Картина, която съдържа текст, графична колекция

Описанието е генерирано автоматичноТЕХНИЧЕСKИ УНИВЕРСИТЕТ – ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра „СИТ“

**СЕМЕСТРИАЛНА ДОМАШНА РАБОТА**

по дисциплината „Базово програмиране”

на тема: „ Информационна система „Сервиз за техника” ”

Вариант 199

|  |  |
| --- | --- |
| Изготвил: Кристиян Красимиров Кирилов | Проверил: |
| Специалност: СИТ |  |
| Група:2a |  |
| Факултетен номер: 23621649 |  |

2024

Съдържание

[Съдържание 2](#_Toc155087060)

[Задание на проекта 3](#_Toc155087061)

[Анализ на решението 4](#_Toc155087062)

[1. Константи, Изброими типове и Структури за данните в програмата 4](#_Toc155087063)

[2. Реализация на условие A 6](#_Toc155087064)

[3. Реализация на условие B 8](#_Toc155087065)

[4. Реализация на условие C 10](#_Toc155087066)

[5. Реализация на условие D 11](#_Toc155087067)

[6. Реализация на условие E 13](#_Toc155087068)

[7. Реализация на условие F 14](#_Toc155087069)

[8. Реализация на условие … - допълнение първо (+ базова задача) 15](#_Toc155087070)

[9. Реализация на условие … - допълнение второ (+ базова задача) 17](#_Toc155087071)

[10. Реализация на допълнение трето (+ базова задача) 19](#_Toc155087072)

[Примерно действие на програмата 20](#_Toc155087073)

[#Условие A 20](#_Toc155087074)

[Условие B 21](#_Toc155087075)

[Условие C 23](#_Toc155087076)

[Условие D 25](#_Toc155087077)

[Условие E 27](#_Toc155087078)

[Условие F 27](#_Toc155087079)

[Допълнение първо 31](#_Toc155087080)

[Допълнение второ 32](#_Toc155087081)

[Допълнение трето 36](#_Toc155087082)

Задание на проекта

**Информационна система „Сервиз за техника“**

Да се напише компютърна програма, реализираща информационна система за сервиз за техника. Програмата съхранява информация за извършените сервизни поръчки в рамките на даден месец (номер на поръчката, ден от месеца, клиент, тип устройство (телефон, лаптоп, телевизор и др.), сериен номер на устройството, проблем според клиента (до 50 символа), име на сервизния техник, извършен ремонт (до 50 символа), цена, дни престой в сервиза, статус: (приета/отказана/завършена). Предполагаемите проблеми и извършените ремонти да бъдат предварително декларирани в отделни списъци с максимален брой по 10 за всеки списък. Максималния брой сервизни поръчки е 100.

**Базова задача**

1. Меню за избор на функциите от програмата.

**Функции от програмата са:**

1. **Добавяне на нова сервизна поръчка:**

a. Добавяне на няколко поръчки, като не трябва да се превишава максималният брой (100).

Пример: Добавяне на списък от поръчки. Въвежда се цяло число n и след него n на брой поръчки. n не може да надвишава свободните елементи в списъка с поръчки. **\*При добавяне на поръчка се въвежда част от информацията (номер на**

**поръчка, вид устройство, номер, и т.н.). Име на сервизен техник, извършен**

**ремонт и т.н. се актуализират след извършения ремонт!**

1. **Извеждане на всички устройства на екрана:**

a. Извеждане на всички устройства, чрез подходящо форматиране в таблица

1. **Търсене и извеждане на екрана:**

a. Търсене и извеждане на устройства по вид

b. Търсене и извеждане на устройства по статус

1. **Сортиране и извеждане на поръчки на екран:**

a. Сортиране на поръчките по дата на приемане

1. **Работа с външен файл(двоичен)**

a. Извеждане на масива с поръчки във файл.

b. Въвеждане на масива с поръчки от файл.

***Допълнение първо (+ базова задача)***

1. **Да се създаде подменю към основното с нови функции за:**

a. Отделяне на завършените поръчки на определен техник и сортирани по ден на приемане.

b. Отделяне на поръчките с определен проблем за дадено устройство(въвежда се от потребителя) и сортиране в намаляващ ред по тип на устройството.

***Допълнение второ (+ базова задача)***

1. **Въвеждане/Актуализация на данни за сервизна поръчка:**

a. Добавяне в структурата на поле вид на поръчката (нормална – за 10 дена, бърза за 5 дена, експресна за 2 дена)

b. Въвеждане/Актуализацията на данни (за извършен ремонт) става по въведен номер

на сервизната поръчка. Ако поръчката присъства в списъка(масива):

**a**. Проверява се статуса й:

Приета поръчка – се актуализира информацията за извършеният ремонт и се пресмята цената.

- Изчисляване на цена: всеки тип ремонт има цена (в лв.) + процент от цената на ремонта за вида поръчка (експресна - 50%, бърза - 20%, нормална - 0% от цената на ремонт). Отказана/завършена – НЕ може да се актуализира информация.

**b.** Ако не присъства в масива, се извежда пояснително съобщение на екрана.

**I. Смяна на статус на поръчка:**

Въвежда се номерът на поръчката (ако я има се въвежда и новият статус). Ако статусът е приета се сменя с новият. Ако статусът е отказана/завършена НЕ СЕ допуска промяна!

***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**

Анализ на решението

Константи, Изброими типове и Структури за данните в програмата

// Деклариране на константи за максимални размери

const unsigned short MAX\_ORDER\_SIZE = 100; - максимален брой поръчки

const unsigned short CLIENT\_NAME\_LENGTH = 23; - максимална дължина на името на клиент

const unsigned short TECH\_TYPE\_LENGTH= 18; - максимална дължина на име на техника

const unsigned short SERIAL\_NUMBER\_LENGTH = 15; - максимална дължина на сериен номер на техника

const unsigned short PROBLEM\_LENGTH = 23; - максимална дължина на обозначение на проблема

const unsigned short TECHNICIAN\_NAME\_LENGTH = 23; - максимална дължина име на техник

const unsigned short RENOVATION\_LENGTH = 4; - максимална дължина обозначение на поправка

const unsigned short STATUS\_SIZE = 15; - максимална дължина на статус на поръчка

const unsigned short ORDER\_TYPE\_SIZE = 14; - максимална дължина на тип на поръчка

// Деклариране на константи масиви

const char INVALID\_CHOICE\_ERROR[]= = "Invalid choice!"; - масив от символи изписващ се при грешен избор

const char\* PROBLEMS[] = { "watered" , "broken screen","audio issue", "black screen", "connection problem", "lack of capacity","cannot be charged", "low battery life", "overheating", "broken keyboard" }; - масив от 10 указатели към константни масиви от символи представляващи проблеми на техники

const char\* RENOVATIONS [] = { "dried and changed parts", "replaced screen", "replaced audio system", "replaced old parts", "drivers update", "added memory", "new adapter", "replaced battery", "cooling added","replaced keycaps" };- масив от 10 указатели към константни масиви от символи представляващи поправки на техники

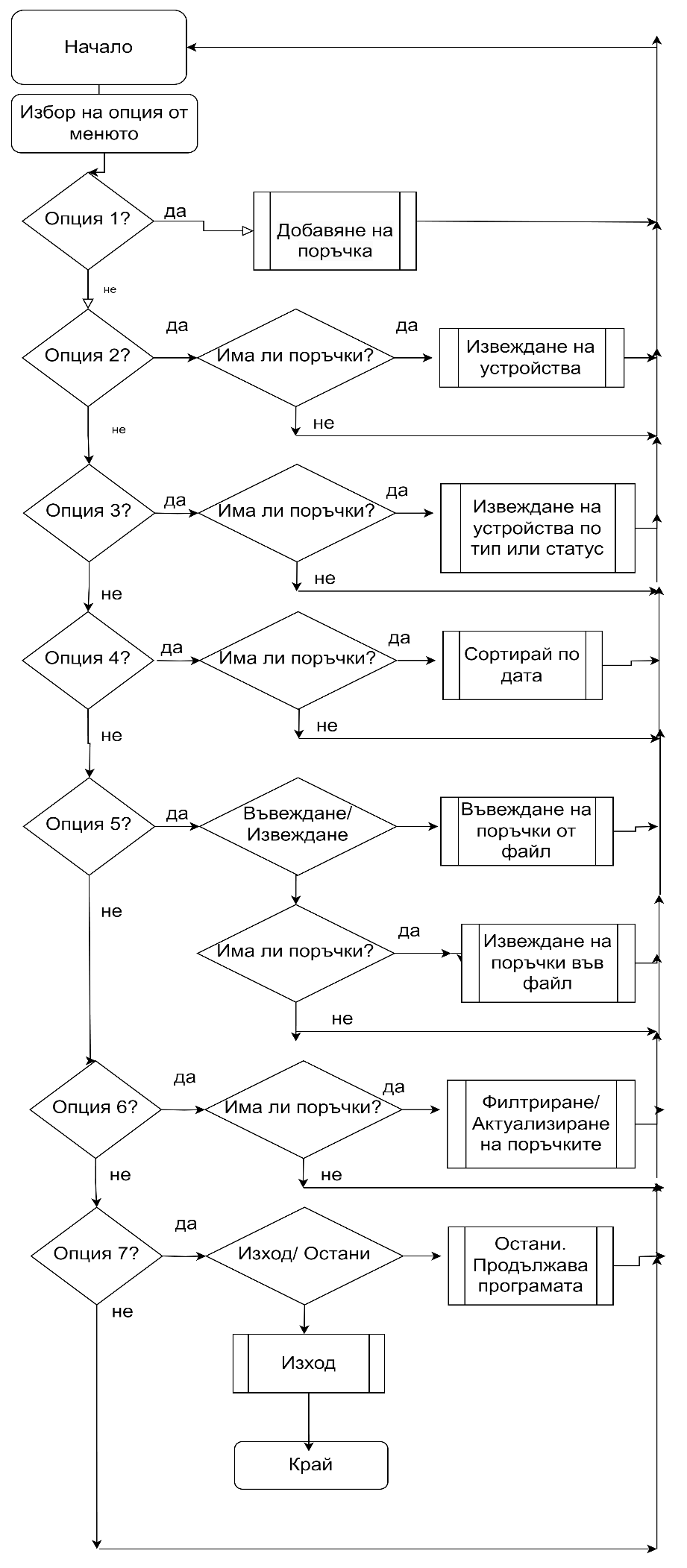
|  |  |  |
| --- | --- | --- |
| Структура | Обяснение | Примерни стойности |
| struct ServiceOrder  {  int number;  int dayOfMonth;  char client[CLIENT\_NAME\_LENGTH];  char typeOfTech[TECH\_TYPE\_LENGTH];  char techSerialNumber[SERIAL\_NUMBER\_LENGTH];  char problem[PROBLEM\_LENGTH];  char nameOfTechnician[TECHNICIAN\_NAME\_LENGTH] = "empty";  char renovation[RENOVATION\_LENGTH] = "empty";  double price = 0.0;  int daysInService = 0;  char status[STATUS\_SIZE] = "accepted";  char orderType[ORDER\_TYPE\_SIZE] = "empty"; | -number – номер на поръчка;  -dayOfMonth – ден от месеца  -client – име на клиента  -typeOfTech – тип устройство(телефон, лаптоп, телевизор и др.)  -techSerialNumber – сериен номер на устройството  -problem – проблем според клиента  -nameOfTechnician – име на сервизния техник  -renovation – извършен ремонт  -price – цена на ремонта  -daysInService - дни престой в сервиза  -status – статус на поръчка(приета, отказана, завършена)  -orderType – вид на поръчка(нормална – за 10 дена, бърза за 5 дена, експресна за 2 дена) | 1  8  Harry Josep  laptop  998-if  broken screen  Ivan Bren  replaced screen  20 lv  10  completed  normal |

Таблица 1. Структура

Реализация на условие A

Алгоритъм Меню с избор на функциите от програмата

Алгоритъмът за Меню с избор на функциите дава възможност на потребителя да избере измежду 7 опции от основното меню. Започва с визуализиране на седемте опции, след което идва проверка дали въведените входни данни от потребителя отговарят на типа на променливата, която се очаква. При преминаване на тази проверка се извикват функции спрямо избора на потребителя. При избор въведен от потребителят, който не съществува, се изписва съобщение за грешен избор. Алгоритъмът се повтаря докато не се избере опцията за изход и потребителя не потвърди решението си .

