Технически Университет Варна

**Факултет:** Факултет по изчислителна техника и автоматизация

**Катедра:** Софтуерни и интернет технологии

**Специалност:** Софтуерни и интернет технологии

**Тема на проект:** Библиотека

**Име:** Тодор Иванов Йорданов

**Факултетен номер:** 20621600

**Вариант** **106**

Да се напише компютърна програма, реализираща информационна система, която поддържа библиотека. Програмата съхранява и обработва данни за наличните в момента книги (Каталожен номер на книгата, Име на книга, Автор, брой книги, дата на издание на книгата, брой заемания, наличност). Максималния брой различни книги в библиотеката е 100. Всяка книга може да има наличност от 0 до 5 броя.

**Базова задача – Сложност ниска**

A. Меню за избор на функциите в програмата. (7 седмица)

B. Добавяне на нови книги (7-8 седмица)

a. Добавяне на една нова книга

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

C. Извеждане на всички книги на екрана (8-9 седмица)

a. Извеждане на книги с най-ниска цена

* 1. b. Извеждане на книги на даден автор

**Допълнение Първо – Сложност средна (+ Базова задача)**

D. Заемане на книга – въвежда се име на книга: (9-10 седмица)

a. Ако присъства в библиотеката трябва да се потвърди и да се отрази в заета книга.

* 1. b. Ако книгата не присъства в колекцията на библиотеката трябва да се изведе пояснително съобщение
  2. c. Ако всички бройки от книгата са заети да се изведе подходящо съобщение

**Допълнение Второ – Сложност висока (+ Базова задача + Допълнение Първо)**

E. Връщане на книгата по име или уникален номер на книга (10-11 седмица)

a. Въвежда се името на книгата за връщане и се отброява като върната

* 1. b. Въвежда се уникалния номер на книгата за връщане и се отброява като върната

**Допълнение Трето – Сложност висока (+ Базова задача + Допълнение Първо + Допълнение Второ)**

F. Одит на книгите в библиотеката в под меню (11-12 седмица)

a. Извеждане на книгите в подреден ред по:

i. По името на автора

ii. По година на издаване

b. Извеждане на книгите с Най-много заемания, сортирани по автор

c. Извеждане на книгите с Най-малко заемания, сортирани по година на издаване

G. Данните в програмата да могат да се запазват във файл между две стартирания на програмата.

**Допълнение Четвърто – (за допълнителни точки)**

H. Допълнителни условия:

a. За точка B при добавянето на една книга да се даде възможност, да се избере дали да се въведе нова книга или да прекъсне въвеждането.

* 1. b. За точка C да се реализира изписването по 5 книги на страница. Пример: В библиотеката има 27 книги. Извеждат се 5 книги избира се следваща страница извеждат се следващите 5 книги и т.н.
  2. c. За точка G да се записват данните в двоичен файл

**Сорс код : Файл с данни :**

**Анализ на решението и упътване за употреба**

**Главна функция “main()”**

Благодарение на функция “**main()**” цялата програма се изпълнява като в отделни фрагменти от кода се извикват определни функции, които изпълняват определени действия съответстващи на условието на задачата. В началото на функция “**main()**” са декларирани 6 броя променливи от тип **int** , които в по-късен етап ще послужат за изпълнението на дадено действие. След това е декларирана и иницилизирана променлива “**n**” от тип **int** имаща стойност 0. Тази променлива **n** в по-късен етап от програмата ще съотвества на броя книги в библиотеката (т.е броя структури в библиотеката). След това е деклариран масив “**LIBRARY[N]**” с **N** на брой структури от потребителски тип “**kniga**”. След това е декларирана променлива “**answer**” от тип **string**. След това се извиква функция “**loadfile(LIBRARY)**” която е за четене от външен файл и след изпълнението на функцията, резултатът , който тя връща е броят на книгите в библиотеката. Променливата **n** присвоява този резултат. Посредством цикъл “**do {…}while(…);**” съдържащ в себе си “**switch(…)**” се извикват определени функции според действията на потребителя. Някои опции на “**switch(…)**”в зависимост от избора на потребителя съдържат в себе си други цикли “**do {…}while(…);**” , а те от своя страна съдържат друг “**switch(…)**” поради простата причина ,че програмата не съдържа в себе си само едно меню, а напротив. Тя се състои от едно главно меню, което от своя страна съдържа в себе си набор от под менюта, като някои под менюта съдържат в себе си други подменюта. Всичко това е изпълнено и съгласувано според условието на заданието, като са добавени някои подобрения (повечето опции от главното меню водят към други под менюта) за по-добро удобство. В главния цикъл “**do {…}while(…);**” посредством “**system(“cls”);**” и “**system("color 74");**” екрана се изчиства и се задава сив фон и червени букви. След това се извиква функция “**menu\_A()**” , която връща стойност съответстваща на избора , който потребителя е направил в главното меню на програмата. Променливата “**choiceA**” присвоява тази стойност. След това се изпълнява главния оператор “**switch(choiceA)**” . Според избора, който потребителия е направил в главното меню на програмата се извършва определено действие. Избора на потребителя е съхранен в променливата “**choiceA**” като в зависимост какъв е той се изпълнява различен **case** в оператора “**switch(choiceA)**”.

При **case ‘B’** се навлиза в нов цикъл “**do {…}while(…);**”. Функция “**PodMenuB()**” се извиква, като резултатът, който тя връща, съответства на избора, който потребителят е направил в **под меню B** на програмата. Променливата “**choiceB**” присвоява този резултат. Реализира се нов оператор “**switch(choiceB)**” , като според избора , който потребителя е направил по-рано в **под менб B** на програмата се изпълнява различен случай (**case**). При **case ‘a’** с оператор **if** се проверява дали локалната променлива **n** съответстваща броя на книгите в библиотеката е равна на символната константа **N** , която е декларирана и иницилизирана със стойност равна на **100** преди функция “**main()**”. Ако стойността на локалната променлива **n** e равна на стойността на символната константа **N**. Се извежда съобщение на екран,че няма място за повече книги в библиотеката, защото всички места за книги са заети, а именно според условието на задачата в библиотеката може да има най-много 100 различни книги. Ако обаче стойността на локалната променлива **n** се различава от стойността на символната константа **N** тогава се извиква функция “**append(n)**”, която е за добавяне на една книга към библиотеката. След това стойността на локалната променлива **n** нараства със 1. Следва цикъл “**while(n!=N)**” който отново сравнява стойностите на двете променливи и ако те се различават цикълът се изпълнява веднъж като се извежда съобщение което пита потребителя дали би желал да добави още една книга към библиотеката. Въвежда се отговор от тип **string.** Ако отговорът е “**yes**”, ”**Yes**” или “**YES**” програмата дава възможност на потребителя да въведе данни за още една книга в библиотеката. Ако отговорът е различен от посочените 3 отговора по-горе , то програмата прекъсва въвеждането на една книга и излиза от **case ‘a’**. При всеки отговор “**yes**” преди програмата да даде възможнст на потребителя да добави една книга в библиотеката първо проверява дали стойностите на променливите **n** и **N** се различават. Ако те са еднакви това означава че всички места в библиотеката са пълни и директно излиза от **case ‘a’**. При **case ‘b’** се извикват 3 функции. Първо се извиква функция “**loadfile(LIBRARY)**” която чете записаните структури от външния файл, ако има такива. След това се извиква функция “**input\_List(LIBRARY,n)**”,която дава възможнст за въвеждане на списък от книги в библиотеката. Променливата **n** присвоява стойността, която извиканата функция “**input\_List(LIBRARY,n)**” връща като резултат от изпълнението си . Стойността, която се състои в локалната променлива **n** отговаря на броя книги в библиотеката. След това се извиква функция “**saveinfile(LIBRARY,n)**”,която записва промените след изпълнението на функция “**input\_List(LIBRARY,n)**” във външен файл.След това чрез “**break**” се излиза от **case ‘b’**. Ако изборът на потребителят в **под менб B** e ‘**c**’, тогава програмата излиза от **под меню B** и връща потребителя в **главното меню** на програмата.

