“Технически университет” – град Варна

Факултет: Факултет по изчислителна техника и автоматизация

Катедра: Компютърни науки и технологии  
Специалност: Компютърни системи и технологии  
Дисциплина: Базово програмиране

Тема: Информационна система на библиотека

Изготвил: Беркджан Хълми

ФН: 20621406

Пълно задание на проекта:

Библиотека

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

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

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.

b. Ако книгата не присъства в колекцията на библиотеката трябва да се изведе пояснително

съобщение

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

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.

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.

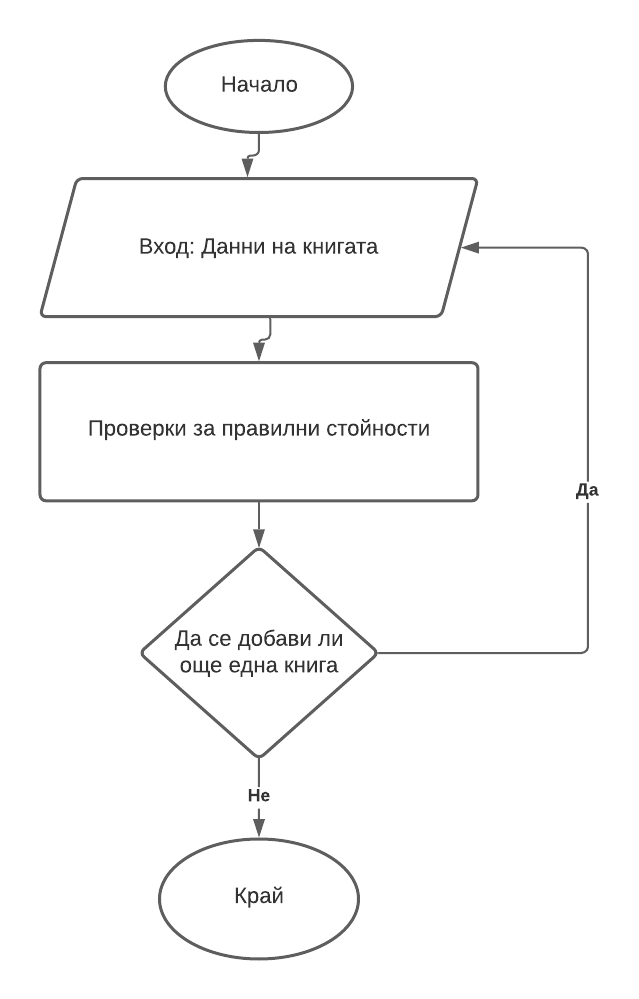
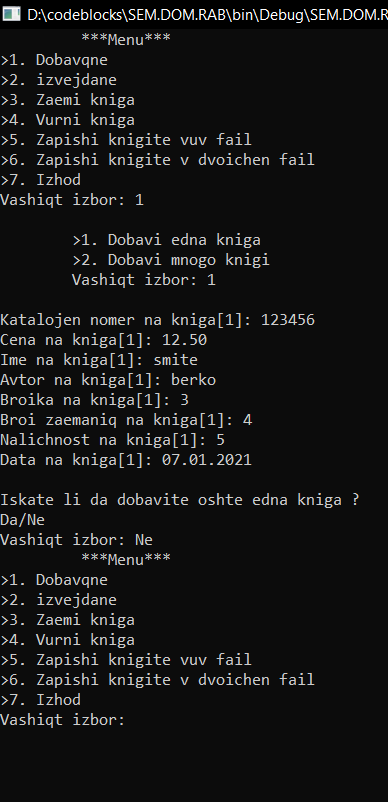
c. За точка G да се записват данните в двоичен файл.

Анализ на решението

Функции:

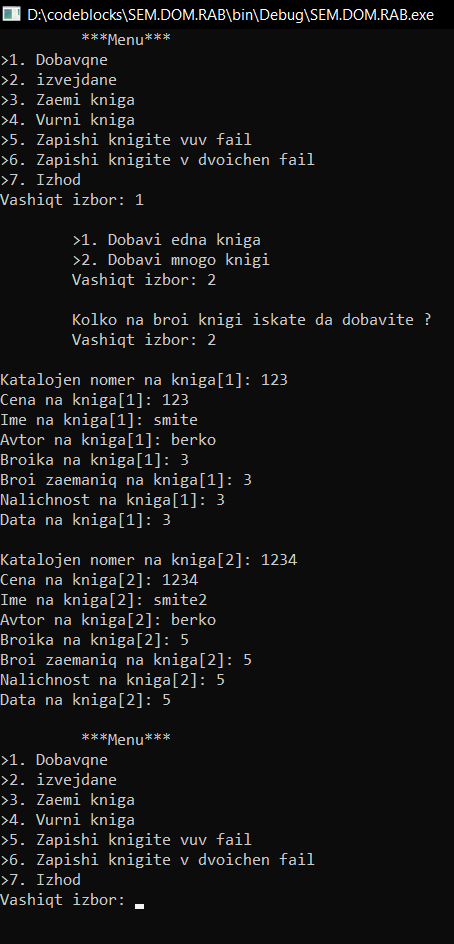
1. (B.a.) void DobaviKniga(Book kniga[100], int& Br)

Изисква се да се въведат конкретните данни на книгата и се записват в масива от структура.



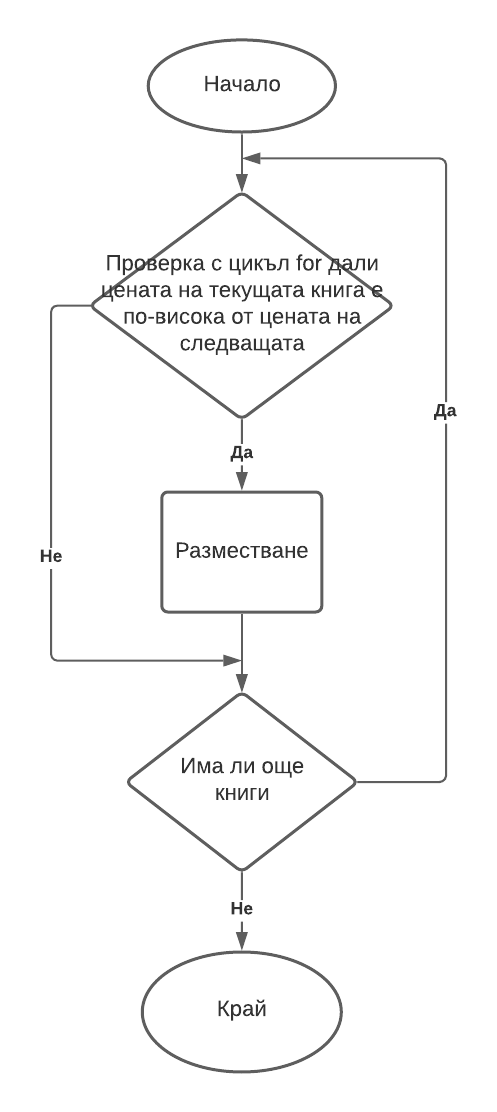
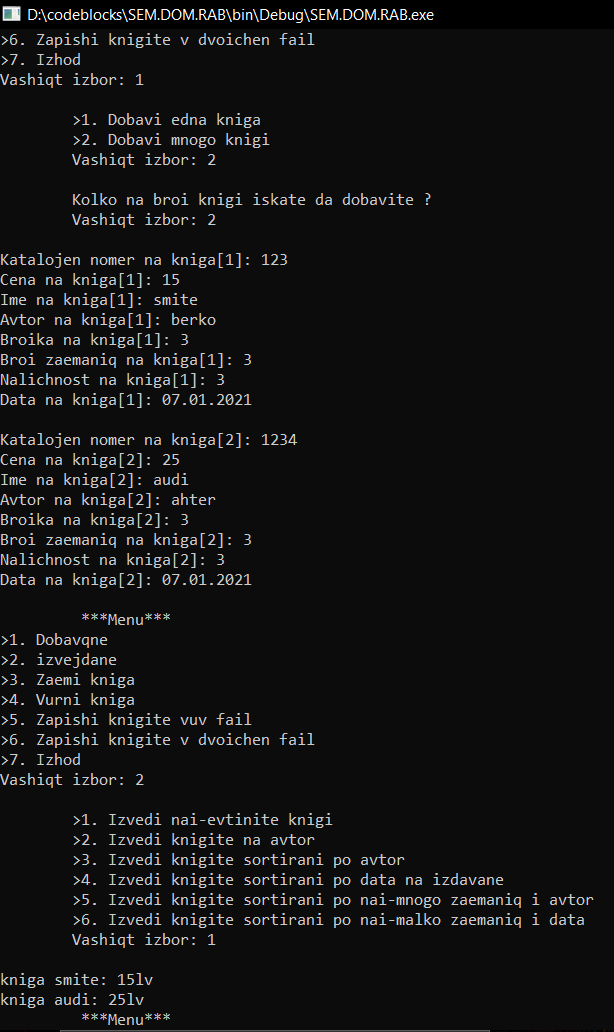
1. (B.b.) void DobaviMnogoKnigi(Book kniga[100], int& Br, int N)