Блок схема на алгоритъма

Фиг. 1 Блок схема на Алгоритъм Меню с избор на функциите

Функция, с която е реализиран алгоритъма

int main()

#### Входни данни на функцията

Няма входни данни.

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

Главната функция съставя цикъл, чрез който при избор на потребителя се извиква конкретна функция и цикъла се повтаря. След край на програмата, при успешно изпълнение, функцията връща код 0.

Реализация на условие B

Алгоритъм Добавяне на нова сервизна поръчка/ поръчки в системата.

При избор 1 от потребителя се извиква функцията за добавяне на сервизна поръчка. Появява се запитване какъв да е броя на поръчките, които да се добавят. Проверява се типа на входните данни, след това дали стойността е по-малка от 1 или по-голяма от 100. След това се проверява дали сумата на текущия брой поръчки с тази на добавените поръчки надвишава допустимия лимит от поръчки. При успешно преминаване на всяка една проверка, започват добавянето на всяка една поръчка една по една. И броят им се покачва съобразно броят на добавените поръчки.

Блок схема на алгоритъма



Фиг. 3.. Блок схема на Алгоритъм въвеждане на данни за сервизна поръчка

Функция, с която е реализиран алгоритъм

void addOrder(const ServiceOrder orders[], int& countOfExistingOrders );

#### Входни данни на функцията

ServiceOrder orders[] – масив от тип структура ServiceOrder, в който се съхраняват данните за всички сервизни поръчки;

int& countOfExistingOrders – променлива от тип int, която е подадена по адрес, която държи информация за броя на съдържащите се сервизни поръчки

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

Няма. Функцията връща данните за сервизните поръчки, през параметъра ServiceOrder orders[]. Промяната на стойността в променливата int& countOfExistingOrders се случа по адрес.

Случайно генерирани данни

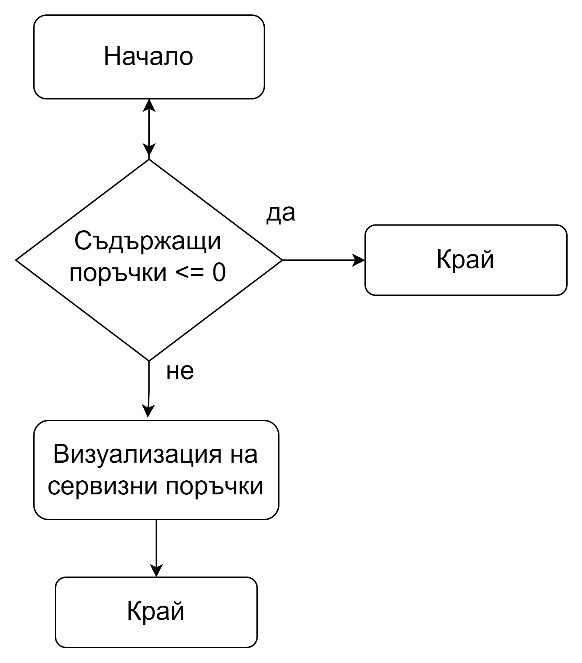
Статуса на поръчка първоначално винаги е (приета), проблема се генерира на случаен принцип зависи от типа устройство, което се въведе. Ако се въведе телефон, може всеки един проблем да се генерира без последния, защото последния проблем не съвпада с това устройство. Ако се въведе телевизор може да се генерира един от първите пет проблема във вече декларирания масив, защото тези проблеми са по-конкретни за телевизор. А при друго устройство като например лаптоп или малка конзола(nintendo), би се генерирал един случаен проблем от всички без изключение.

Реализация на условие C

Алгоритъм Извеждане всички данни на сервизна поръчка

При избор на опция 2 се визуализират всички налични сервизни поръчки и пълната информация за тях, оформена в табличен вид. Ако няма текущи сервизни поръчки, то ще се изпише надпис, че няма налични поръчки.

Блок схема на алгоритъма



Фиг. 4. Алгоритъм Извеждане на сервизни поръчки

Входни данни на функцията

void displayAllDevices (ServiceOrder orders[],int countOfExistingOrders);

#### Входни данни на функцията

ServiceOrder orders[]- списък със сервизните поръчки

int countOfExistingOrders – брой текущи сервизни поръчки

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

Няма.

Помощни функции

void displayHeadInformation () – отпечатва заглавният ред на таблицата

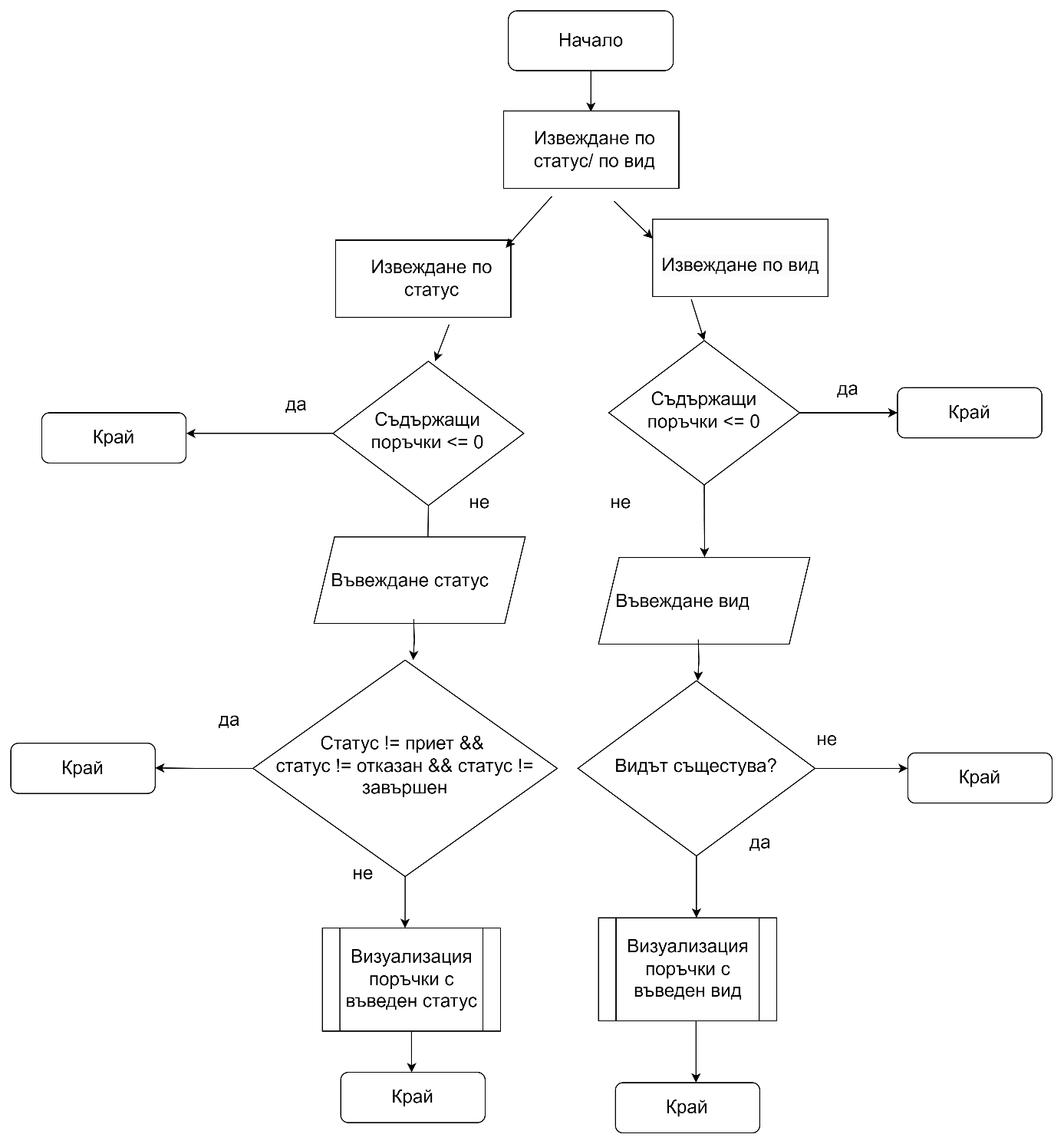
void displayCurrentOrder () – отпечатва текущ ред с данни на една сервизна поръчка

Реализация на условие D

Алгоритъм Извеждане на устройства по вид или по статус

При избор 3 от потребителя ,той получава възможността да избере измежду две опции. Опция А извежда устройствата по вид въведен от потребителя. Опция Б извежда устройства по статус на сервизната поръчка. Цикъл минава през масива до бройката на текущите поръчки и чрез проверка визуализира само желаните от потребителя поръчки. Ако не се срещне желания вид, се визуализира съобщение, че такъв вид не съществува. Ако въведения статус не е сред текущите поръчки, ще се изведе съобщение, че няма поръчки с такъв статус.

#### Блок схема на алгоритъма



Фиг. 5. Блок схема на Алгоритъм Извеждане на сервизни поръчки по вид или статус

Функции, които реализират алгоритъма

void searchAndDisplayByType (ServiceOrder orders [],int countOfExistingOrders);

#### Входни данни на функцията

ServiceOrder orders[]- списък със сервизните поръчки

int countOfExistingOrders – брой текущи сервизни поръчки

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

Няма изходни данни. Функцията извежда устройства, които са от желаната вид или статус.

#### Помощни функции

void lowerCharArray (char arr[], int size) – преобразува целия масив от символи в малки букви

void displayHeadInformation () – отпечатва заглавният ред на таблицата

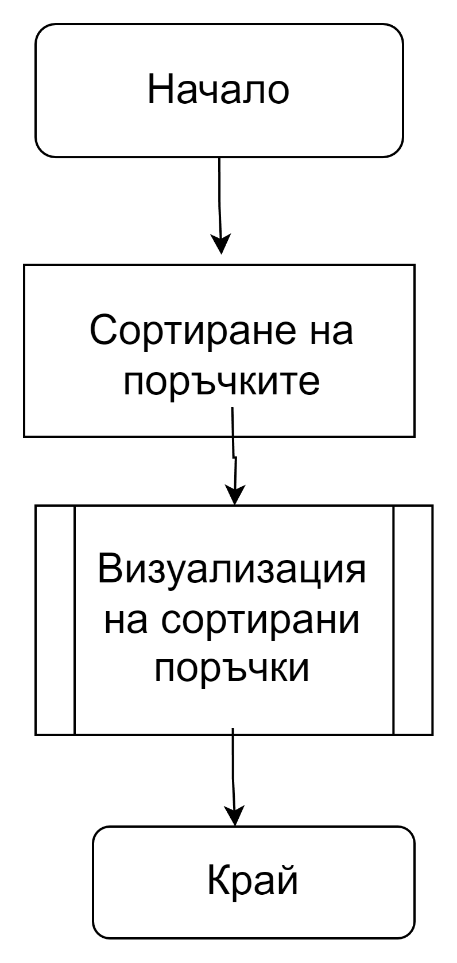
void displayCurrentOrder (ServiceOrder orders [],int countOfExistingOrders) – отпечатва текущ ред с данни на една сервизна поръчка

Реализация на условие E

Алгоритъм Сортиране на студенти по малко име по азбучен ред

При избор на опция 4 от потребителя, сервизните поръчки се сортират по дата на приемане с помощта на bubble sort. При успешно преминаване на сортировката се визуализира надпис, че поръчките са подредени.