При избор на потребителя от **главното меню** на опция ‘**C**’ програмата изпълнява заложения код в **case ‘C’** на главния оператор “**switch()**” във функция “**main()**”.Първо се извиква функция “**loadfile(LIBRARY)**”, която чете записаните структури от външния файл, ако има такива. След това стойността която функция “**loadfile(LIBRARY)**” връща след изпълнението си се присвовява на локалната променлива **n** , която съответства на броя книги в библиотеката. След това се извиква функция “**output\_All(LIBRARY,n)**” която извежда всички книги от библиотеката и техните данни ако въобще има такива. След това се изпълнява нов цикъл “**do(…)while(…);**”. В този цикъл се извиква функция “**PodMenuC()**” като след изпълнението си тя връща резултат съответстващ на избора на потребителя ,който той е направил в **под меню С** на програмата. След това променлива “**choiceC**” присвоява резултатът , който функцията “**PodMenuC()**” е върнала след изпълнението си. След това се изпълнява оператър “**switch(choiceC)**” , като според изборът, който потребителят е направил по-рано се изпълнява различен случай (**case**). При **case’a’** първо се извиква функция “**loadfile(LIBRARY)**” която чете записаните структури от външния файл, ако има такива. След това стойността която функция “**loadfile(LIBRARY)**” връща след изпълнението си се присвовява на локалната променлива **n** , която съответства на броя книги в библиотеката. След това се извиква функция “**output\_Min\_Price(LIBRARY, n);**” която извжда книгите с най-ниска цена в библиотеката на екран. След това програмата напуска **case’a’**. При **case’b’** първо се извиква функция “**loadfile(LIBRARY)**” която чете записаните структури от външния файл, ако има такива. След това стойността която функция “**loadfile(LIBRARY)**” връща след изпълнението си се присвовява на локалната променлива **n** , която съответства на броя книги в библиотеката. След това се извиква функция “**output\_Books\_Of\_Certain\_Author(LIBRARY, n);**” която извжда книгите на даден автор в библиотеката на екран. След това програмата напуска **case’b’**. Ако потребителят избере опция ‘**c**’ от **под меню C** програмата излиза от **под меню C** и се връща в **главното меню** на програмата.

При избор на потребителя от **главното меню** на опция ‘**D**’ програмата изпълнява заложения код в **case ‘D’** на главния оператор “**switch()**” във функция “**main()**”.Първо се извиква функция “**loadfile(LIBRARY)**”, която чете записаните структури от външния файл, ако има такива. След това стойността която функция “**loadfile(LIBRARY)**” връща след изпълнението си се присвовява на локалната променлива **n** , която съответства на броя книги в библиотеката.След това се извиква функция “**Zaemane\_Na\_Kniga\_Po\_Ime(LIBRARY, n);**” ,която дава възможност на потребителя да заеме временно дадена книга от библиотеката по въведено име . След това промените извършени от потребителя се запазват във външния файл посредством извикване на функция “**saveinfile(LIBRARY,n)**”. След изпълнението на **case D** програмата връща потребителя в главното меню автоматично.

При избор на потребителя от **главното меню** на опция ‘**E**’ програмата изпълнява заложения код в **case ‘E’** на главния оператор “**switch()**” във функция “**main()**”. Програмата навлиза в нов цикъл “**do(…)while(…);**”. Извиква се функция “**PodMenuE()**”, като резултатът който тя връща след изпълнението си съответства на избора който потребителя е направил в **под меню Е**. Този избор се присвоява на променливата “**choiceE**”. След това се изпълнява оператор “**switch(choiceE)**”. При **case ‘a’** първо се извиква функция “**loadfile(LIBRARY)**”, която чете записаните структури от външния файл, ако има такива. След това стойността която функция “**loadfile(LIBRARY)**” връща след изпълнението си се присвовява на локалната променлива **n** , която съответства на броя книги в библиотеката. След това се извиква функция “**Vrushtane\_Na\_Kniga\_Po\_Ime(LIBRARY, n)**” , която дава възможност на потребителя да върне книга в библиотеката, която преди това е заел по въведено име на книгата. След това се извиква функция “**saveinfile(LIBRARY,n)**”,която записва промените след изпълнението на функция “**Vrushtane\_Na\_Kniga\_Po\_Ime(LIBRARY, n)**”във външен файл. След това програмата напуска **case ‘a’**. При **case ‘b’** първо се извиква функция “**loadfile(LIBRARY)**”, която чете записаните структури от външния файл, ако има такива. След това стойността която функция “**loadfile(LIBRARY)**” връща след изпълнението си се присвовява на локалната променлива **n** , която съответства на броя книги в библиотеката. След това се извиква функция “**Vrushtane\_Na\_Kniga\_Po\_Call\_Number(LIBRARY, n)**” , която дава възможност на потребителя да върне книга в библиотеката, която преди това е заел по въведен каталожен номер на книгата. След това се извиква функция “**saveinfile(LIBRARY,n)**”,която записва промените след изпълнението на функция “**Vrushtane\_Na\_Kniga\_Po\_Call\_Number(LIBRARY, n)**”във външен файл. След това програмата напуска **case ‘b’**. При избор на опция ‘**c**’ от страна на потребителя, когато е в **под меню Е** , програмата излиза от **под меню Е** и зарежда на екран отново **главното меню**.

При избор на потребителя от **главното меню** на опция ‘**F**’ програмата изпълнява заложения код в **case ‘F’** на главния оператор “**switch()**” във функция “**main()**”.Програмата навлиза в нов цикъл “**do(…)while(…);**”. Извиква се функция “**PodMenuF()**”, като резултатът който тя връща след изпълнението си съответства на избора който потребителя е направил в **под меню F**. Този избор се присвоява на променливата “**choiceF**”. След това се изпълнява оператор “**switch(choiceF)**”. При **case ‘a’** програмата навлиза в нов цикъл “**do(…)while(…);**”.Извиква се функция “**PodMenu\_aF()**”, като резултатът който тя връща след изпълнението си съответства на избора който потребителя е направил в **под меню F**. Този избор се присвоява на променливата “**option\_aF**”. След това се изпълнява оператор “**switch(option\_aF)**”. При **case 1** първо се извиква функция “**loadfile(LIBRARY)**”, която чете записаните структури от външния файл, ако има такива. След това стойността която функция “**loadfile(LIBRARY)**” връща след изпълнението си се присвовява на локалната променлива **n** , която съответства на броя книги в библиотеката. След това се извиква функция “**Sort\_By\_Name\_Of\_Author(LIBRARY, n)**” , която сортира книгите в библиотеката по име на автор тоест по азбучен ред . След това програмата излиза от **case 1** . При **case 2** първо се извиква функция “**loadfile(LIBRARY)**”, която чете записаните структури от външния файл, ако има такива. След това стойността която функция “**loadfile(LIBRARY)**” връща след изпълнението си се присвовява на локалната променлива **n** , която съответства на броя книги в библиотеката. След това се извиква функция “**Sort\_By\_Year\_Of\_Release(LIBRARY, n)**” , която сортира книгите в библиотеката по година на издание от най-стара до най-нова книга. Ако две книги са издадени през една и съща година, тогава се гледат месецът и деня на издание. След това програмата излиза от **case 2** . При избор на потребителя на опция ‘c’ програмата излиза от **под меню ‘a’** и връща потребителя в **под меню F**. В **под меню F** при **case ‘b’** се изпълнява оператор **if**, който сравнява дали стойността на **n** е равна на 0. Ако това е вярно, на екран се появява събщение че няма книги в библиотеката . В противен случай се извиква функция “**loadfile(LIBRARY)**”, която чете записаните структури от външния файл, ако има такива. След това стойността която функция “**loadfile(LIBRARY)**” връща след изпълнението си се присвовява на локалната променлива **n** , която съответства на броя книги в библиотеката. След това се извиква функция “**Sort\_Most\_Lendings\_By\_Author(LIBRARY, n)**” , която извежда на екран книгите в библиотеката с най-много заемания сортирани по име на автор(азбучен ред). След това програмата напуска **case ‘b’** . При **case ‘c’** се изпълнява оператор **if**, който сравнява дали стойността на **n** е равна на 0. Ако това е вярно, на екран се появява събщение че няма книги в библиотеката . В противен случай се извиква функция “**loadfile(LIBRARY)**”, която чете записаните структури от външния файл, ако има такива. След това стойността която функция “**loadfile(LIBRARY)**” връща след изпълнението си се присвовява на локалната променлива **n** , която съответства на броя книги в библиотеката. След това се извиква функция “**Sort\_Least\_Lendings\_By\_Year(LIBRARY, n)**” , която извежда на екран книгите в библиотеката с най-малко заемания сортирани по година на издание(от най-стари до най-нови). След това програмата напуска **case ‘c’**. При избор на потребителя на опция ‘**d**’ в **под меню F** , програмата излиза от **под меню F** и се връща в **главното меню**.