Извиква се същата функция за добавяне на книга обаче се изпълнява толкова пъти, колкото е зададено от потребителя.



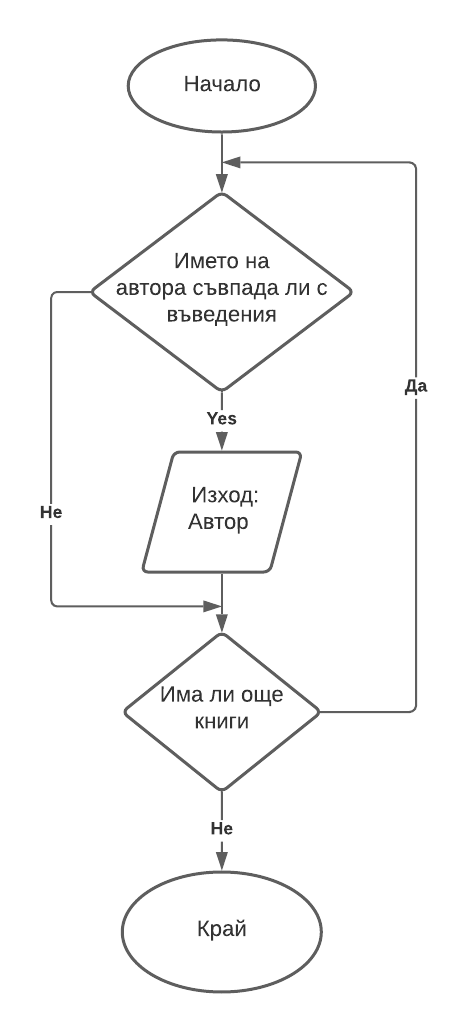
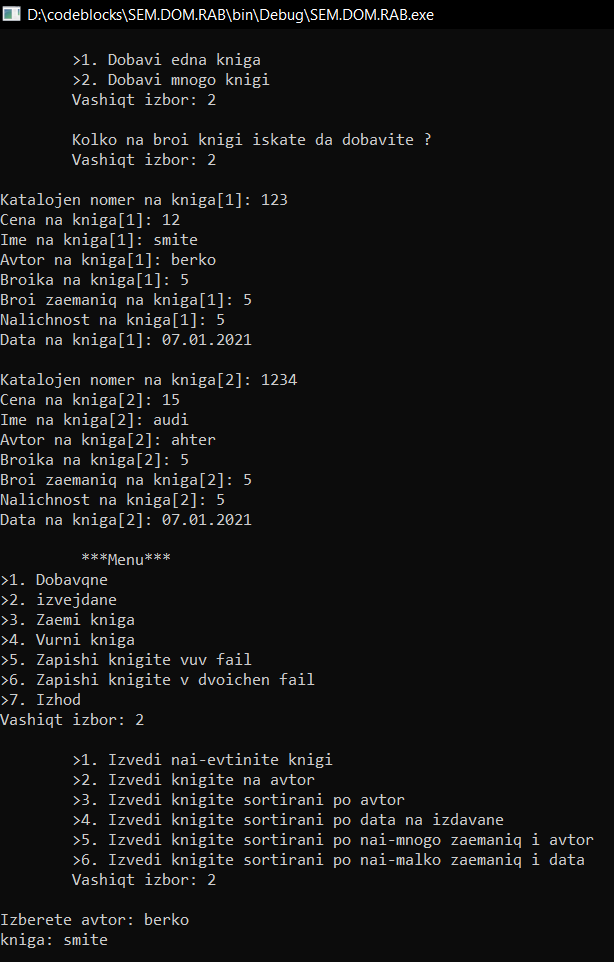
1. (C.a.) void BubbleSort(Book kniga[100], int& Br, float&, MinCena, string& MinIme)

Използва се начина на мехурчето за да се подредят най-евтините книги, най-горе. Прави се проверка на текущата книга със следващата и ако цената й е по-висока, се разменят местата им, така по-евтината изплува нагоре.



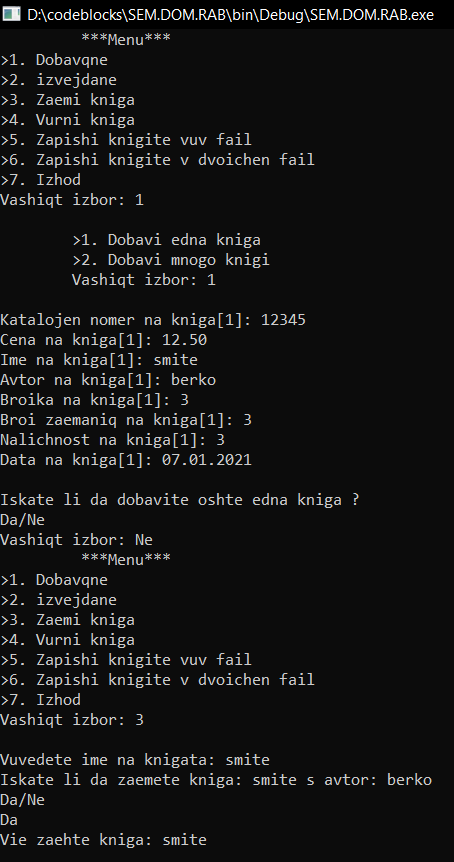
1. (C.b.) int IzvediPoAvtor(Book kniga[100], int Br, string ImeNaAvtor)