Блок схема на алгоритъма



Фиг. 6. Алгоритъм Сортиране на сервизни поръчки по дата на приемане

Функция с която е реализиран алгоритъма

void sortOrdersByDatetime (ServiceOrder orders [], int countOfExistingOrders) – сортира поръчките по дата на издаване възходящо

#### Входни данни на функцията

ServiceOrder orders[]- списък със сервизните поръчки

int countOfExistingOrders – брой текущи сервизни поръчки

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

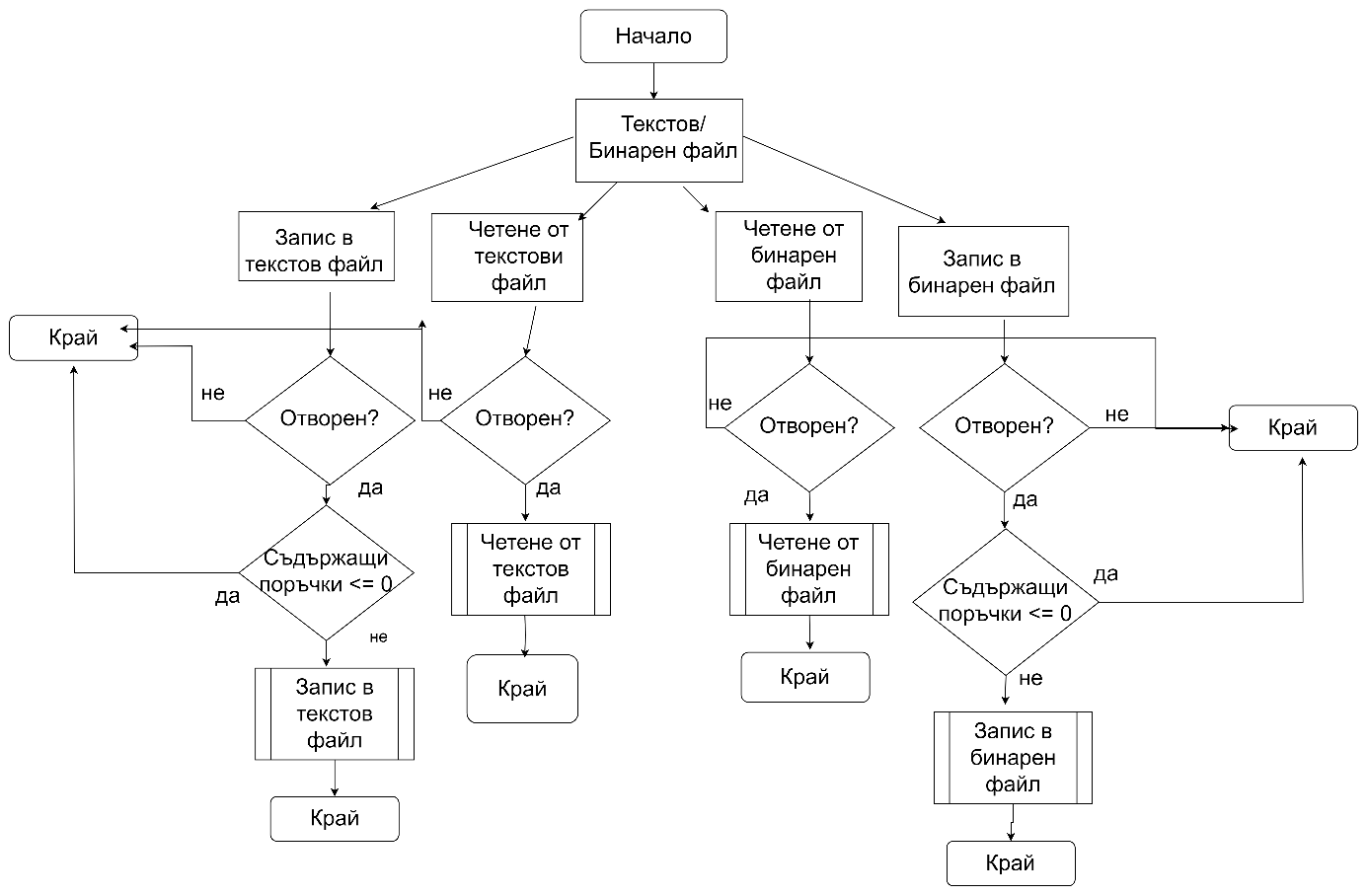
Няма изходни данни. Функцията сортира масива със сервизни поръчки по дата на приемане.

Реализация на условие F

Алгоритъм Извеждане на масива със сервизни поръчки във файл или въвеждане на сервизни поръчки от файл (текстови, бинарен)

При избор на опция 5 от потребителя, има възможност да се въведат данни във файл или да се изведат от файл. Потребителят има възможност за работа с информация както с текстови, така и с бинарен файл.

Блок схема на алгоритъма



Функции, с които е реализиран алгоритъма

void importFromTextFile (ServiceOrder orders [],int& countOfExistingOrders) – чете данни от текстов файл

void exportToTextFile (ServiceOrder orders [],int& countOfExistingOrders) – записва данни в текстов файл

void importFromBinaryFile (ServiceOrder orders [],int& countOfExistingOrders) – чете данни от бинарен файл

void exportToBinaryFile (ServiceOrder orders [],int& countOfExistingOrders) – записва данни от бинарен файл

#### Входни данни на функцията

ServiceOrder orders[] – масив от тип структура ServiceOrder, в който се съхраняват данните за всички сервизни поръчки;

int& countOfExistingOrders – променлива от тип int, която е подадена по адрес, която държи информация за броя на съдържащите се сервизни поръчки

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

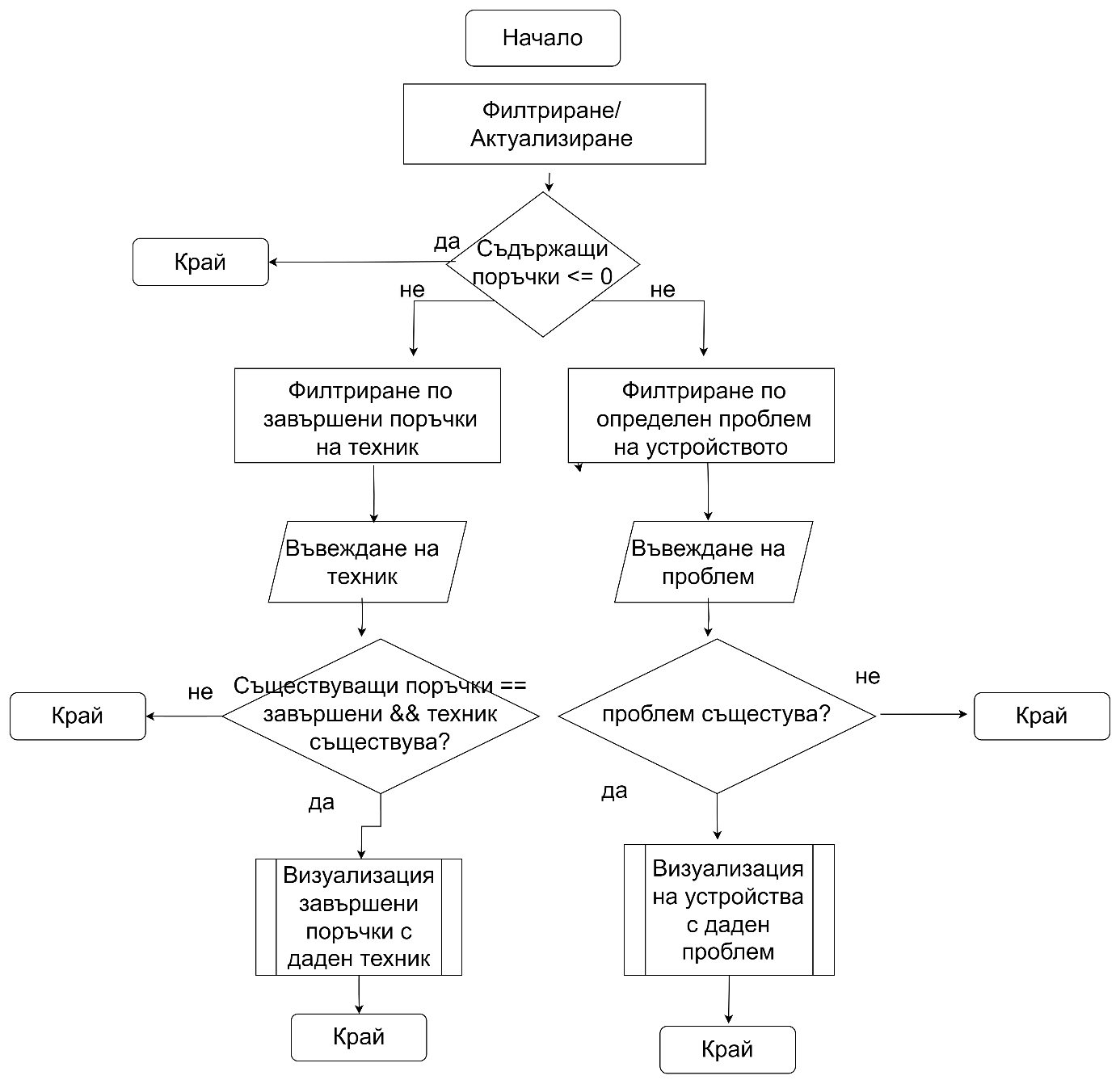
Съобщение за грешка се изписва при неуспешно отваряне на файла и съобщение за успех при успешно записване или четене от файл.

Реализация на условие … - допълнение първо (+ базова задача)

Анализ на алгоритъма, който трябва да се реализира

При избор на опция 6 от потребителя се появява подменю с опции А и Б. Филтриране или актуализиране и добавяне на информация. При избор А се появяват нови две възможности. Опция 1 е филтрация на всички завършени поръчки при въведен от потребителя техник. Опция 2 е филтрация на всички устройства с проблем въведен от потребителя. Ако техникът не съществува или просто няма завършени поръчки ще се покаже надпис, че този техник няма завършени поръчки. Ако проблемът не съществува се изписва, че няма поръчки с такъв проблем.

Блок схема на алгоритъма



Функции, с които е реализиран алгоритъма

void filterOptions (ServiceOrder orders [],int countOfExistingOrders) – показва вътрешното меню с опции за филтриране

void printTechnicianSortedOrders (ServiceOrder orders [],int countOfExistingOrders) – филтрира и визуализира сервизните поръчки по завършени и по даден техник като сортира по дата на приемане

void printOrdersWithGivenProblem (ServiceOrder orders [],int countOfExistingOrders) – филтрира и визуализира устройствата с даден проблем, като ги сортира по тип на устройстовото в низходящ ред

#### Входни данни на функцията

ServiceOrder orders[]- списък със сервизните поръчки

int countOfExistingOrders – брой текущи сервизни поръчки

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

Функциите не извеждат изходни данни. Те само филтрират и показват на екрана данни.