При избор на потребителя на **опция ‘G’** от **главното меню**. Екрана се чисти и излиза съобщение което информира потребителя че въведените данни и направените промени от страна на потребителя ако въобще е правил такива са запазени. Дори и потребителя не избере **опция ‘G’** от **главното меню** , програмата пак ще запази всички въведени данни и направени промени ако въобще има такива. Това е ще се извърши с цел да не се изгубят данни въведени от потребителя при излизане от програмата.

При избор на потребителя на **опция ‘I’** от **главното меню** , програмата се затваря.

**Функция char menu\_A()**

Тази функция извежда на екран **главното меню на програмата** при извикването си във функция “**main()**”. Дава се възможност на потребителя да въведе своя избор в зависимост от това какво иска да върши програмата като на края на функцията , изборът който потребителя е направил се връща като резултат във функция “**main()**”.

**Функция char PodMenuB()**

Тази функция извежда на екран **под меню B** при извикването си във функция “**main()**”. Дава се възможност на потребителя да въведе своя избор в зависимост от това какво иска да върши програмата като на края на функцията , изборът който потребителя е направил се връща като резултат във функция “**main()**”.

**Функция char PodMenuC()**

Тази функция извежда на екран **под меню C** при извикването си във функция “**main()**”. Дава се възможност на потребителя да въведе своя избор в зависимост от това какво иска да върши програмата като на края на функцията , изборът който потребителя е направил се връща като резултат във функция “**main()**”.

**Функция char PodMenuE()**

Тази функция извежда на екран **под меню E** при извикването си във функция “**main()**”. Дава се възможност на потребителя да въведе своя избор в зависимост от това какво иска да върши програмата като на края на функцията , изборът който потребителя е направил се връща като резултат във функция “**main()**”.

**Функция char PodMenuF()**

Тази функция извежда на екран **под меню F** при извикването си във функция “**main()**”. Дава се възможност на потребителя да въведе своя избор в зависимост от това какво иска да върши програмата като на края на функцията , изборът който потребителя е направил се връща като резултат във функция “**main()**”.

**Функция int PodMenu\_aF()**

Тази функция извежда на екран **под меню ‘a’** на **под меню F** при извикването си във функция “**main()**”. Дава се възможност на потребителя да въведе своя избор в зависимост от това какво иска да върши програмата като на края на функцията , изборът който потребителя е направил се връща като резултат във функция “**main()**”.

**Функция kniga input()**

В тази функция е декларирана една променлива ‘**k**’ от потребителски тип структура “**kniga**”. В началото на функцията “**k**” е инициализирана като празна структура. След това се дава възможност на потребителя да се въведат данни за една книга както следва : **име**,**автор**,**каталожен номер**,**цена**,**дата на издание**(година на издание,месец на издание,ден на издание),**брой книги в библиотеката от дадената книга**,**заети бройки от дадената книга**. След тези данни които се въвеждат от потребителя се извеждат **наличните бройки от дадената книга**. След това функцията връща като резултат данните които потребителя вече е въвел под формата на структура.

**Функция int input\_List(kniga library[], int n)**

Формални параметри на тази функция са **kniga library[]** и **int n**. Най-напред чрез функция **do{…}while(…);** се дава възможност на потребителя да въведе колкото броя различни книги (**m**) иска да добави в библиотеката.(в диапазон от 1 до 100) . След това чрез оператор **if** се проверява дали броят на книгите,които е въвел потребителя + броя на вече същестуващите различни книги в библиотеката дали надхвърля ограничението а именно (100). Ако ограничението не се надхвърли тогава се дава възможност на потребителя да въведе списък от книги като във един цикъл започващ от **n(**старияброй книги**)** и продължаващ докато не се достигне новият брой книги , който е стария + новия (**n**+**m**) се извиква функция “**input()**” и след изпълнението си тя връща като стойност една структура . Елемент на масива от структури “**library[i]**” ,който също е една структура от потребителски тип “**kniga**” присвоява структурата от вече изпълнената функция “**input()**”. След като програмата излезе от цикъла функцията “**int input\_List(kniga library[], int n)**” връща като стойност сбора на старите + новите книги (**n**+**m**). Ако при проверката с оператор **if** се установи че (**m**+**n**) e равно на 100, тогава на екран се извежда съобщение , което информира потребителя че няма място за **m** на брой нови книги и изписва за колко книги има място библиотеката(**100**-**n**).След това като резултат функцията “**int input\_List(kniga library[], int n)**” връща броя на старите книги (**n**).

**Функция void saveinfile(kniga library[], int n)**

В тази функция се декларира локална файлова променлива “**fp**”. След това се отваря външния файл , който е двоичен за запис. Ако има проблем при отварянето на външния файл, програмата прекъсва. В противен случай целия масив **library[]** се записва във външния файл структура по структура с целия му размер(“**n**” структури по размера на структурата “**kniga**”).След това файла се затваря.

**Функция void append(int n)**

В началото на тази функция се декларира една променлива “**a**” от потребителски тип структура “**kniga**” и се инициализира като празна. Декларира се локална файлова променлива “**fp**”. След това се отваря външния файл, който е двоичен за добавяне. Ако има проблем с отварянето на външния файл , програмата прекъсва. В противен случай се извиква функция “**input()**” и след изпълнението си тя връща като стойност една структура . Променливата “**a**”,който също е една структура от потребителски тип “**kniga**” присвоява структурата от вече изпълнената функция “**input()**”. След това структурата “**a**” се записва еднократно към края на външния файл и файла се затваря.

**Функция void output\_All(kniga library[], int n)**

В тази функция първоначално се декларира една булева променлива “**flag**”и се иницилизира със стойност **лъжа(false)**. След това в цикъл започващ от 0 и изпълняващ се докато **i** е по-малко от **n** една по една се извждат данните за всяка една книга(структура ) на екран. В цикъла булевата променлива “**flag**” присвоява стнойност **истина (true)**. След като програмата излезе от цикъла чрез оператор **if** проверяваме стойността на булевата променлива “**flag**”. Ако те я е **false** на екран се извежда че няма въведени книги в библиотеката.