Въвежда се име на автор и му се извеждат всичките книги



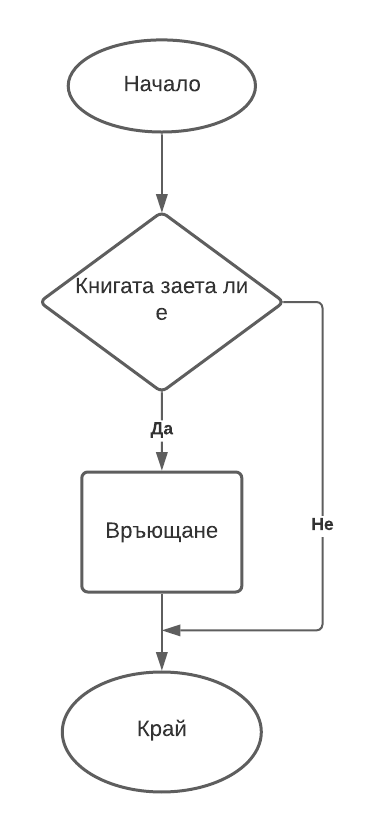
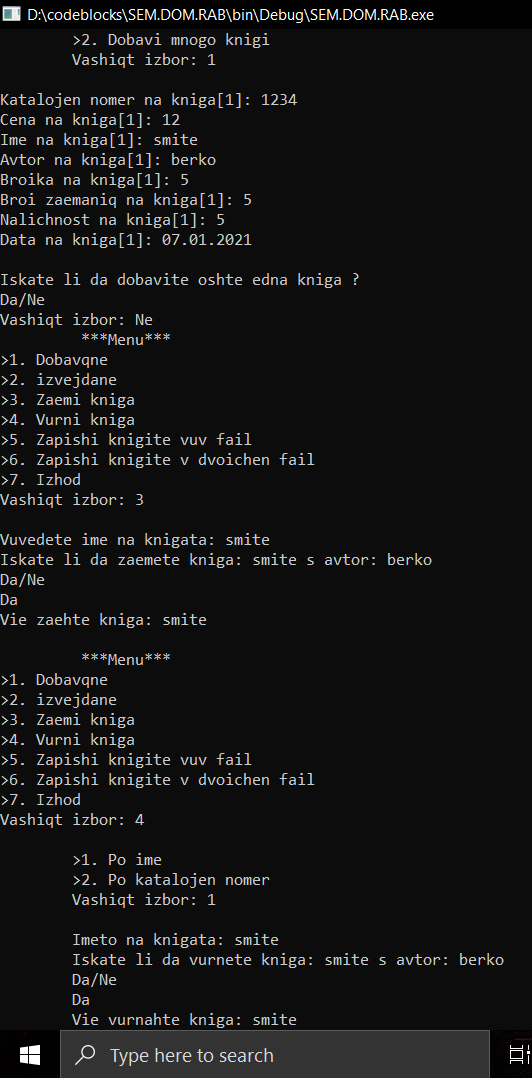
1. (D.a.) int ZaemiKniga(Book kniga[100], int& Br, string ImeNaKniga, string choice)

Въвежда се име на книга и ако то съвпада с някое от масива, се проверява дали тази книга е заета. Ако не е заета, се заема, ако е заета, се извежда подходящо съобщение.



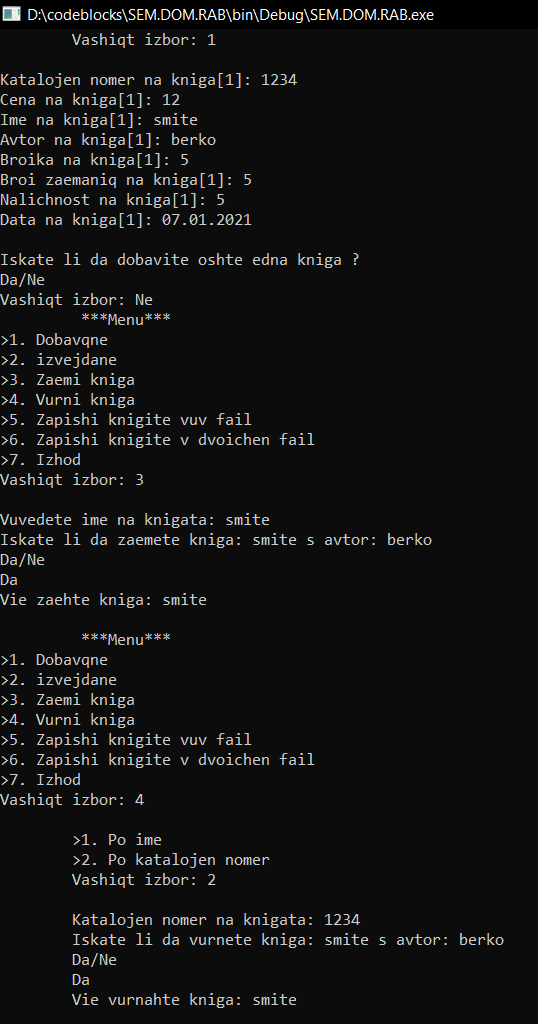
1. (E.a) int VurniKnigaPoIme(Book kniga[100], int& Br, string ImeNaKniga, string choice)

Въвежда се име на книга и ако то съвпада с някое от масива, се проверява дали тази книга е заемана, ако е, се връща. Ако не е заемана се извежда подходящо съобщение.



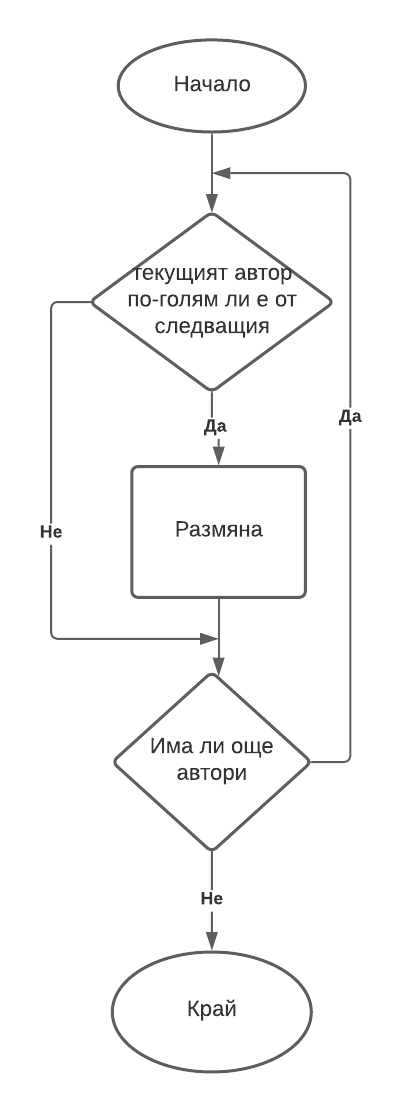
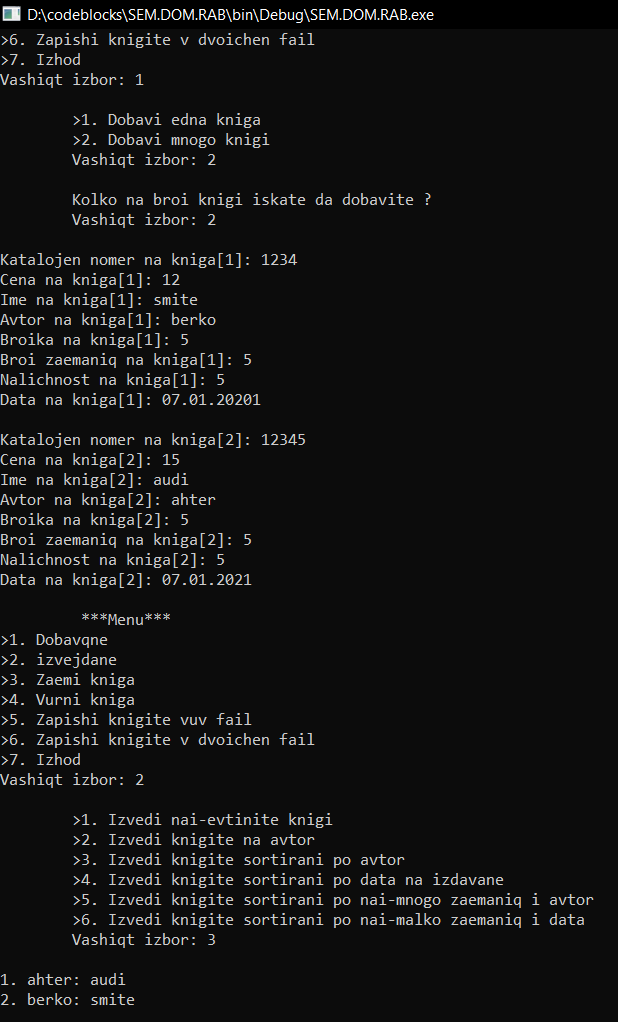
1. (E.b.) int VurniKnigaPoNomer(Book kniga[100], int& Br, int NomerNaKniga, string choice)