#### 8.4 Допълнителни функции

void lowerCharArray (char arr[], int size) – преобразува целия масив от символи в малки букви

void sortOrdersByDatetime (ServiceOrder orders [],int countOfExistingOrders) – сортира поръчките по дата на издаване възходящо

void displayHeadInformation () – отпечатва заглавният ред на таблицата

void displayCurrentOrder (ServiceOrder orders [],int countOfExistingOrders) – отпечатва текущ ред с данни на една сервизна поръчка

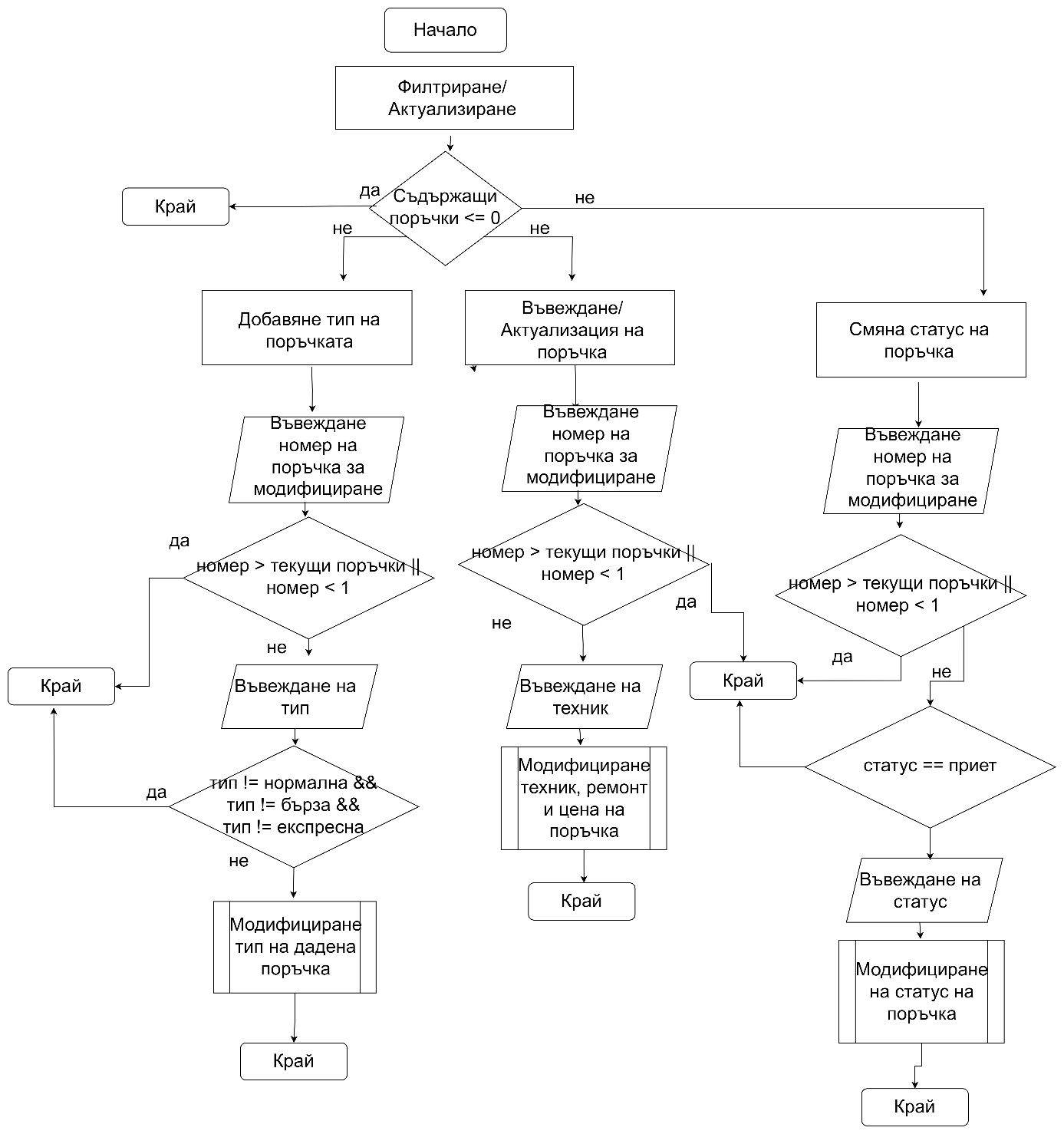
void sortOrdersByTechTypeDescending (ServiceOrder orders [],int countOfExistingOrders) – сортира по тип устройство низходящ ред

Реализация на условие … - допълнение второ (+ базова задача)

Анализ на алгоритъма, който трябва да се реализира

При избор на опция 6 от потребителя се появява подменю с още две опции. Първата опция с филтрирането беше разгледана. Втората опция Б е с добавяне и актуализиране на информация. Появяват се нови три опции. Опция номер едно позволява на потребителя да добави тип на сервизната поръчка, който тип преди това е без стойност означен с „empty”. След модифицирането на типа поръчка, потребителя има възможност вече да добави информация за поръчката като избере опция 2 само при условие, че статуса на поръчката е „приета“. Потребителят ще бъде изискван да въведе име на техник, което ще попълни автоматично поправката на случайно генерирания проблем, ще съобрази дните за престой спрямо типа на сервизната поръчка, който може да бъде (нормален, бърз и експресен) и ще изчисли цената на поправката. Опция номер 3 е възможна само и единствено, когато статуса на поръчката е приета. При отказана и завършена статусът на поръчката не може да се редактира.

Блок схема на алгоритъма

Функции, с които е реализиран алгоритъма

void addOrderType (ServiceOrder orders [],int countOfExistingOrders) – добавя тип на сервизната поръчка

void insertUpdateData (ServiceOrder orders [],int countOfExistingOrders) – добавя/ актуализира данни на сервизна поръчка

void changeStatusOfOrder (ServiceOrder orders [],int countOfExistingOrders) – променя статуса на поръчка само единствено, ако е статуса й е „приета“

#### Входни данни на функцията

ServiceOrder orders[]- списък със сервизните поръчки

int countOfExistingOrders – брой текущи сервизни поръчки

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

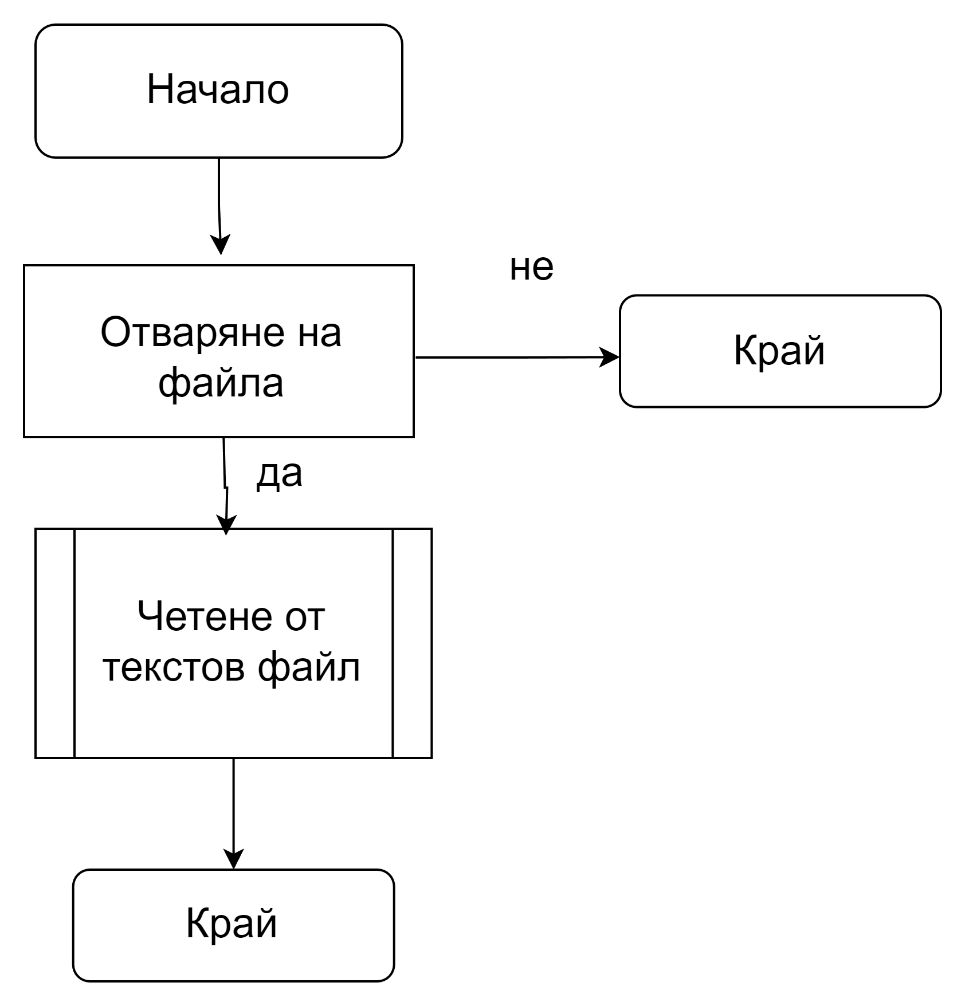
След успешно изпълнение на всяка една от трите функции, се изписва текст, какво се е променило успешно.

Реализация на допълнение трето (+ базова задача)

Анализ на алгоритъма, който трябва да се реализира

Със стартиране на програмата се извиква функцията за четене от текстов файл .

Блок схема на алгоритъма



Функция с която е реализиран алгоритъма

void importFromTextFile (ServiceOrder orders [],int& countOfExistingOrders)

#### Входни данни на функцията

ServiceOrder orders[]- списък със сервизните поръчки

int countOfExistingOrders – брой текущи сервизни поръчки

Изходни данни на функцията или данни, които се извеждат

Изписва се текст, че файлът е прочетен успешно, ако се е отворил и че е неуспешно, когато не е успял да се отвори.

Примерно действие на програмата

#Условие A

Снимка на изгледа с примерни входни данни

Входни данни

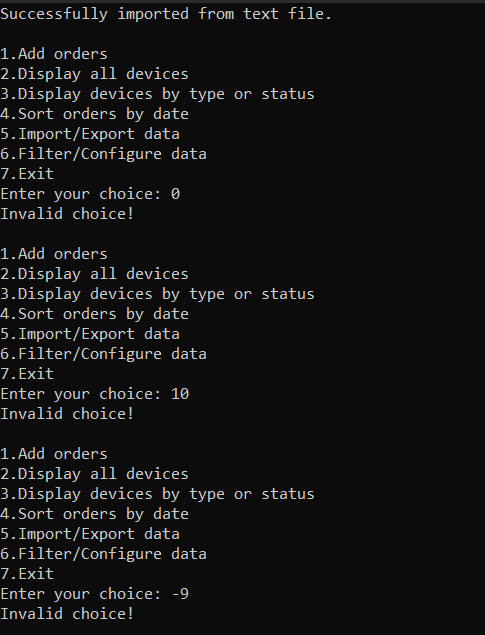
Въвеждане на опция от главното меню. При въвеждане на опция са възможни следните ситуации:

А) Въвеждане на опция < 1 – в този случай, ще се покаже грешка „Невалиден избор“.

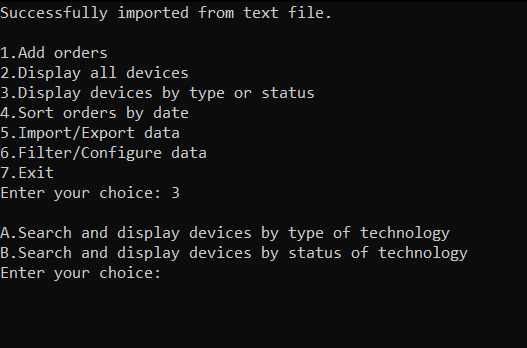
Б) Въвеждане на опция > последният номер на опция – в този случай ,ще се покаже грешка „Невалиден избор“.

* В) Въвеждане на опция в интервала [1 – предпоследната опция] – в този случай програмата изпълнява действията, заложени за изпълнение при дадената опция
* Г) Въвеждане на последната опция – в този случай това е опция „Изход“, което води до прекратяване на изпълнението на цялата програма.