**Функция int loadfile(kniga library[])**

Първоначално се декларира локална файлова променлива “**fp**”. След това се декларира променлива “ **pos**” от тип **int**. След това се декларира променлива “**n**” от тип **int** и се инициализира със стойност 0. После се декларира една променлива “**b**” от потребителски тип структура “**kniga**” и се инициализира като празна. След това се отваря външния файл , който е двоичен за четене. При проблем с отварянето или намирането на файла функцията връща като стойност променливата **n** имаща стойност 0. В противен случай , ако файла се намери и отвори без проблеми тогава програмата позиционира указателя в края на външния файл **.** След това променливата “**pos**” от тип **int** присвоява броя на байтовете докъдето е стигнал указателя и външния файл се затваря . След това променливата “**n**” присвоява резултатът от делението на броя байтове на размера на една структура като резултатът който се присвоява от променливата “**n**” показва колко е броят на книгите в библиотеката. След това отново се отваря външния файл , който е двоичен за четене. При проблем с отварянето на външния файл , програмата прекъсва. В противен случай програмата навлиза в цикъл започващ от 0 и изпълняващ се докато i е по-малко от **n(броя на книгите в библиотеката).** В този цикъл програмата чете от файла в структурата “**b**” една по една книгите като подава размера на структурата и адреса на променливата “**b**”. **По този начин изчита всички данни от файла**. След това структурата library[i] присвоява структурата “**b**”, като по този начин се съхраняват една по една структурите които са били прочетени от файла в масива “**library[]**”. След това файла се затваря. На края функцията връща като стойност броя на изчислените книги “**n**”.

**Функция void output\_Min\_Price(kniga library[], int n)**

В началото на тази функция се декларира променлвиа “**br**” от тип **int** и се инициализира със стойност 0. След това се декларира булева променлива “**flag**” и се инициализира със стойност **лъжа(false)**. След това се декларират два масива от указатели към структури “**temp[N]**” и “**loc[N]**” от тип **kniga** . В един цикъл “**for (int i = 0; i < n; i++)”** структура по структура **loc[i]** присвоява адреса на структурата **library[i]**. След това в двоен цикъл по метода на мехурчето структурите на масива **loc[N]** се сортират по възходящ ред.След това оператор if проверява дали **n(броя на книгите)** е повече от 1. Ако е така булевата променлива **flag** си присвоява стойност **истина(true)**. След това програмата навлиза в цикъл **“for (int i = 1; i < n; i++)”** като се извеждат данните за първата структура (книга) от масива **loc[N]** понеже след сортировката по метода на мехурчето във възходящ ред книгата която е първа от масива е с най ниска цена и след това се проверява дали има други книги от масива **loc[N]** чиито цени са равни на цената на първата книга в масива . Ако има и други такива книги отговарящи на условието то и те се извеждат на екран. Ако първия оператора **if** по-рано е дал стойност лъжа тогава се програмата се премества във втория оператор **if** където се проверява дали броя на книгите е равен на 1. Ако е така се извеждат данните само за първата книга в масива защото при това положение ако броя на книгите в масива не е по-голям от 1 то ще има само една книга и тогава няма да има смисъл да се прави проверка за други книги и да се върти цикъл for . При това положение булевата променлива “**flag**” присвоява стойност **истина(true)**. Ако в тази функция нито един от двата оператора **if** не даде стойност истина тогава програмата отива директно на края на функцията където има оператор **if** който проверява стойността на булевата променлива “**flag”.** Ако тя има стойност лъжа то на екрана се извежда че в библиотеката няма въведени книги.

**Функция void output\_Books\_Of\_Certain\_Author(kniga library[], int n)**

В началото на функцията се декларира булева променлива “**flag**” и се инициализира със стойност **лъжа(false)**. След това се декларира променлива **AUTHOR[100]** която е **string** оттипа **C string .** След това буфера се чисти от паразитни символи. Дава се възможност на потребителя да въведе името на автора чиято книга иска да види изведена на екрана. В един цикъл който съдържа в себе си оператор **if** се сравнява името на въведения автор от потребителя и името на автора на всяка една отделна книга от библиотеката. Ако има съвпадение сравняването ще върне стойност 0 в оператора **if** и затова преди функцията за сравнение на двете имена на автори се слага отрицание **!(not)** което би обърнало резултата 0 в 1 ако има съвпадение и тогава биха се извели данните на дадена книга , като с това и булевата променлива “**flag”** би си присвоила стойност истина . След като се излезе от цикъла следва оператор **if** който проверява стойността на булевата променлива “**flag**”. Ако тя е равна на лъжа тогава на екрана ще се изведе съобщение което ще информира потребителя че в библиотеката няма книга на автора който той е въвел преди това .

**Функция void Zaemane\_Na\_Kniga\_Po\_Ime(kniga library[], int n)**

В началото на функцията е деклариран масив “**choosed[N]**” от **N** на брой цели числа.След това се декларира булева променлива “**flag**” и се инициализира със стойност **лъжа(false)**. След това се декларира променлива **NameOfBook[100]** която е **string** оттипа **C string .** След това се дава възможност на потребителя да въведе името на книгата която иска да заеме временно от библиотеката. В един цикъл for посредством два оператора **if** се проверява дали книгата която потребителя е въвел по рано присъства в библиотеката и колко бройки от нея са налични . Ако книгата присъства в библиотеката и има налични бройки то дадената книга се заема като в нейните данни се отразява че броя на заетите книги от дадената книга нараства с 1 и броя на наличните книги от дадената книга намалява с 1. Тези действия не се визулизират на екрана в момента на изпълнението на функцията. Разликата може да се види ако предварително се знаят данните на дадена книга и се избере **под меню C** след като функцията **Zaemane\_Na\_Kniga\_Po\_Ime(kniga library[], int n)** е била изпълнена. На екрана в момента на изпълняването на действието се извeждa съобщение което информира потребителя че дадена книга е била заета успешно от библиотеката. В противен случай ако няма налични бройки от дадена книга в библиотеката на екрна ще се изведе съобщение което ще информира потребителя че всички бройки от книгата която иска да заеме са изчерпани.Ако някой от двата оператора **if** се изпълни поне веднъж в хода на въртене на целия цикъл то булевата променлива “**flag**” ще си присвои стойност **истина**. След като се излезе от цикъла се изпълнява оператор **if** който проверява стойността на булевата променлива “**flag**”. Ако тя е **лъжа** ще се изведе съобщение на екрана което ще информира потребителя че книгата която иска да заеме не съществува в библиотеката.