Въвежда се каталожен номер на книга и ако то съвпада с някое от масива, се проверява дали тази книга е заемана, ако е, се връща. Ако не е заемана се извежда подходящо съобщение.



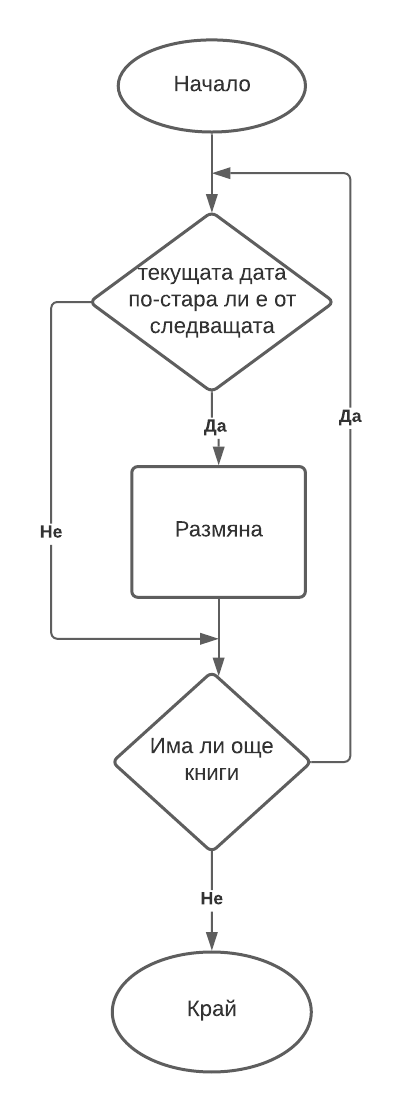
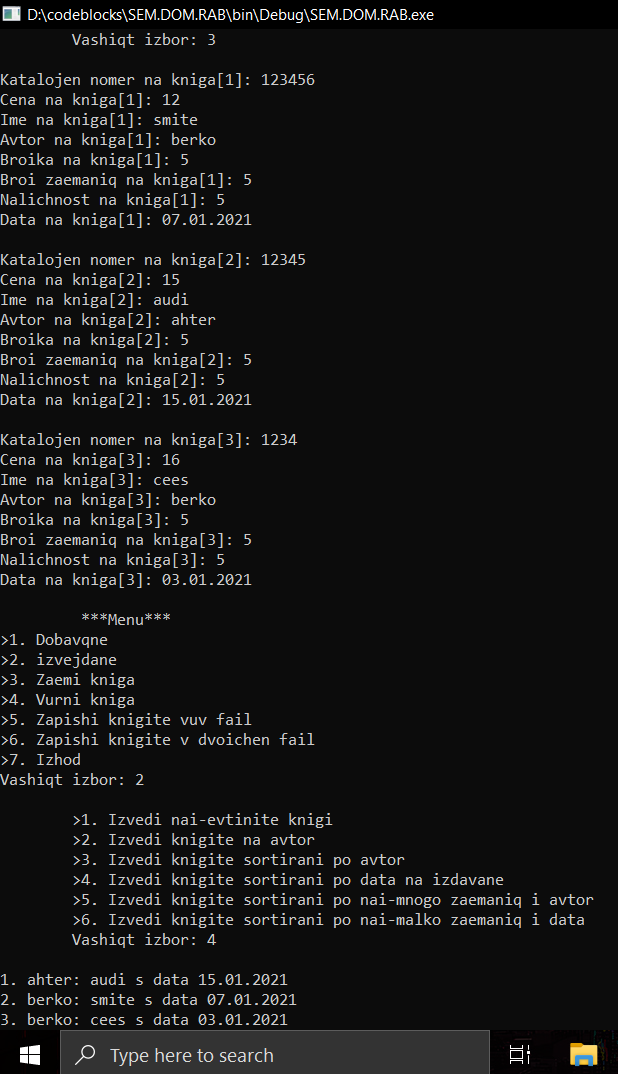
1. (F.a.i) void SortPoAvtor(Book kniga[100], int& Br)

Прави се проверка на текущия автор със следващия, името на автора се проверява буква по буква и този, който е най-първичен, се извежда пръв.