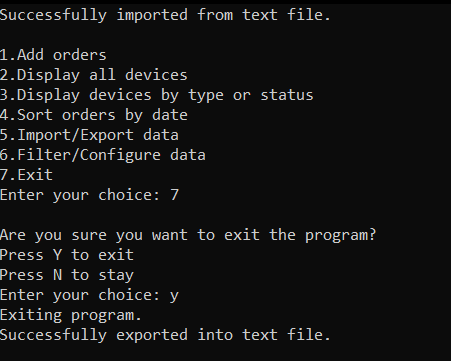
Снимка на изгледа с примерни изходни данни



Фиг. 7. Въвеждане на невалидна опция - Случай А) и Б)



Фиг. 8. Въвеждане на валидна опция на меню В)

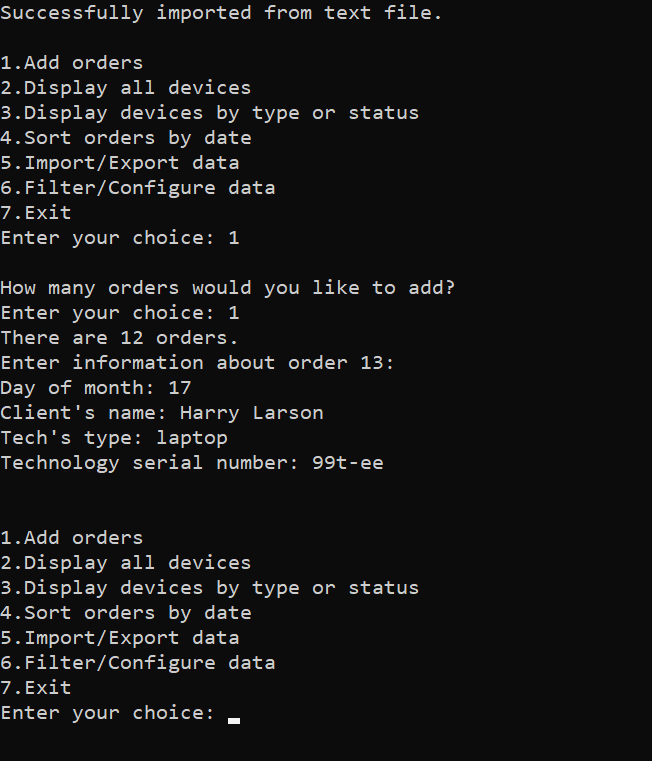


Фиг. 9 Въвеждане на номера на последната опция – Изход – Случай Г)

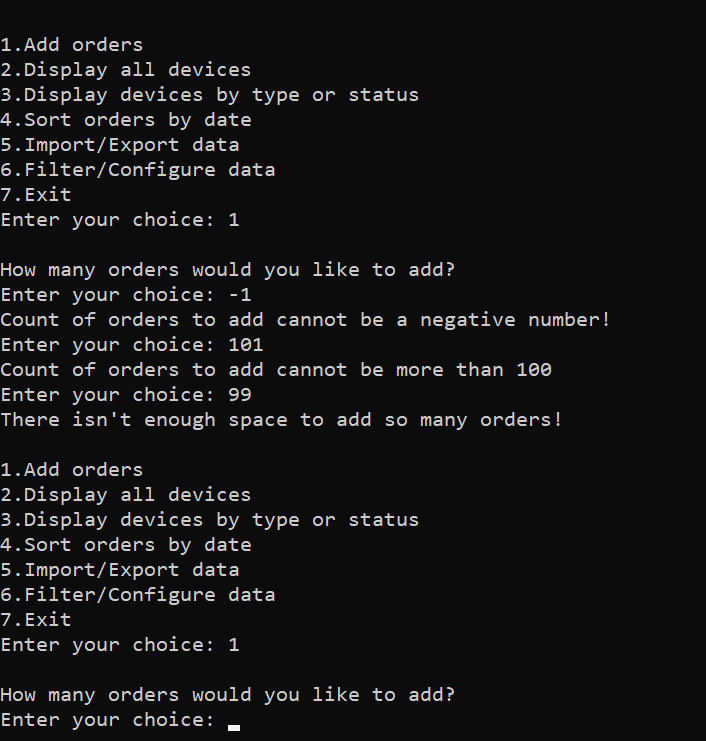
Условие B

Примерни входни данни

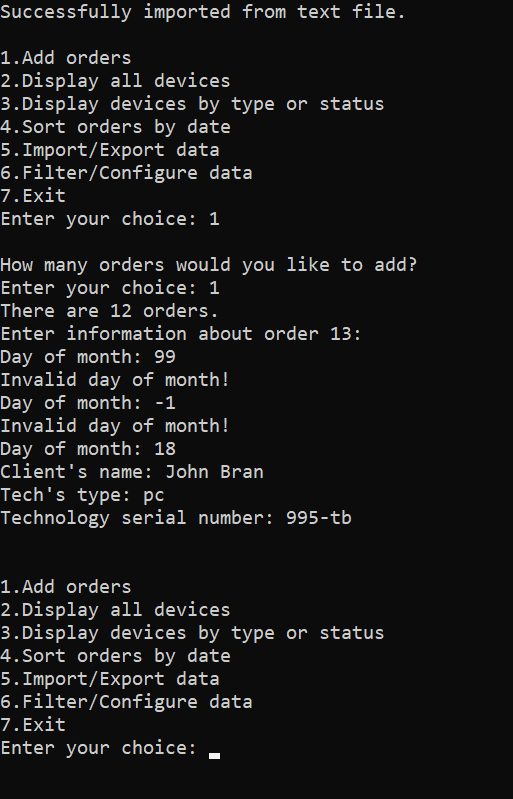
Въвежда се броят на поръчките, които иска потребителя да се добавят. При въвеждането се спазват изискванията за въвеждане на данни спрямо типовете им! Предвидена е валидация, ако типът се сгреши, броят е по-голям от наличния, броят е отрицателен и ако датата е извън рамките (1-31). При въвеждане на стойности извън допустимите се очаква ново въвеждане с коректна стойност.



Фиг. 10. Въвеждане коректни данни на поръчка



Фиг. 11. Въвеждане некоректни данни на поръчка



Фиг. 12. Въвеждане некоректни данни на поръчка

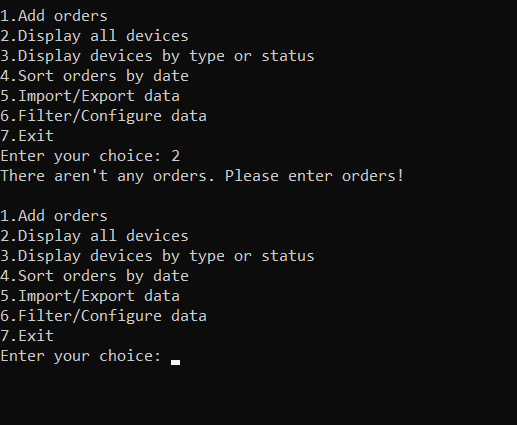
Снимка на изгледа с примерни данни

Условие C

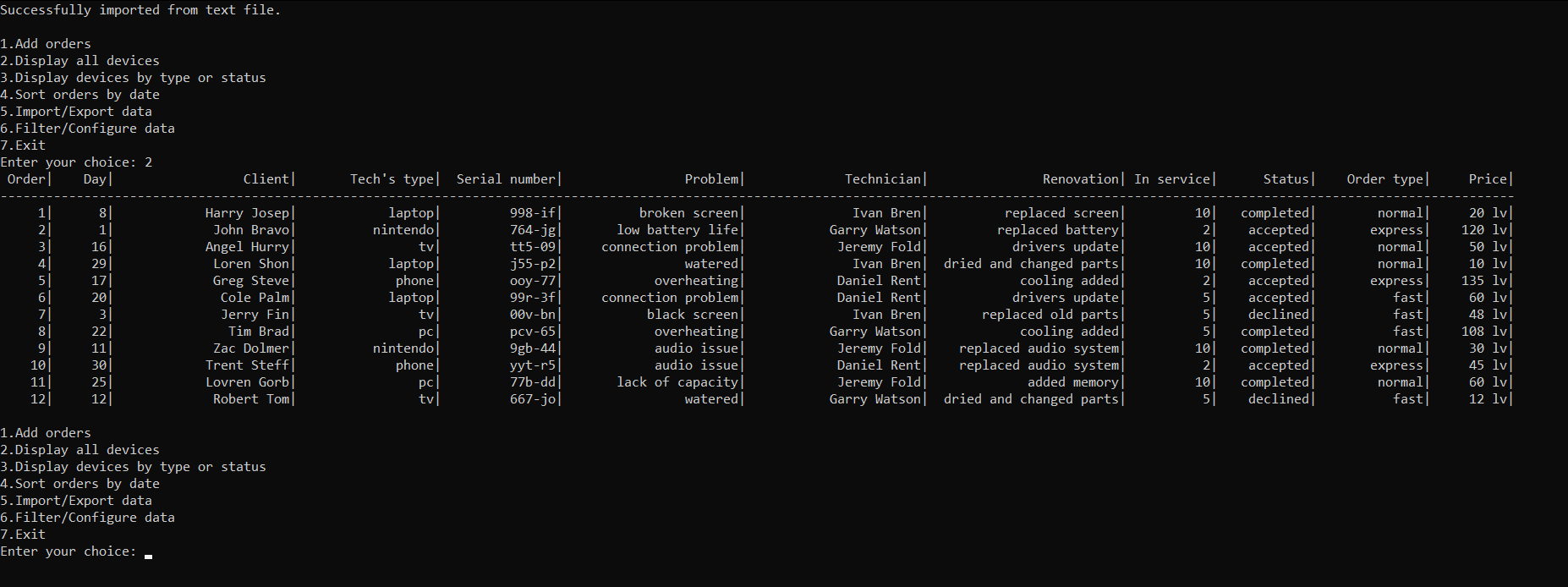
Примерни входни данни

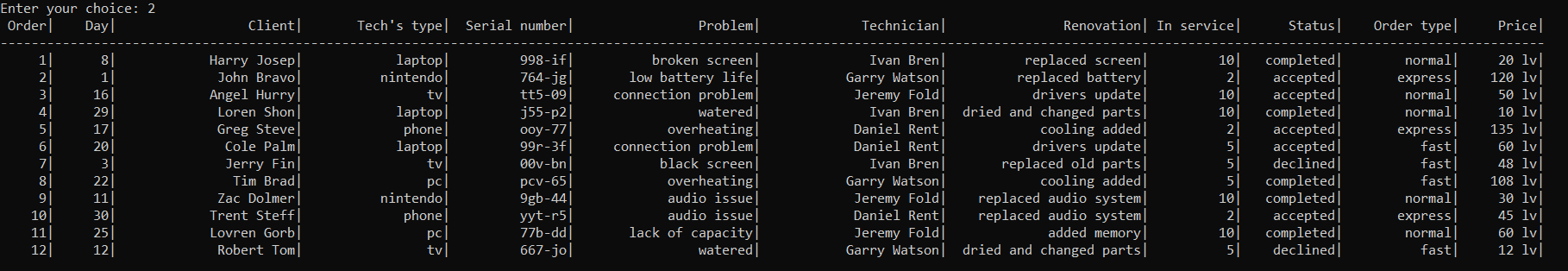
Извеждането на студенти на екрана (отпечатване) не изисква въвеждане на входни данни. Предвидена е проверка за празен масив (списък) със студенти и извеждане на подходящо съобщение. При наличие на стойности се отпечатват студентите в таблица.

Снимка на изгледа с примерни данни



Фиг. 13. Извеждане на празен списък със студенти.





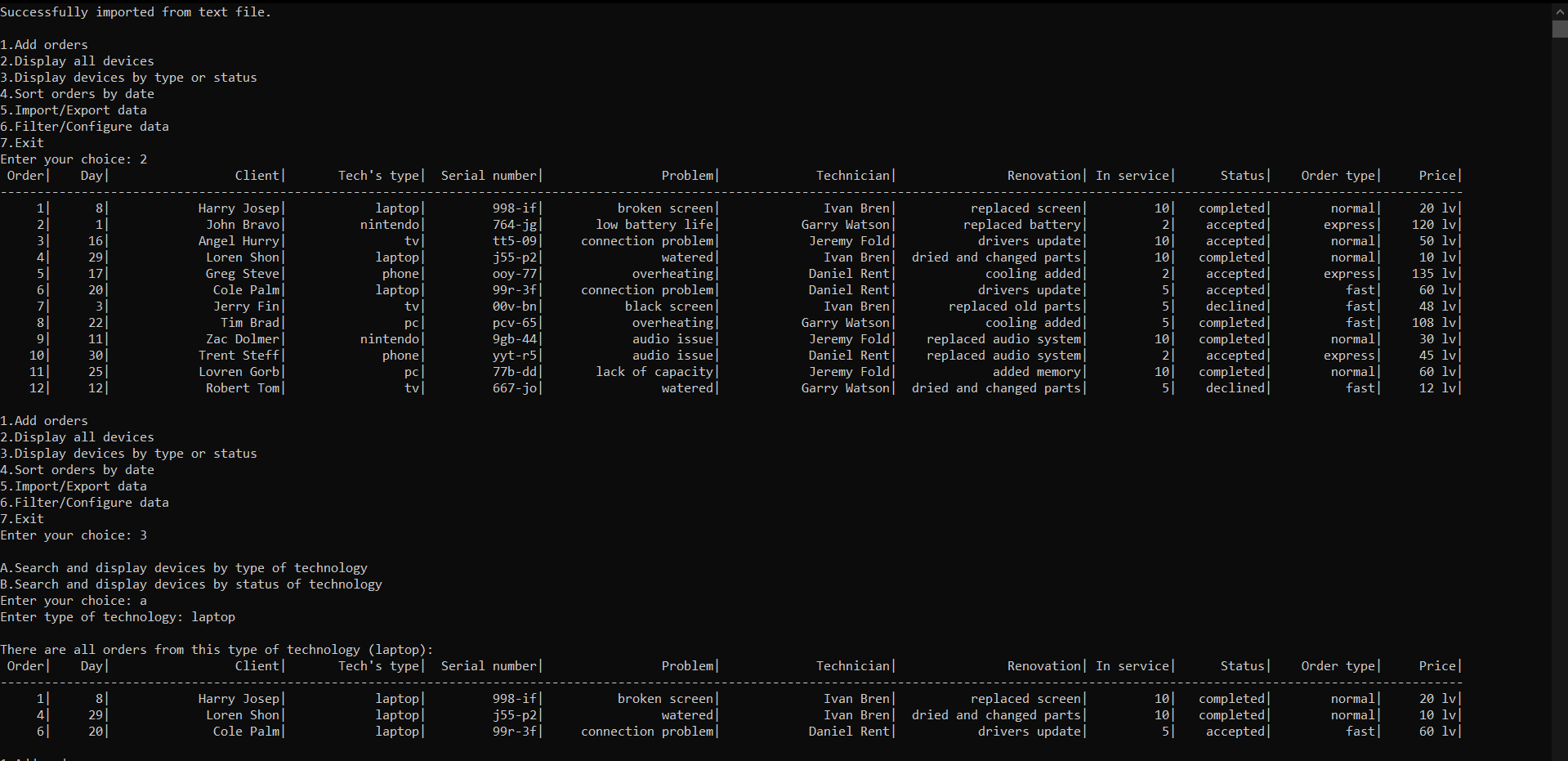
Фиг. 14. Извеждане на списък със студенти в таблица.