**Функция void Vrushtane\_Na\_Kniga\_Po\_Ime(kniga library[], int n)**

В началото на функцията е деклариран масив “**choosed[N]**” от **N** на брой цели числа.След това се декларира булева променлива “**flag**” и се инициализира със стойност **лъжа(false)**. След това се декларира променлива **NameOfBook[100]** която е **string** оттипа **C string .** След това се дава възможност на потребителя да въведе името на книгата която иска да върне в библиотеката. В един цикъл for посредством два оператора **if** се проверява дали книгата която потребителя е въвел по рано присъства в библиотеката и колко бройки от нея са налични . Ако книгата присъства в библиотеката и не всичките и бройки са в наличност то дадената книга се връща като в нейните данни се отразява че броя на заетите книги от дадената книга намалява с 1 и броя на наличните книги от дадената книга нараства с 1. Тези действия не се визулизират на екрана в момента на изпълнението на функцията. Разликата може да се види ако предварително се знаят данните на дадена книга и се избере **под меню C** след като функцията **Vrushtane\_Na\_Kniga\_Po\_Ime(kniga library[], int n)** е била изпълнена. На екрана в момента на изпълняването на действието се извeжда съобщение което информира потребителя че дадена книга е била върната успешно от библиотеката. В противен случай ако всички бройки са налични от дадена книга в библиотеката на екрна ще се изведе съобщение което ще информира потребителя че всички бройки от книгата която иска да върне вече са в пълна наличност .Ако някой от двата оператора **if** се изпълни поне веднъж в хода на въртене на целия цикъл то булевата променлива “**flag**” ще си присвои стойност **истина**. След като се излезе от цикъла се изпълнява оператор **if** който проверява стойността на булевата променлива “**flag**”. Ако тя е **лъжа** ще се изведе съобщение на екрана което ще информира потребителя че книгата която иска да върне не съществува в библиотеката.

**Функция void Vrushtane\_Na\_Kniga\_Po\_Call\_Number(kniga library[], int n)**

В началото на функцията е деклариран масив “**choosed[N]**” от **N** на брой цели числа.След това се декларира булева променлива “**flag**” и се инициализира със стойност **лъжа(false)**. След това се декларира променлива **Call\_Number[50]** която е **string** оттипа **C string .** След това се дава възможност на потребителя да въведе каталожния номер на книгата която иска да върне в библиотеката. В един цикъл for посредством два оператора **if** се проверява дали книгата с каталожен номер коойто потребителя е въвел по рано присъства в библиотеката и колко бройки от нея са налични . Ако книгата присъства в библиотеката и не всичките и бройки са в наличност то дадената книга се връща като в нейните данни се отразява че броя на заетите книги от дадената книга намалява с 1 и броя на наличните книги от дадената книга нараства с 1. Тези действия не се визулизират на екрана в момента на изпълнението на функцията. Разликата може да се види ако предварително се знаят данните на дадена книга и се избере **под меню C** след като функцията **Vrushtane\_Na\_Kniga\_Po\_Call\_Number(kniga library[], int n)** е била изпълнена. На екрана в момента на изпълняването на действието се извeжда съобщение което информира потребителя че дадена книга е била върната успешно от библиотеката. В противен случай ако всички бройки са налични от дадена книга в библиотеката на екрна ще се изведе съобщение което ще информира потребителя че всички бройки от книгата която иска да върне вече са в пълна наличност .Ако някой от двата оператора **if** се изпълни поне веднъж в хода на въртене на целия цикъл то булевата променлива “**flag**” ще си присвои стойност **истина**. След като се излезе от цикъла се изпълнява оператор **if** който проверява стойността на булевата променлива “**flag**”. Ако тя е **лъжа** ще се изведе съобщение на екрана което ще информира потребителя че книгата с каталожен номер който е въвел по-рано не съществува в библиотеката.

**Функция void Sort\_By\_Name\_Of\_Author(kniga library[], int n)**

В началото на функцията се декларира булева променлива “**flag**” и се инициализира със стнойност **лъжа**. След това се декларират два масива от указатели към структури “**temp[N]**” и “**loc[N]**” от тип **kniga** . В един цикъл “**for (int i = 0; i < n; i++)”** структура по структура **loc[i]** присвоява адреса на структурата **library[i]**. След това в двоен цикъл по метода на мехурчето структурите на масива **loc[N]** се сортират по възходящ ред според имената на авторите на всяка книга. След това посредством един цикъл “**for (int i = 0; i < n; i++)”** на екран се извеждат всички книги от библиотеката сортирани по имена на автори(**по азбучен ред**).Ако цикълът се изпълни поне веднъж то булевата променлива “**flag**” ще си присвои стойност истина. След това посредством оператор **if** се проверява стойността на булевата променлива “**flag**”. Ако те я **лъжа** на екран ще се изведе съобщение което ще информира потребителя че в библиотеката няма книги.

**Функция void Sort\_By\_Year\_Of\_Release(kniga library[], int n)**

В началото на функцията се декларира булева променлива “**flag**” и се инициализира със стнойност **лъжа**. След това се декларират два масива от указатели към структури “**temp[N]**” и “**loc[N]**” от тип **kniga** . В един цикъл “**for (int i = 0; i < n; i++)”** структура по структура **loc[i]** присвоява адреса на структурата **library[i]**. След това в двоен цикъл по метода на мехурчето структурите на масива **loc[N]** се сортират по възходящ ред според годината на издание на всяка книга. Ако годината на някоя книга съвпадне с годината на друга книга то те сортират отново във възхдящ ред но вече се гледа месеца на издание на книгите. Ако и годините и месеците на книгите съвпадат тогава се проверяват дните ,в които книгите са били издадени и се сортират отново във възходящ ред по метода на мехурчето. След като сортировката на всички книги се извърши тогава се навлиза в цикъл **for** чрез който се извеждат вече сортираните по година на издание книги във възходящ ред. Ако този последен цикъл за функцията се изпълни поне ведънж то булевата променлива **flag** ще си присвои стойност истина. На края на цялата функция чрез оператор **if** се проверява стойността на булевата променлива “**flag**”. Ако те я **лъжа** на екран ще се изведе съобщение което ще информира потребителя че в библиотеката няма книги.

**Функция void Sort\_Most\_Lendings\_By\_Author(kniga library[], int n)**

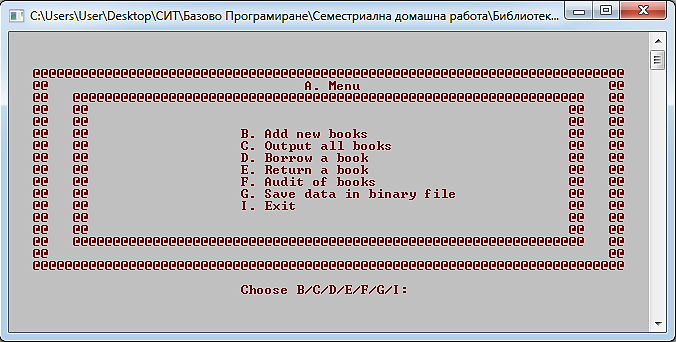
В началото на функцията се декларират три масива от указатели към структури “**temp[N]**” , “**loc[N]**” и “**Most\_Lendings[N]**” от тип **kniga** . В един цикъл “**for (int i = 0; i < n; i++)”** структура по структура **loc[i]** присвоява адреса на структурата **library[i]**. След това посредством двоен цикъл масивът от указатели към структури **loc[N]** се сортира по метода на мехурчето във низходящ ред според броя заемания на книгите в библиотеката. След като се извърши сортировката първата структура **Most\_Lendings[0]**  на масива от указатели към структури **Most\_Lendings[N]** присвоява стойността на първата структура **loc[0]** от вече сортирания масив **loc[N].** Декларира се променлива “**naa**” от тип **int** която ще отговаря за броя на книгите в масива от указатели към структури **Most\_Lendings[N]** и в същото време се инициализира със стойност 1. В цикъл “**for (int i = 1; i < n; i++)**” чрез оператор **if** се проверява дали книгите от масива **loc[N]** от указатели към структури имат същия брой заемания като първата книга от масива а именно книгата която е с най-много заемания . Ако това е така структурата **Most\_Lendings[i]** присвоява стойността на структурата  **loc[i]** като заедно с това се увеличава стойността на локалната променлива **naa** отговаряща за броя на книгите в масива от указатели към струкутри **Most\_Lendings[N].**  След това посредством двоен цикъл for се извършва сортиране по метода на мехурчето във възходящ ред според имената на авторите(**по азбучен ред**) на книгите от масива от указатели към структури **Most\_Lendings[N].** След това чрез цикъл **for (int i = 0; i < naa; i++)** се извеждат книгите от масива от указатели към структури **Most\_Lendings[N]** вече сортирани по азбучен ред.