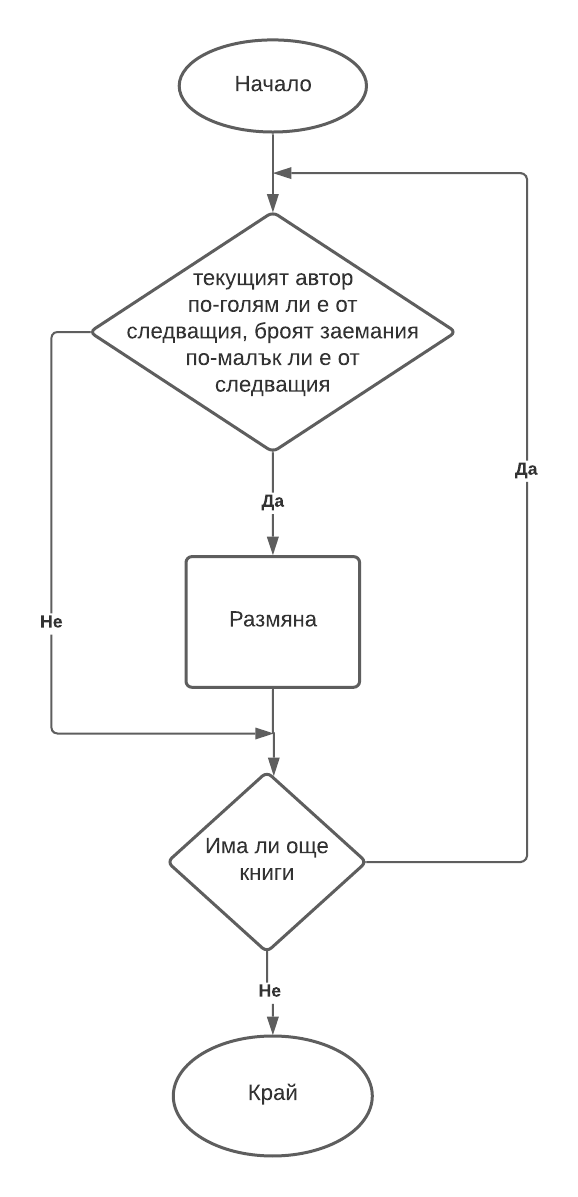
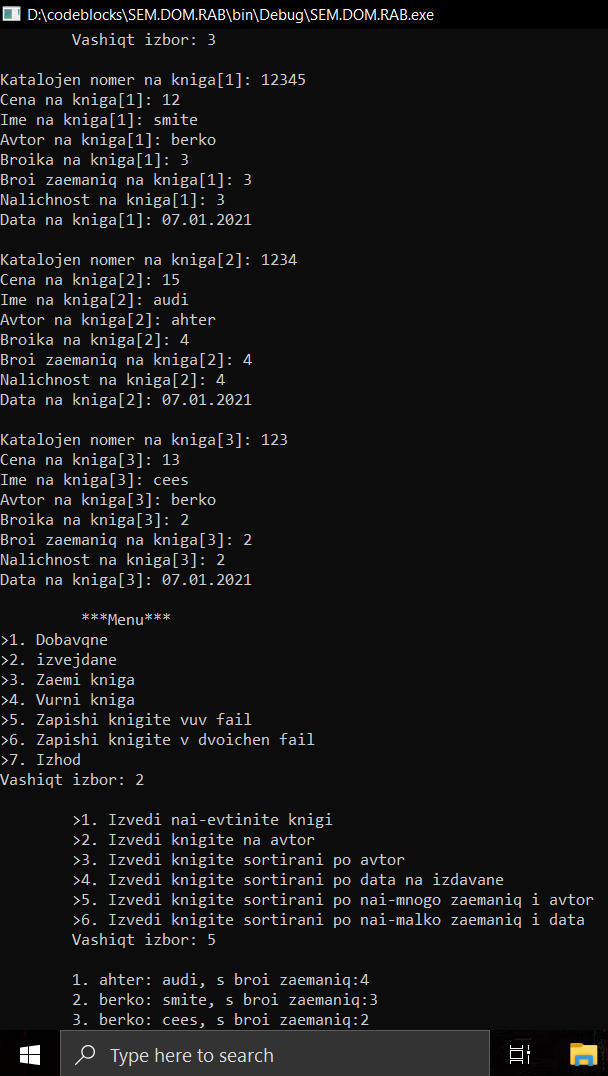
1. (F.a.ii) void SortPoData(Book kniga[100], int& Br)

Всяка книга се проверява една по една и с помощта на bubblesort, най-новата се изкачва отгоре.



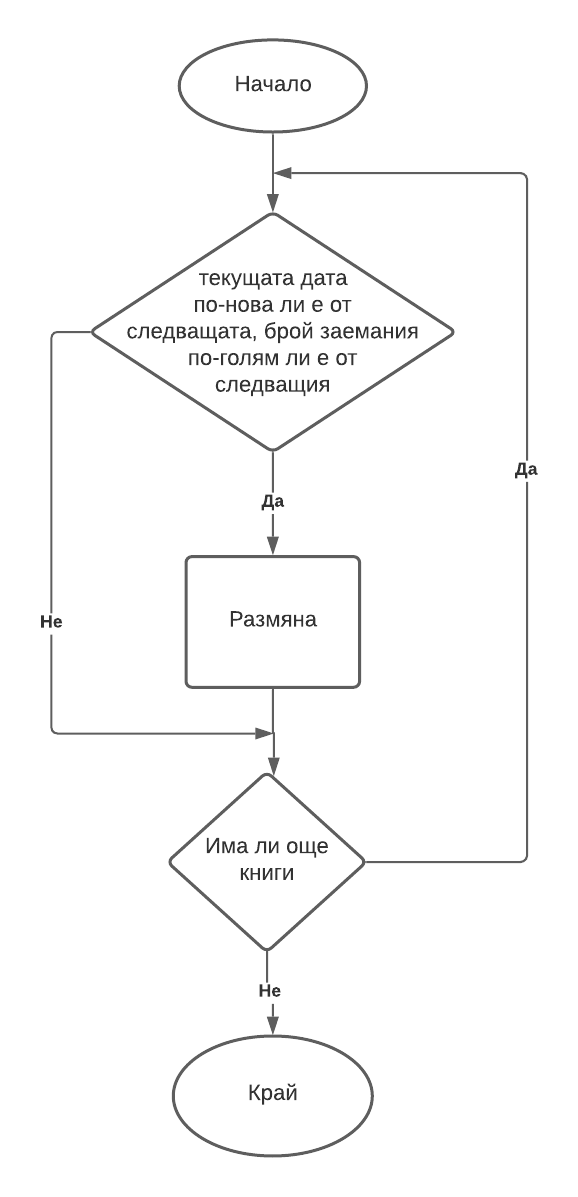
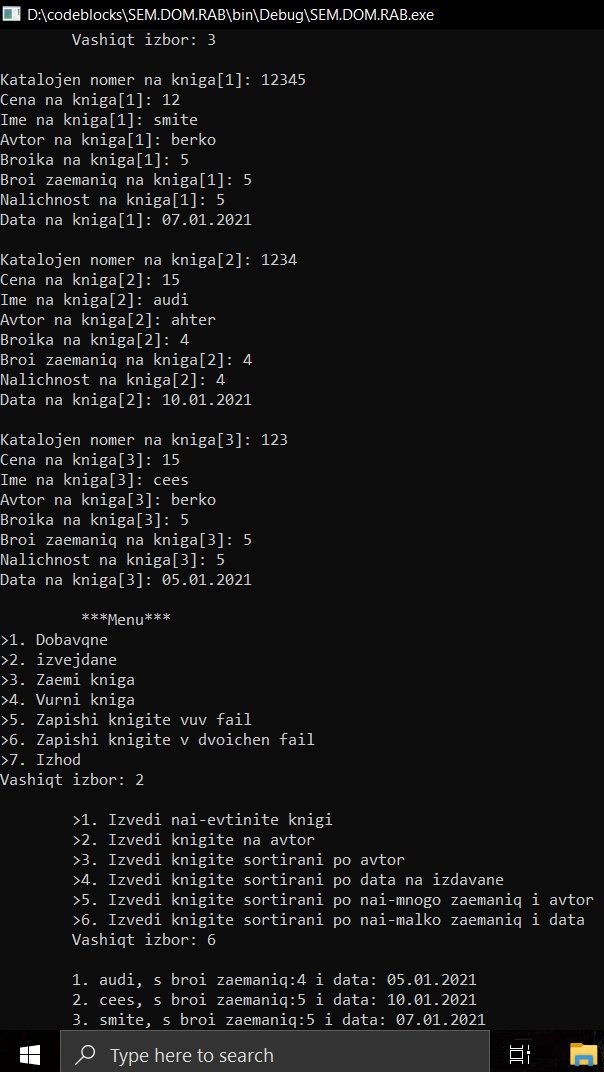
1. (F.b) void SortPoBrZaemaniqIAvtor(Book kniga[100], int& Br)

Първо се сортира масива по автор по низходящ ред и след това се проверяват книгите на този автор, съответно книгите се подреждат по брой заемания.



