йорданова

Въведете днешна дата:15032021
0

Сумата за връщане е 75лв.
```

15. Допълнение трето

15.1. Снимка на изгледа с примерни входни данни

Не се въвеждат входни данни

15.2. Снимка на изгледа с примерни изходни данни