**Функция void Sort\_Least\_Lendings\_By\_Year(kniga library[], int n)**

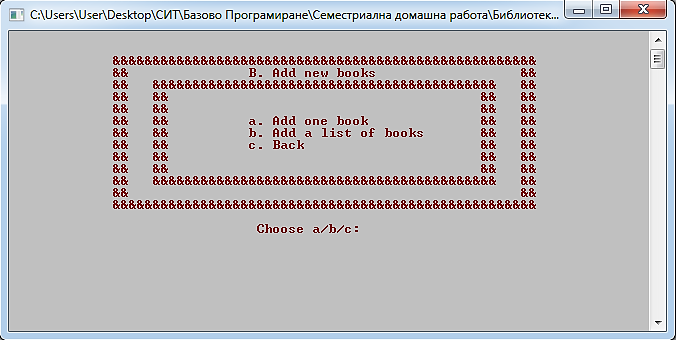
В началото на функцията се декларират три масива от указатели към структури “**temp[N]**” , “**loc[N]**” и “**Least\_Lendings[N]**” от тип **kniga** . В един цикъл “**for (int i = 0; i < n; i++)”** структура по структура **loc[i]** присвоява адреса на структурата **library[i]**. След това посредством двоен цикъл масивът от указатели към структури **loc[N]** се сортира по метода на мехурчето във възходящ ред според броя заемания на книгите в библиотеката. След като се извърши сортировката първата структура **Least\_Lendings[0]**  на масива от указатели към структури **Least\_Lendings[N]** присвоява стойността на първата структура **loc[0]** от вече сортирания масив **loc[N].** Декларира се променлива “**naa**” от тип **int** която ще отговаря за броя на книгите в масива от указатели към структури **Least\_Lendings[N]** и в същото време се инициализира със стойност 1. В цикъл “**for (int i = 1; i < n; i++)**” чрез оператор **if** се проверява дали книгите от масива **loc[N]** от указатели към структури имат същия брой заемания като първата книга от масива а именно книгата която е с най-малко заемания . Ако това е така структурата **Least\_Lendings[i]** присвоява стойността на структурата  **loc[i]** като заедно с това се увеличава стойността на локалната променлива **naa** отговаряща за броя на книгите в масива от указатели към струкутри **Least\_Lendings[N].**  След това посредством двоен цикъл for се извършва сортиране по метода на мехурчето във възходящ ред според годината на издание на книгите(във възходящ ред) от масива от указатели към структури **Least\_Lendings[N]**. Ако годините на издание на две книги съвпаднат тогава се сравняват по месеца и деня на издание. След това чрез цикъл **for (int i = 0; i < naa; i++)** се извеждат книгите от масива от указатели към структури **Least\_Lendings[N]** вече сортирани по възходящ ред.

**Примерно действие на програмата (снимки на входни и изходни данни)**

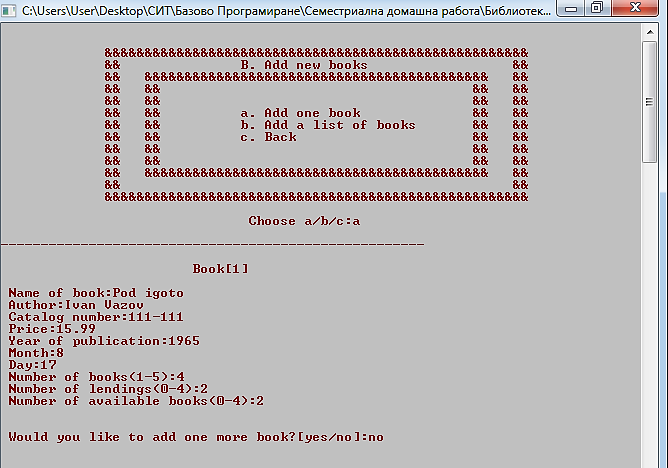
**Начален изглед на програмата**

****

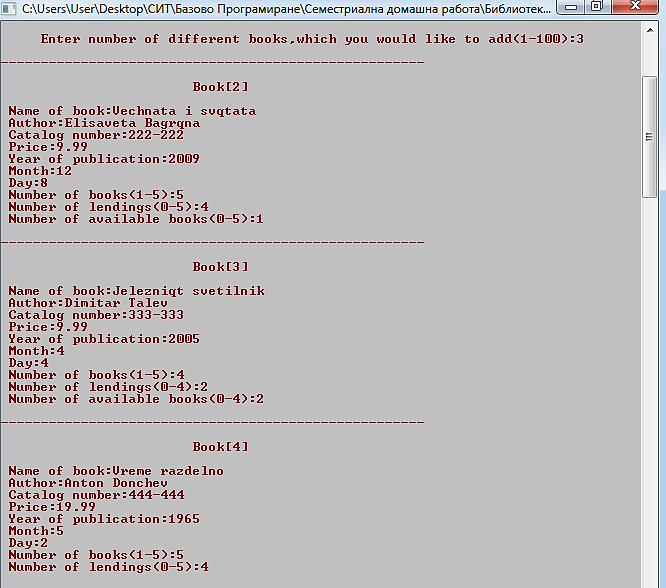
**Изглед при избор на под меню B**

****

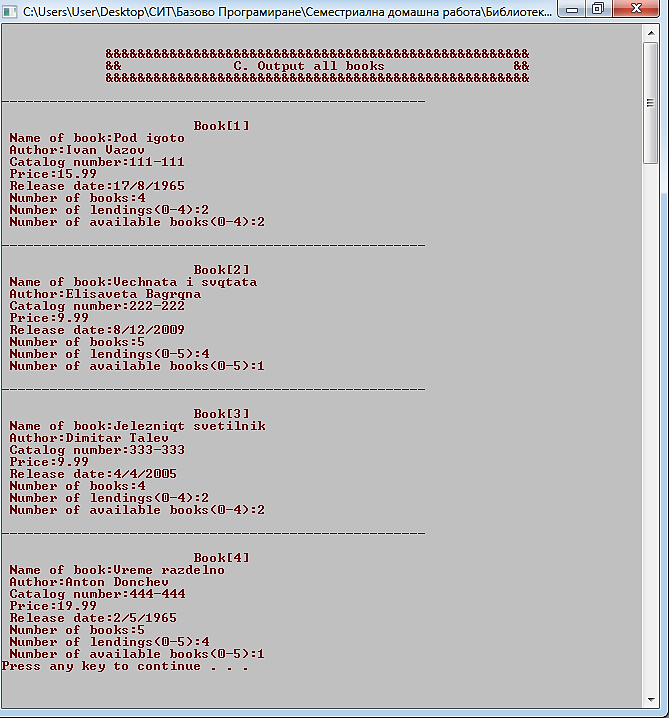
**Изглед при избор на опция а. от под меню В**

****

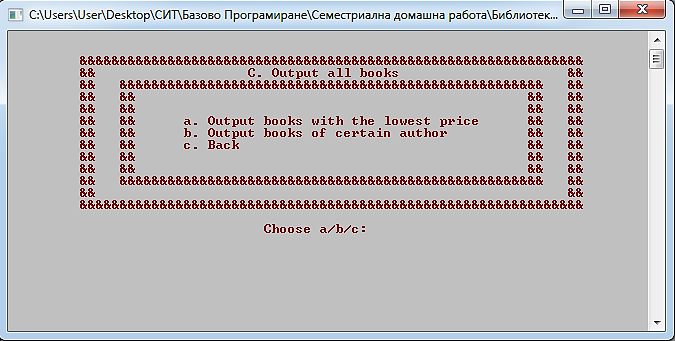
**Изглед при избор на опция b. от под меню В**