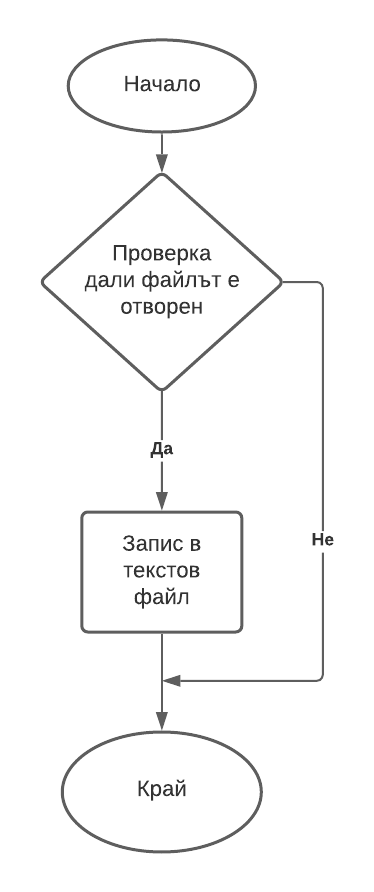
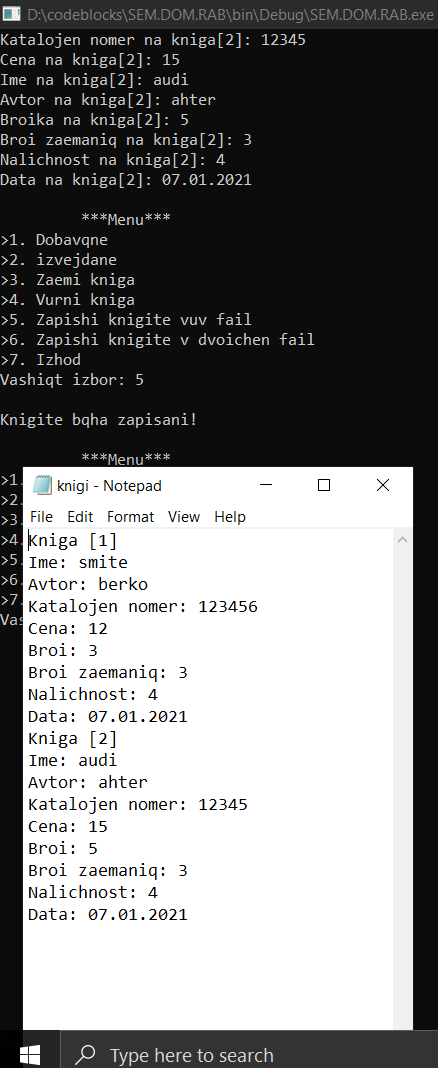
1. (F.c.) void SortPoBrZaemaniqIData(Book kniga[100], int& Br)

Първо се сортира масива по брой заемания в низходящ ред и след това им се проверяват датите, съответно книгите се подреждат по дата.



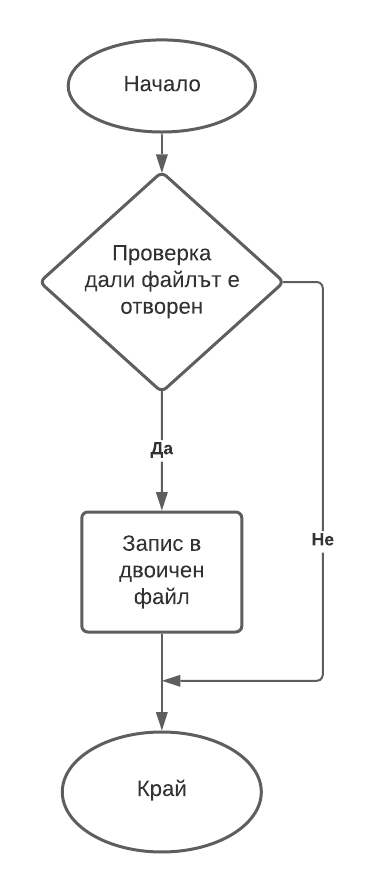
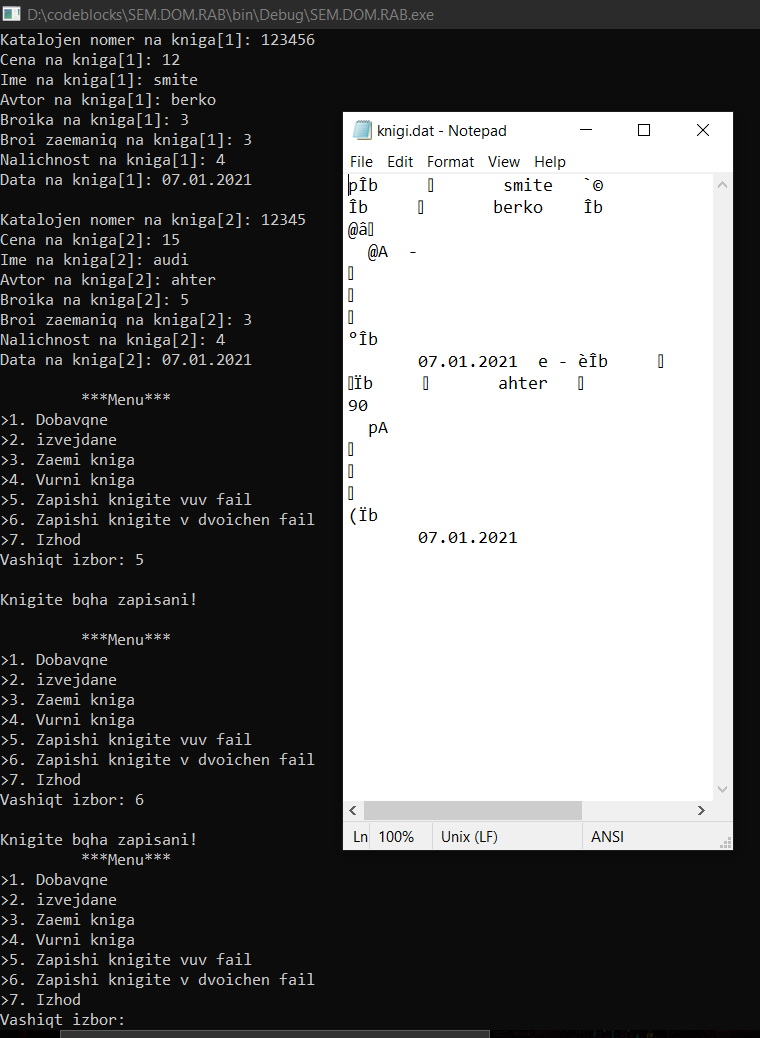
1. (G) void CreateFile(Book kniga[100],int& Br)

Първо се прави проверка дали създаването на файла е успешно и ако е, се записват данните на книгите в текстов файл. Ако отварянето на файла не е успешно, се извежда подходящо съобщение.



1. (H.c) void CreateBinaryFile(Book kniga[100], int& Br)

Първо се прави проверка дали създаването на файла е успешно и ако е, се записват данните на книгите в двоичен файл. Ако отварянето на файла не е успешно, се извежда подходящо съобщение.