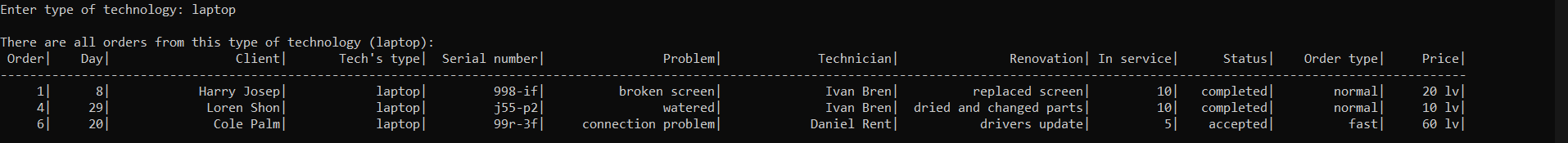
Условие D

Примерни входни данни

При избор за извеждане на устройствата по вид или по статус, има проверки при грешни входни данни като погрешен тип, невалиден избор, както и невалиден вид или статус.

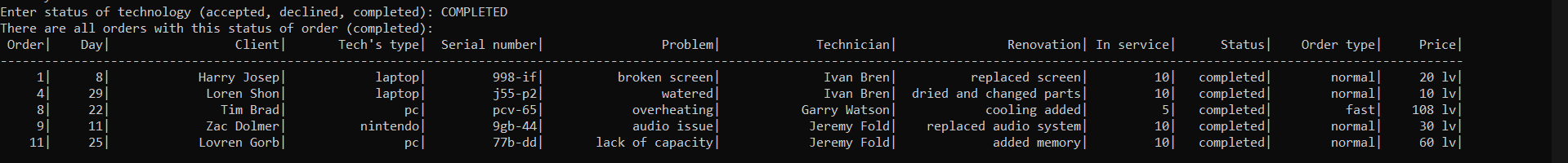
Снимка на изгледа с примерни данни



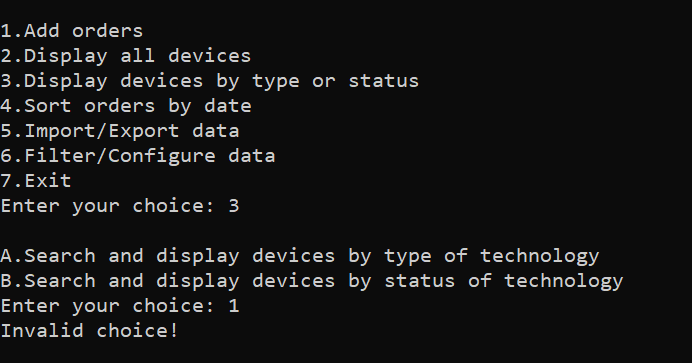


Фиг. 15. Търсене и извеждане на устройства по вид

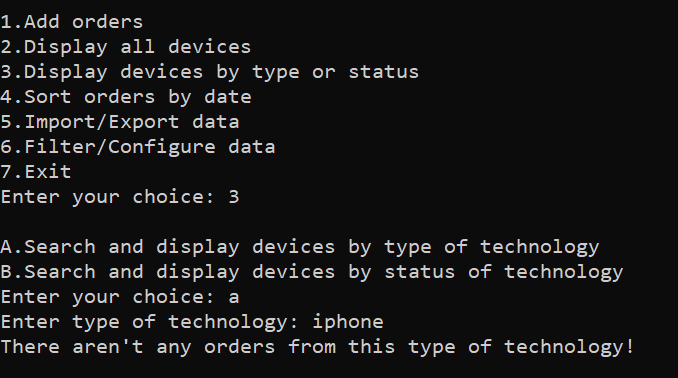




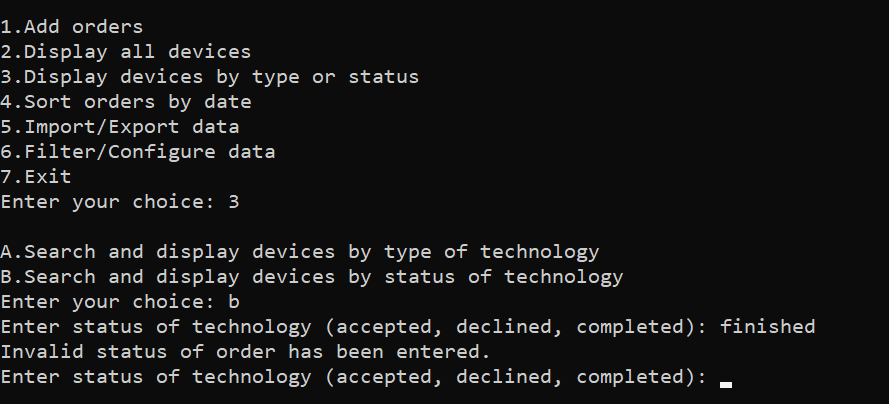
Фиг. 16. Търсене и извеждане на устройства по статус



Фиг. 17. Невалиден избор



Фиг. 18. Невалиден вид

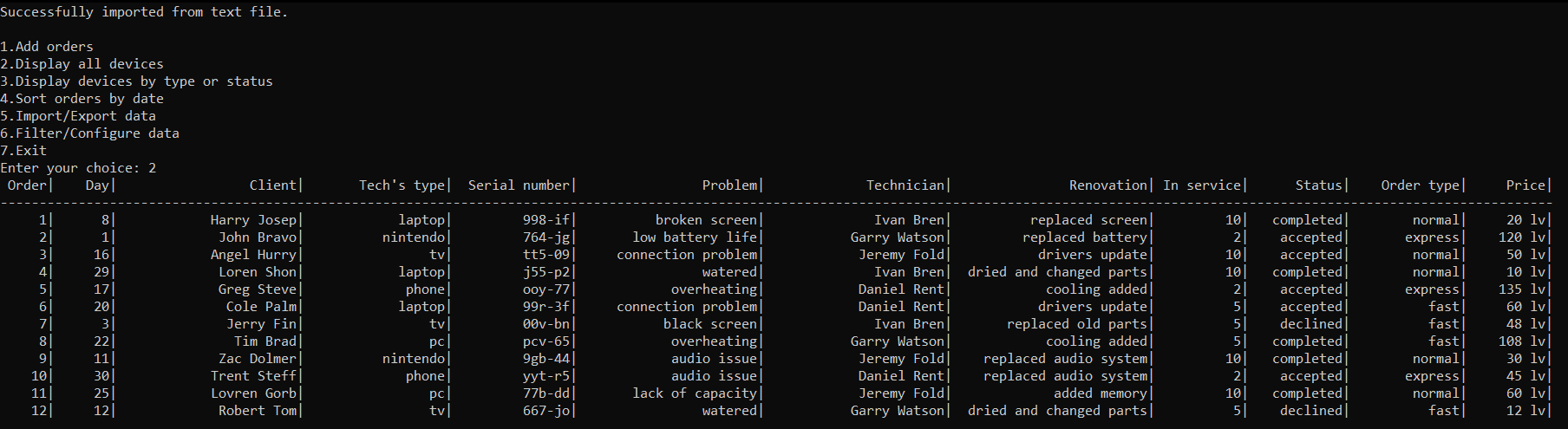


Фиг. 19. Невалиден статус

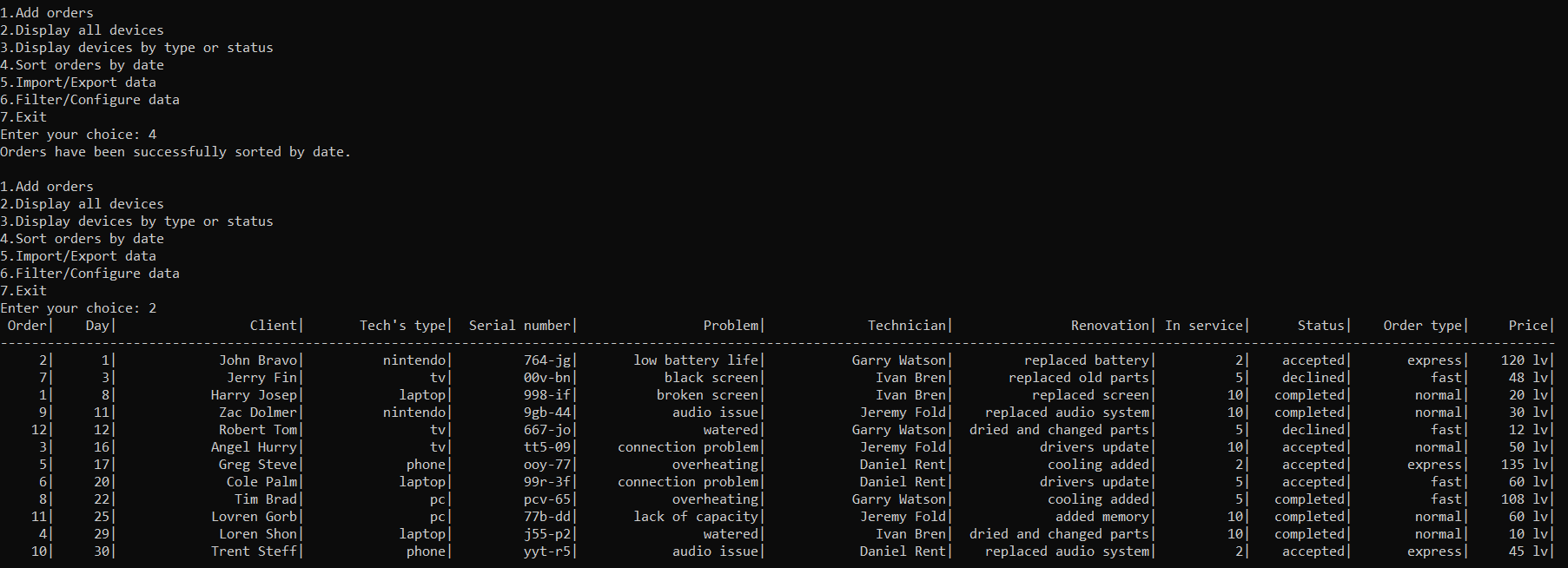
Условие E

Примерни входни данни.

При избор 4 потребителят има възможност да сортира данните по дата на приемане. Ако няма поръчки, то ще изпише съобщение.