****

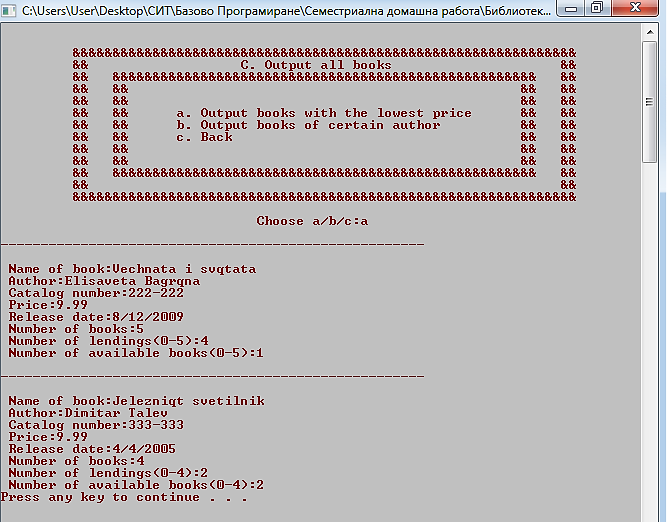
**Изглед при избор на под меню C**

****

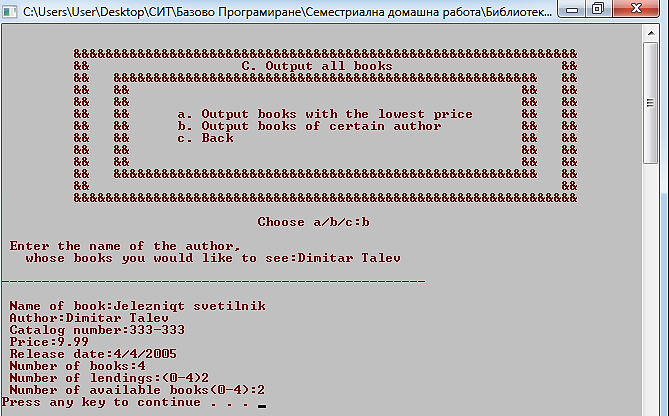
**Изглед след като се навлезе в под меню C**

****

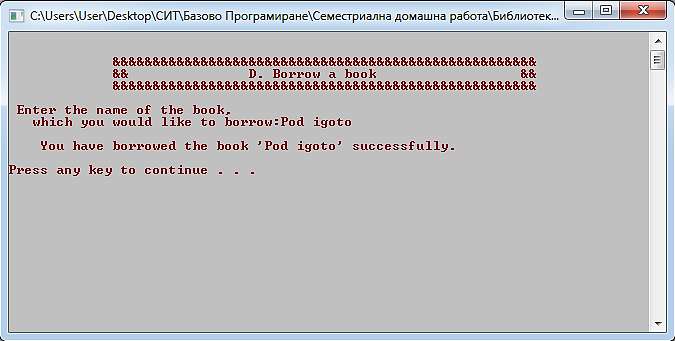
**Изглед при избор на опция a. от под меню C**

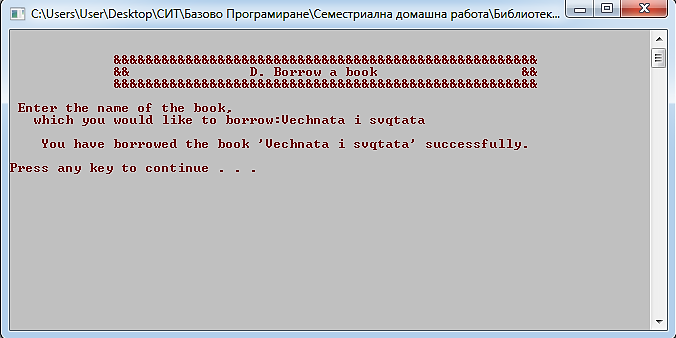
****

**Изглед при избор на опция b. от под меню C**

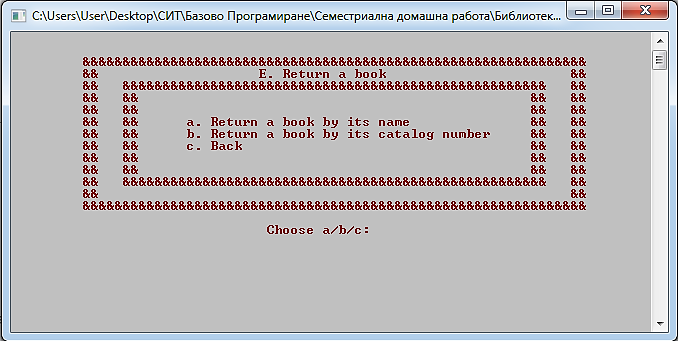
****

**Изглед при избор на опция D от главното меню**

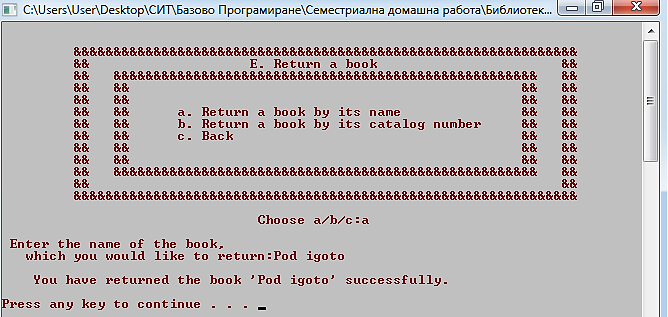
****

****

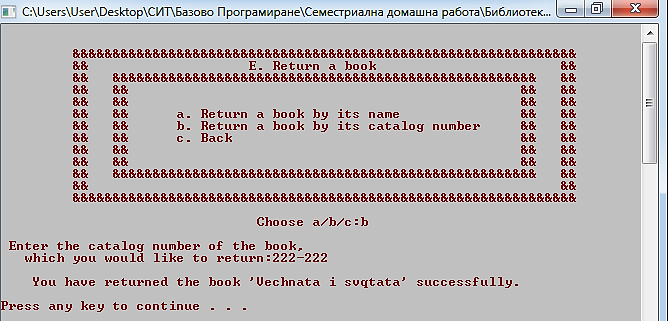
**Изглед при избор на под меню Е**



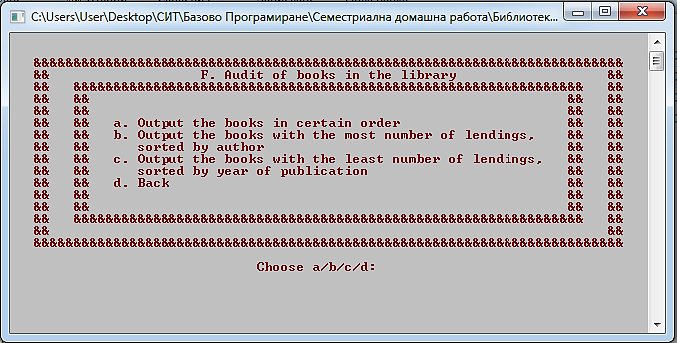
**Изглед при избор на опция a. от под меню E**