Кода на програмата:

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

int i = 0;

struct Book

{

string Ime, Avtor, Data;

int K\_nomer = 0, Br\_knigi = 0, Br\_zaemaniq = 0, Nalichnost = 0;

float Cena = 0;

bool Zaet = false;

};

void DobaviKniga(Book kniga[100], int& Br)

{

cout << "Katalojen nomer na kniga[" << i + 1 << "]: ";

cin >> kniga[i].K\_nomer;

while (kniga[i].K\_nomer <= 0)

{

cout << "Molq vuvedete pravilen katalojen nomer \n";

cout << "Katalojen nomer na kniga[" << i + 1 << "]: ";

cin >> kniga[i].K\_nomer;

}

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

{

while (kniga[i].K\_nomer == kniga[k].K\_nomer)

{

cout << "Kniga s tozi katalojen nomer veche sushtestvuva \n\n";

cout << "Katalojen nomer na kniga[" << i + 1 << "]: ";

cin >> kniga[i].K\_nomer;

}

}

cout << "Cena na kniga[" << i + 1 << "]: ";

cin >> kniga[i].Cena;

cin.ignore();

cout << "Ime na kniga[" << i + 1 << "]: ";

getline(cin, kniga[i].Ime);

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

{

while (kniga[i].Ime == kniga[k].Ime)

{

cout << "Kniga s tova ime veche sushtestvuva \n\n";

cout << "Ime na kniga[" << i + 1 << "]: ";

getline(cin, kniga[i].Ime);

}

}

cout << "Avtor na kniga[" << i + 1 << "]: ";

getline(cin, kniga[i].Avtor);

cout << "Broika na kniga[" << i + 1 << "]: ";

cin >> kniga[i].Br\_knigi;

cout << "Broi zaemaniq na kniga[" << i + 1 << "]: ";

cin >> kniga[i].Br\_zaemaniq;

cout << "Nalichnost na kniga[" << i + 1 << "]: ";

cin >> kniga[i].Nalichnost;

while (kniga[i].Nalichnost < 0 || kniga[i].Nalichnost > 5)

{

cout << "Nalichnostta trqbwa da bude mejdu 0 i 5 \n";

cout << "Molq vuvedete nalichnost na knigata: ";

cin >> kniga[i].Nalichnost;

}

cin.ignore();

cout << "Data na kniga[" << i + 1 << "]: ";

getline(cin, kniga[i].Data);

cout << "\n";

Br++;

i++;

}

void DobaviMnogoKnigi(Book kniga[100], int& Br, int N)

{

cout << "\tKolko na broi knigi iskate da dobavite ?\n";

cout << "\tVashiqt izbor: ";

cin >> N; cout << "\n";

while (N <= 0 || N >= 100)

{

cout << "Molq vuvedete broi mejdu 1 i 100 \n";

cout << "Vashiqt izbor: ";

cin >> N;

}

for (int k = 0; k < N; k++)

{

DobaviKniga(kniga, Br);

}

}

void BubbleSort(Book kniga[100], int& Br, float& MinCena, string& MinIme)

{

for (int l = Br; l >= 1; l--)

{

for (int m = 0; m < l - 1; m++)

{

if (kniga[m].Cena > kniga[m + 1].Cena)

{

MinCena = kniga[m].Cena;

kniga[m].Cena = kniga[m + 1].Cena;

kniga[m + 1].Cena = MinCena;

MinIme = kniga[m].Ime;

kniga[m].Ime = kniga[m + 1].Ime;

kniga[m + 1].Ime = MinIme;

}

}

}

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

{

cout << "kniga " << kniga[n].Ime << ": " << kniga[n].Cena << "lv \n";

}

}

void SortPoAvtor(Book kniga[100], int& Br)

{

string Avtor, Kniga;

for (int l = Br; l > 1; l--)

{

for (int k = 0; k < l - 1; k++)

{

if (kniga[k].Avtor > kniga[k + 1].Avtor)

{

Avtor = kniga[k].Avtor;

kniga[k].Avtor = kniga[k + 1].Avtor;

kniga[k + 1].Avtor = Avtor;

Kniga = kniga[k].Ime;

kniga[k].Ime = kniga[k + 1].Ime;

kniga[k + 1].Ime = Kniga;

}

}

}

int nomer = 1;

for (int k = 0; k < Br; k++)

{

cout << nomer << ". " << kniga[k].Avtor << ": " << kniga[k].Ime << "\n";

nomer++;

}

cout << "\n";

}

void SortPoData(Book kniga[100], int& Br)

{

string Avtor, Kniga, Data, d, dd, m, mm, y, yy;

for (int l = Br; l > 1; l--)

{

for (int k = 0; k < l - 1; k++)

{

d = kniga[k].Data.substr(0, 2);

dd = kniga[k + 1].Data.substr(0, 2);

m = kniga[k].Data.substr(4, 6);

mm = kniga[k + 1].Data.substr(4, 6);

y = kniga[k].Data.substr(7, 10);

yy = kniga[k + 1].Data.substr(7, 10);

if (y > yy)

{

}

else if (y == yy)

{

if (m > mm)

{

}

else if (m == mm)

{

if (d > dd)

{

}

else if (d == dd)

{

}

else

{

Avtor = kniga[k].Avtor;

kniga[k].Avtor = kniga[k + 1].Avtor;

kniga[k + 1].Avtor = Avtor;

Kniga = kniga[k].Ime;

kniga[k].Ime = kniga[k + 1].Ime;

kniga[k + 1].Ime = Kniga;

Data = kniga[k].Data;

kniga[k].Data = kniga[k + 1].Data;

kniga[k + 1].Data = Data;

}

}

else

{

Avtor = kniga[k].Avtor;

kniga[k].Avtor = kniga[k + 1].Avtor;

kniga[k + 1].Avtor = Avtor;

Kniga = kniga[k].Ime;

kniga[k].Ime = kniga[k + 1].Ime;

kniga[k + 1].Ime = Kniga;

Data = kniga[k].Data;

kniga[k].Data = kniga[k + 1].Data;

kniga[k + 1].Data = Data;

}

}

else

{

Avtor = kniga[k].Avtor;

kniga[k].Avtor = kniga[k + 1].Avtor;

kniga[k + 1].Avtor = Avtor;

Kniga = kniga[k].Ime;

kniga[k].Ime = kniga[k + 1].Ime;

kniga[k + 1].Ime = Kniga;

Data = kniga[k].Data;

kniga[k].Data = kniga[k + 1].Data;

kniga[k + 1].Data = Data;

}

}

}

int nomer = 1;

for (int k = 0; k < Br; k++)

{

cout << nomer << ". " << kniga[k].Avtor << ": " << kniga[k].Ime << " s data " << kniga[k].Data << "\n";

nomer++;

}

cout << "\n";

}

void SortPoBrZaemaniqIAvtor(Book kniga[100], int& Br)

{

string Avtor, Kniga, Ime;

int Br\_Avtor = 0, Br\_Zaemaniq = 0;

for (int l = Br; l > 1; l--)

{

for (int k = 0; k < l - 1; k++)

{

if (kniga[k].Avtor > kniga[k+1].Avtor)

{

Ime = kniga[k].Avtor;

kniga[k].Avtor = kniga[k + 1].Avtor;

kniga[k + 1].Avtor = Ime;

Br\_Zaemaniq = kniga[k].Br\_zaemaniq;

kniga[k].Br\_zaemaniq = kniga[k + 1].Br\_zaemaniq;

kniga[k + 1].Br\_zaemaniq = Br\_Zaemaniq;

Kniga = kniga[k].Ime;

kniga[k].Ime = kniga[k + 1].Ime;

kniga[k + 1].Ime = Kniga;