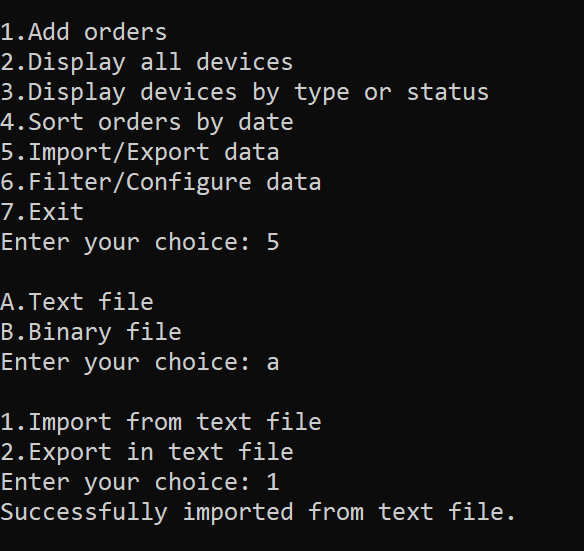
Фиг. 20. Всички данни



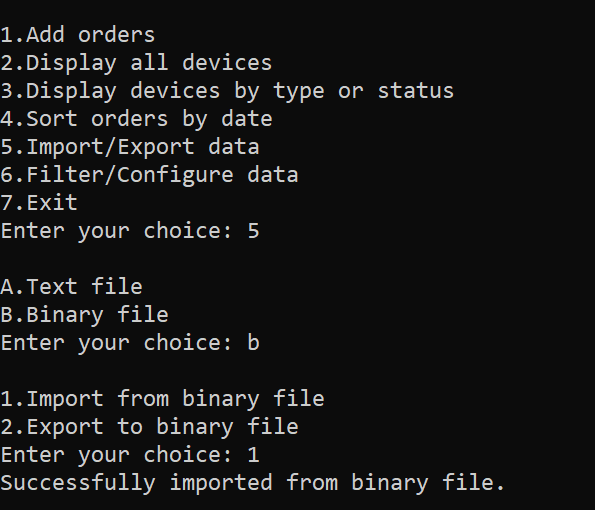
Фиг. 21. Сортирани по дата на издаване

Условие F

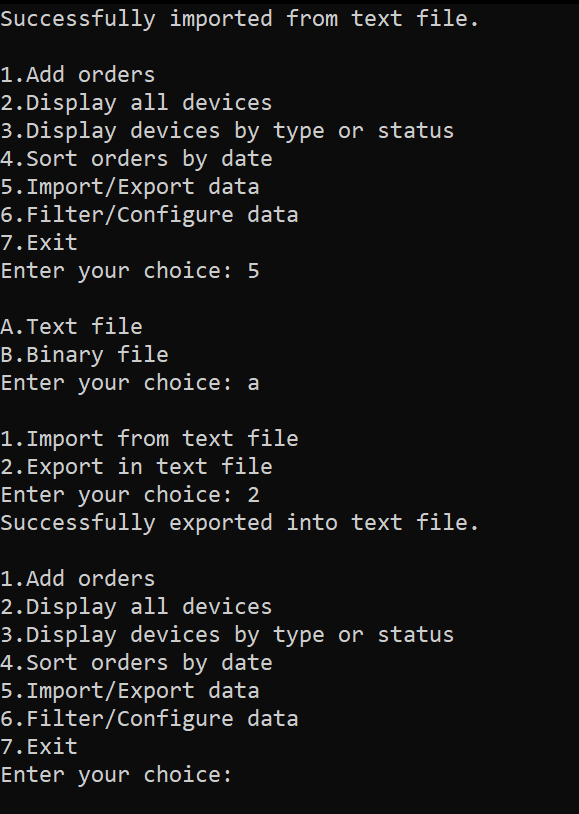
При избор 5 потребителят има възможност да чете данни от текстов файл и бинарен файл, както и да записва в текстов файл и бинарен файл. Ако няма поръчки изписва грешка.



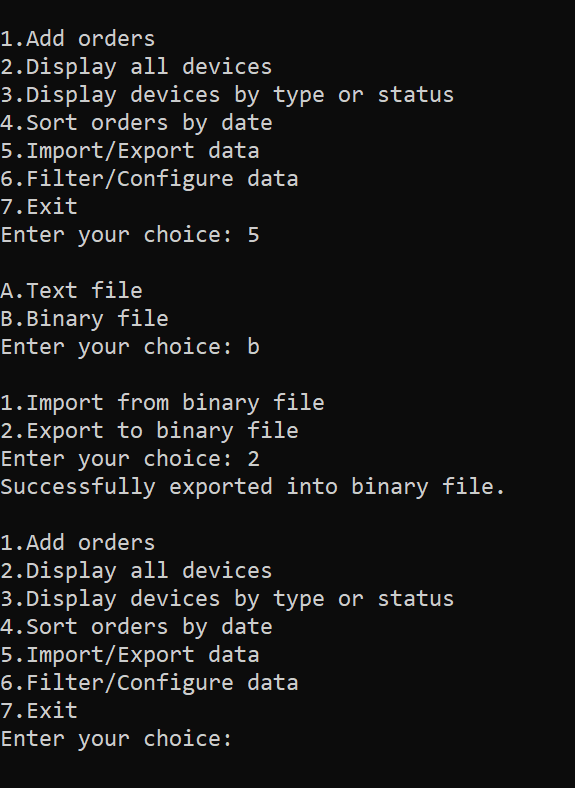
Фиг. 22. Успешно четене от текстов файл



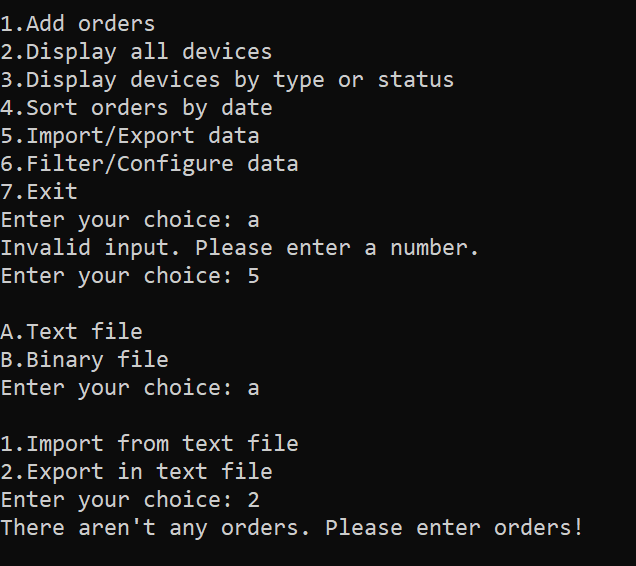
Фиг. 23. Успешно четене от бинарен файл



Фиг. 24. Успешен запис в текстов файл



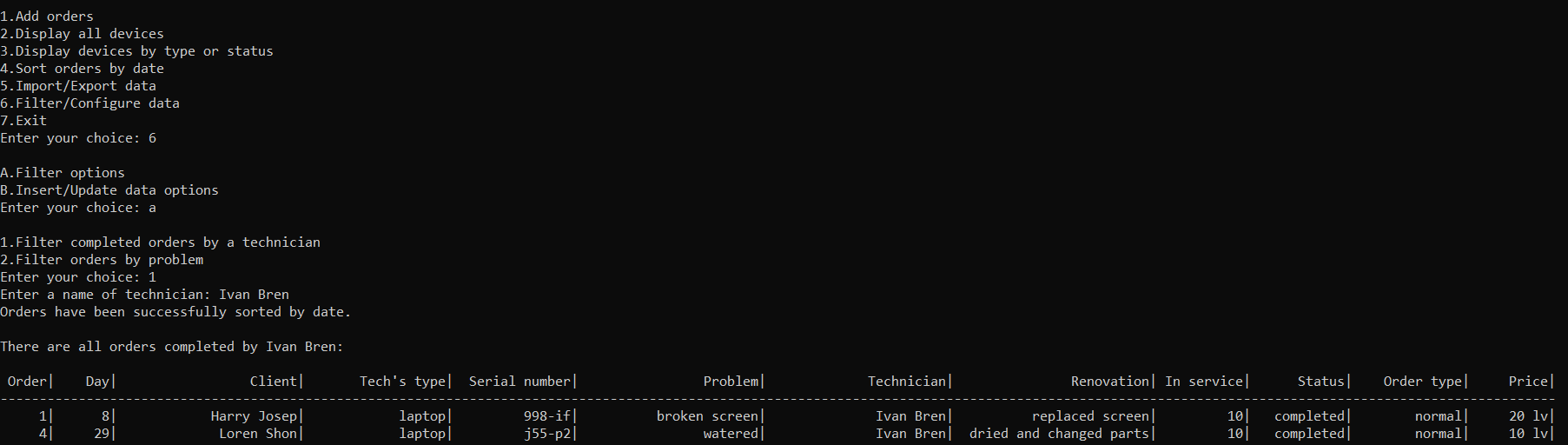
Фиг. 25. Успешен запис в бинарен файл



Фиг. 26.Неуспешен запис в текстов файл и грешен тип входни данни

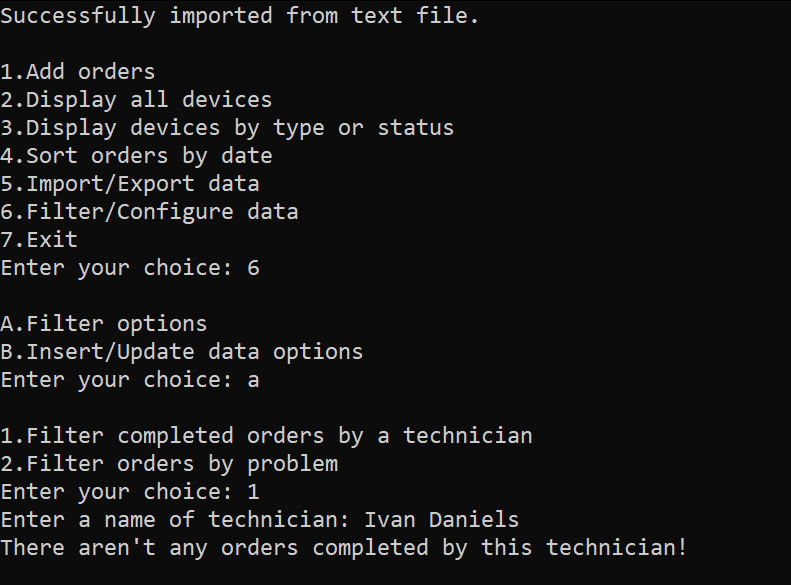
Допълнение първо

Снимка на изгледа с примерни входни данни



Фиг. 27.Успешно филтриране по завършени поръчки на даден техник

Снимка на изгледа с примерни изходни данни

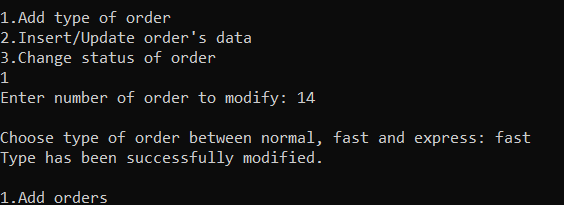


Фиг. 28.Неуспешено филтриране по завършени поръчки на даден техник. (Такъв техник не съществува)

Допълнение второ

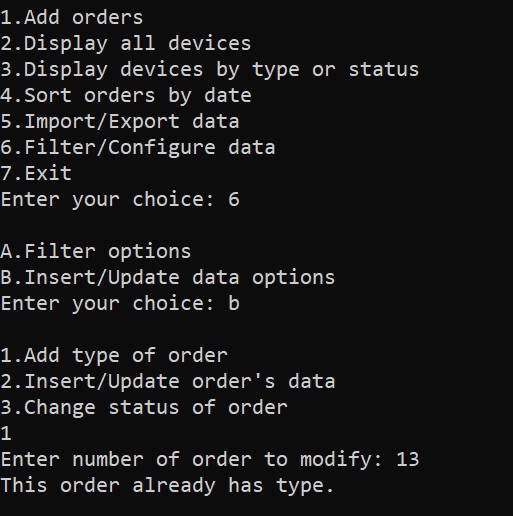
Снимка на изгледа с примерни входни данни