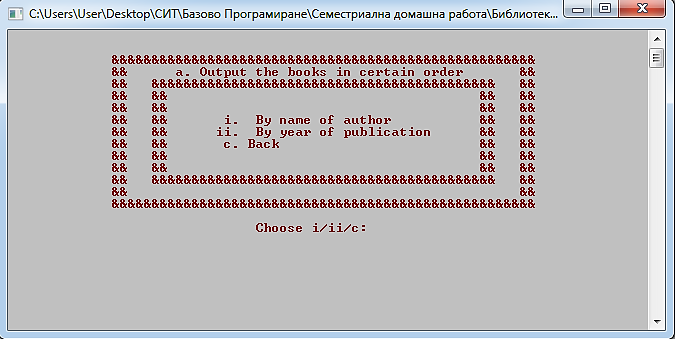
**Изглед при избор на опция b. от под меню E**

****

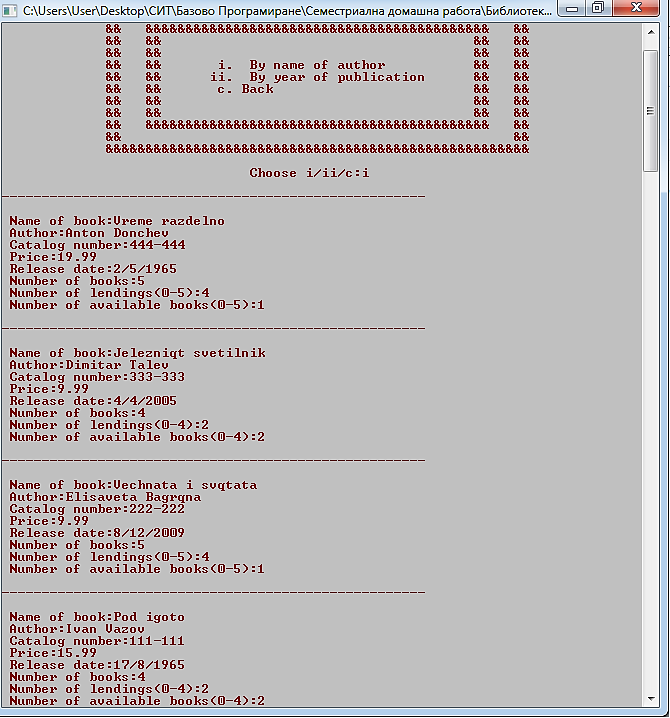
**Изглед при избор на под меню F**

****

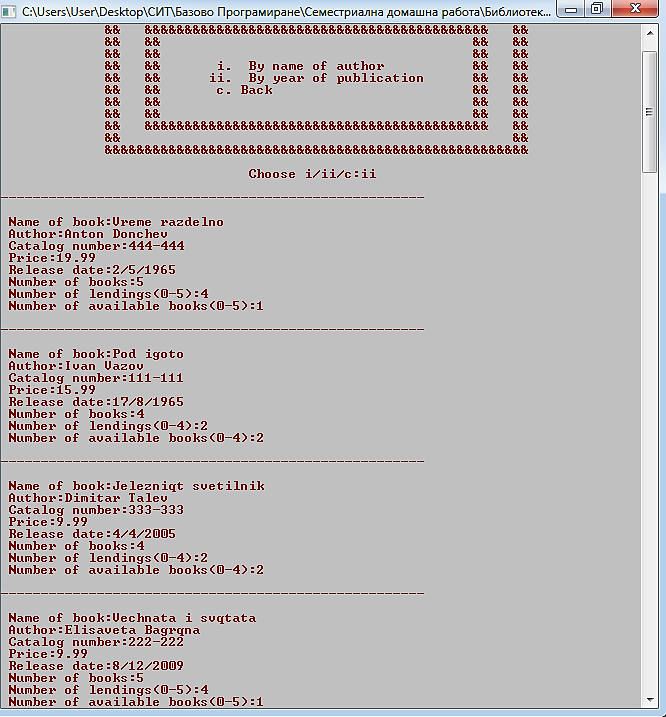
**Изглед при избор на под меню a. от под меню F**

****

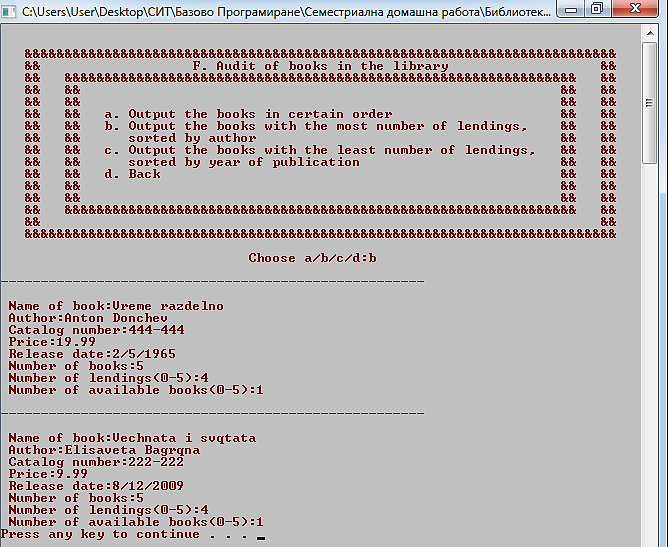
**Изглед при избор на опция i. от под меню a. от под меню F**

****

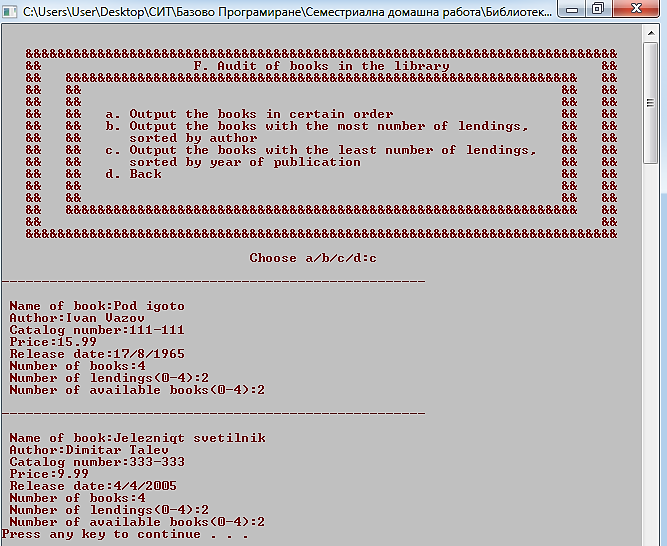
**Изглед при избор на опция ii. от под меню a. от под меню F**

****

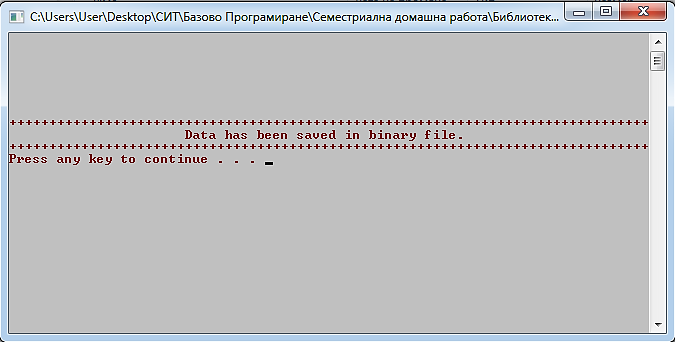
**Изглед при избор на опция b. от под меню F**

****

**Изглед при избор на опция c. от под меню F**

****

**Изглед при избор на опция G. от главното меню**

****

**Ако се изберe опция I. от главното меню тогава потребителя излиза от програмата.**

**Забележка :**

* Точка **а.** от Допълнение Четвърто е изпълнена.
* Точка **c.** от Допълнение Четвърто е изпълнена.