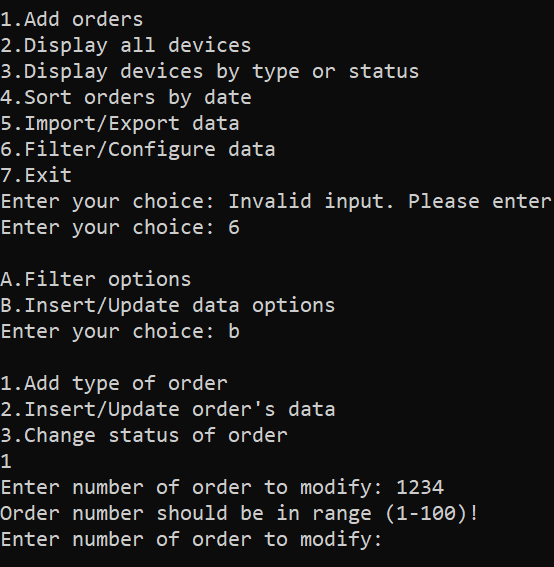
Фиг. 29.Успешно модифициране на типа на поръчката

Снимка на изгледа с примерни изходни данни



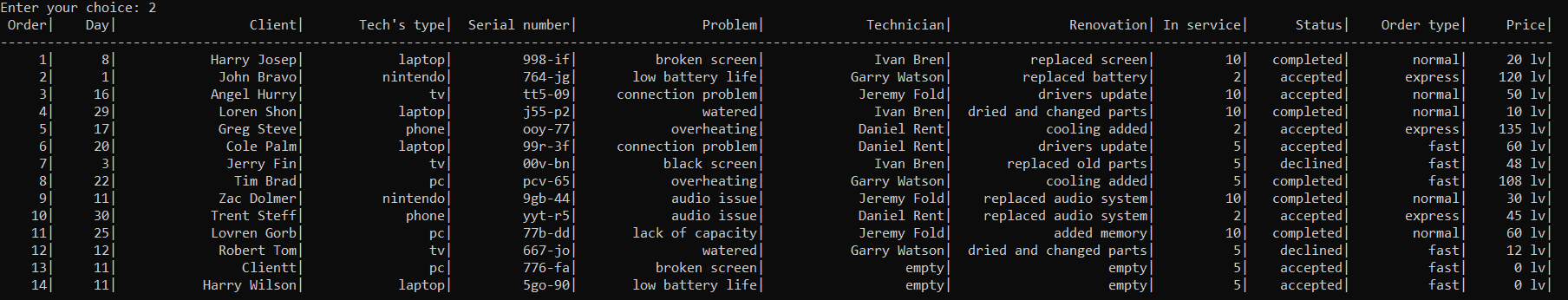
Фиг. 30.Неуспешено модифициране на типа на поръчката, защото вече си има тип

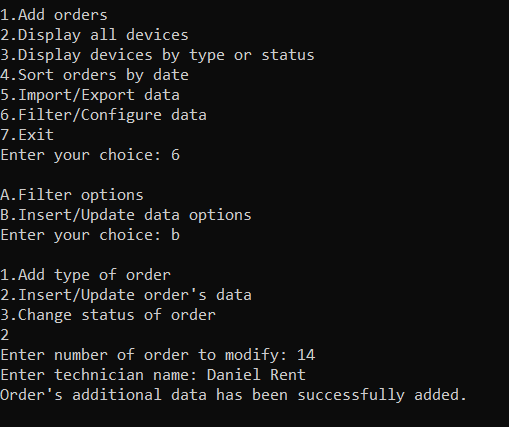
Снимка на изгледа с примерни изходни данни

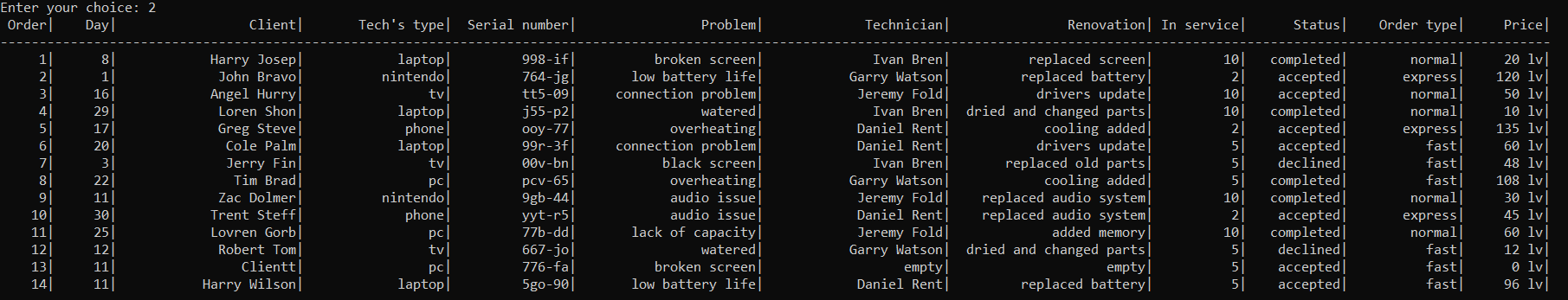


Фиг. 31.Некоректни входни данни

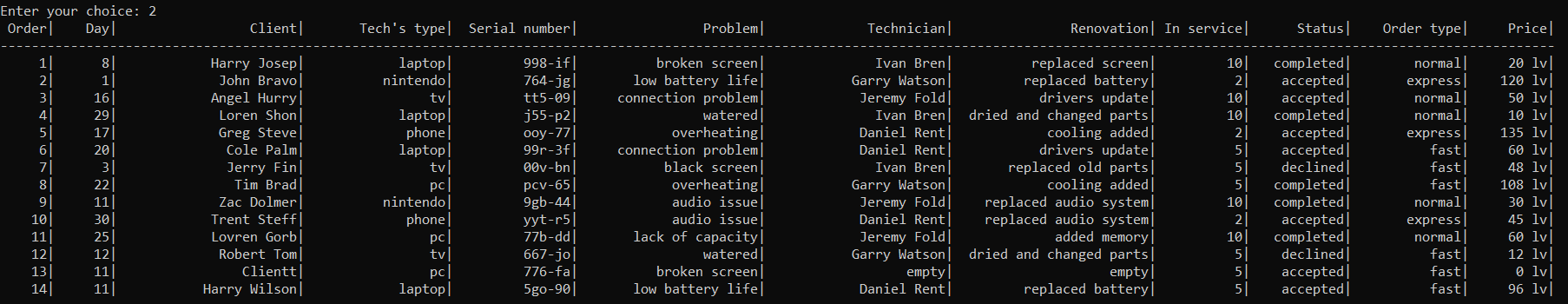
Снимка на изгледа с примерни входни данни

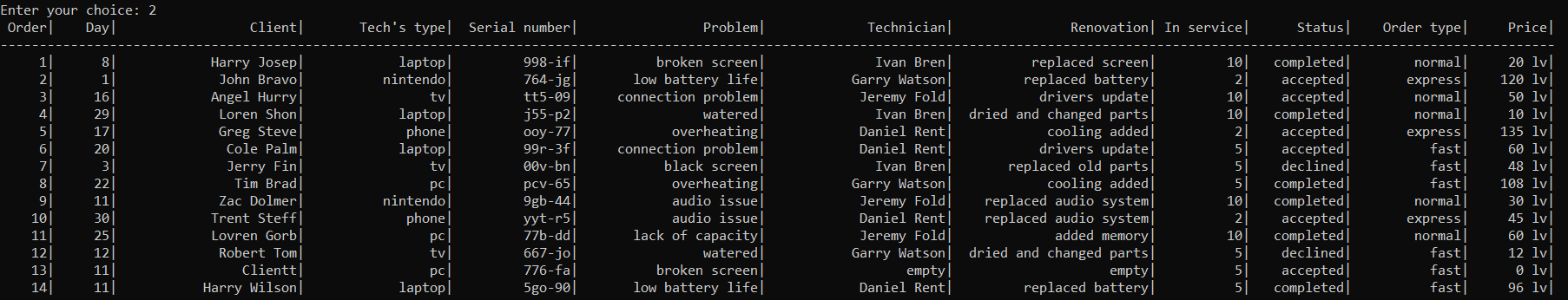
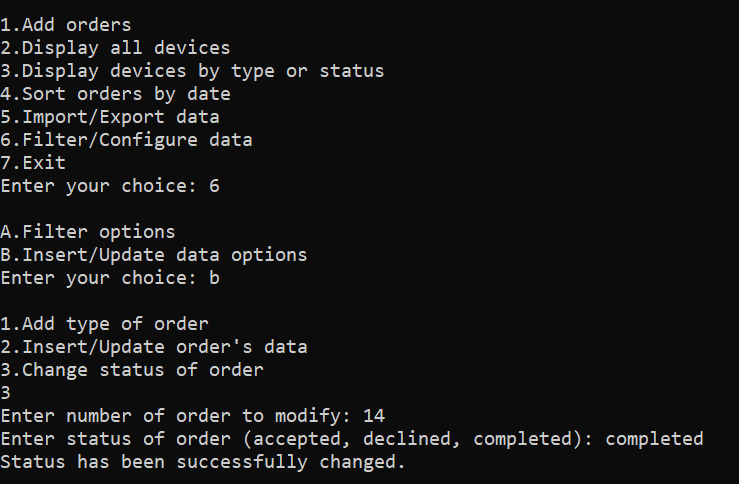






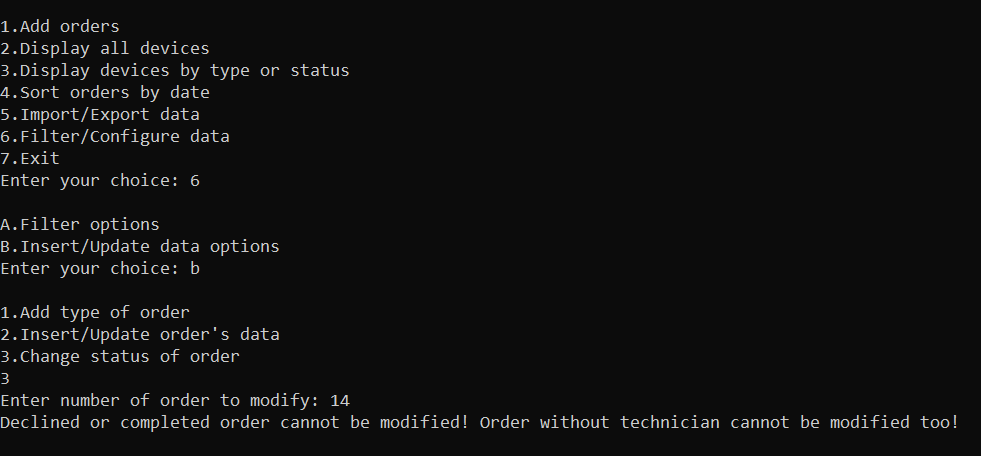
Фиг. 32.Успешно добавяне на данни на поръчката (техник, проблем, цена)





Фиг. 33.Успешно модифициране статуса на поръчка (приета)

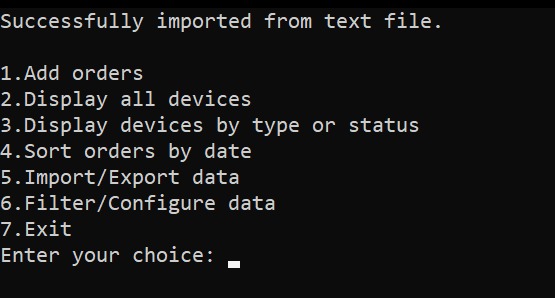
Снимка на изгледа с примерни изходни данни



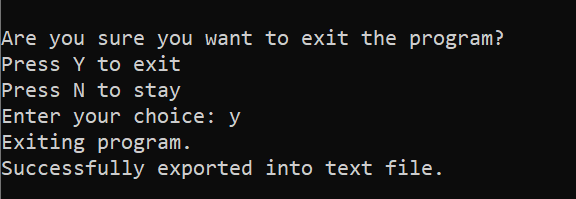
Фиг. 34.Неуспешно модифициране на типа на поръчката

Допълнение трето

Снимка на изгледа с примерни входни данни



Фиг. 35.Успешно четене от текстови файл при стартиране на програмата



Фиг. 36.Успешен запис в текстови файл след приключване на програмата