}

}

}

for (int l = Br; l > 1; l--)

{

for (int k = 0; k < l - 1; k++)

{

if (kniga[k].Avtor > kniga[k+1].Avtor && kniga[k].Br\_zaemaniq < kniga[k + 1].Br\_zaemaniq)

{

Ime = kniga[k].Avtor;

kniga[k].Avtor = kniga[k + 1].Avtor;

kniga[k + 1].Avtor = Ime;

Br\_Zaemaniq = kniga[k].Br\_zaemaniq;

kniga[k].Br\_zaemaniq = kniga[k + 1].Br\_zaemaniq;

kniga[k + 1].Br\_zaemaniq = Br\_Zaemaniq;

Kniga = kniga[k].Ime;

kniga[k].Ime = kniga[k + 1].Ime;

kniga[k + 1].Ime = Kniga;

}

}

}

int nomer = 1;

for (int k = 0; k < Br; k++)

{

if (Br > 0)

{

cout << "\t" << nomer << ". " << kniga[k].Avtor << ": " << kniga[k].Ime << ", s broi zaemaniq:" << kniga[k].Br\_zaemaniq << "\n";

nomer++;

}

else

{

cout << "\t Nqma takuv avtor";

break;

}

}

Br\_Avtor = 0;

cout << "\n";

}

void SortPoBrZaemaniqIData(Book kniga[100], int& Br)

{

string Avtor, Kniga, Ime, Data, d, dd, m, mm, y, yy;

int Br\_Avtor = 0, Br\_Zaemaniq = 0;

for (int l = Br; l > 1; l--)

{

for (int k = 0; k < l - 1; k++)

{

if (kniga[k].Br\_zaemaniq < kniga[k + 1].Br\_zaemaniq)

{

Ime = kniga[k].Avtor;

kniga[k].Avtor = kniga[k + 1].Avtor;

kniga[k + 1].Avtor = Ime;

Br\_Zaemaniq = kniga[k].Br\_zaemaniq;

kniga[k].Br\_zaemaniq = kniga[k + 1].Br\_zaemaniq;

kniga[k + 1].Br\_zaemaniq = Br\_Zaemaniq;

Kniga = kniga[k].Ime;

kniga[k].Ime = kniga[k + 1].Ime;

kniga[k + 1].Ime = Kniga;

}

}

}

for (int l = Br; l > 1; l--)

{

for (int k = 0; k < l - 1; k++)

{

d = kniga[k].Data.substr(0, 2);

dd = kniga[k + 1].Data.substr(0, 2);

m = kniga[k].Data.substr(4, 6);

mm = kniga[k + 1].Data.substr(4, 6);

y = kniga[k].Data.substr(7, 10);

yy = kniga[k + 1].Data.substr(7, 10);

if (y > yy && kniga[k].Br\_zaemaniq < kniga[k + 1].Br\_zaemaniq)

{

}

else if (y == yy && kniga[k].Br\_zaemaniq < kniga[k + 1].Br\_zaemaniq)

{

if (m > mm && kniga[k].Br\_zaemaniq < kniga[k + 1].Br\_zaemaniq)

{

}

else if (m == mm && kniga[k].Br\_zaemaniq < kniga[k + 1].Br\_zaemaniq)

{

if (d > dd && kniga[k].Br\_zaemaniq < kniga[k + 1].Br\_zaemaniq)

{

}

else if (d == dd && kniga[k].Br\_zaemaniq < kniga[k + 1].Br\_zaemaniq)

{

}

else

{

Avtor = kniga[k].Avtor;

kniga[k].Avtor = kniga[k + 1].Avtor;

kniga[k + 1].Avtor = Avtor;

Kniga = kniga[k].Ime;

kniga[k].Ime = kniga[k + 1].Ime;

kniga[k + 1].Ime = Kniga;

Data = kniga[k].Data;

kniga[k].Data = kniga[k + 1].Data;

kniga[k + 1].Data = Data;

Br\_Zaemaniq = kniga[k].Br\_zaemaniq;

kniga[k].Br\_zaemaniq = kniga[k + 1].Br\_zaemaniq;

kniga[k + 1].Br\_zaemaniq = Br\_Zaemaniq;

}

}

else

{

Avtor = kniga[k].Avtor;

kniga[k].Avtor = kniga[k + 1].Avtor;

kniga[k + 1].Avtor = Avtor;

Kniga = kniga[k].Ime;

kniga[k].Ime = kniga[k + 1].Ime;

kniga[k + 1].Ime = Kniga;

Data = kniga[k].Data;

kniga[k].Data = kniga[k + 1].Data;

kniga[k + 1].Data = Data;

Br\_Zaemaniq = kniga[k].Br\_zaemaniq;

kniga[k].Br\_zaemaniq = kniga[k + 1].Br\_zaemaniq;

kniga[k + 1].Br\_zaemaniq = Br\_Zaemaniq;

}

}

else

{

Avtor = kniga[k].Avtor;

kniga[k].Avtor = kniga[k + 1].Avtor;

kniga[k + 1].Avtor = Avtor;

Kniga = kniga[k].Ime;

kniga[k].Ime = kniga[k + 1].Ime;

kniga[k + 1].Ime = Kniga;

Data = kniga[k].Data;

kniga[k].Data = kniga[k + 1].Data;

kniga[k + 1].Data = Data;

Br\_Zaemaniq = kniga[k].Br\_zaemaniq;

kniga[k].Br\_zaemaniq = kniga[k + 1].Br\_zaemaniq;

kniga[k + 1].Br\_zaemaniq = Br\_Zaemaniq;

}

}

}

int nomer = 1;

for (int k = 0; k < Br; k++)

{

if (Br > 0)

{

cout << "\t" << nomer << ". " << kniga[k].Ime << ", s broi zaemaniq:" << kniga[k].Br\_zaemaniq << " i data: " << kniga[k].Data << "\n";

nomer++;

}

else

{

cout << "\t Nqma takuv avtor";

break;

}

}

Br\_Avtor = 0;

cout << "\n";

}

int IzvediPoAvtor(Book kniga[100], int Br, string ImeNaAvtor)

{

int Br\_Avtor = 0;

for (int k = 0; k < Br; k++)

{

if (ImeNaAvtor == kniga[k].Avtor)

{

cout << "kniga: " << kniga[k].Ime << "\n";

Br\_Avtor++;

}

}

cout << "\n";

return Br\_Avtor;

}

int ZaemiKniga(Book kniga[100], int& Br, string ImeNaKniga, string choice)

{

int Br\_Knigi = 0;

cout << "Vuvedete ime na knigata: ";

cin.ignore();

getline(cin, ImeNaKniga);

for (int k = 0; k < Br; k++)

{

if (ImeNaKniga == kniga[k].Ime && kniga[k].Zaet == false)

{

Br\_Knigi++;

if (kniga[k].Nalichnost > 0)

{

cout << "Iskate li da zaemete kniga: " << kniga[k].Ime << " s avtor: " << kniga[k].Avtor << "\n";

cout << "Da/Ne \n";

LOOP:getline(cin, choice);

if (choice == "Da" || choice == "da" || choice == "dA" || choice == "DA")

{

cout << "Vie zaehte kniga: " << kniga[k].Ime << "\n\n";

kniga[k].Br\_zaemaniq++;

kniga[k].Nalichnost--;

kniga[k].Zaet = true;

break;

}

else if (choice == "Ne" || choice == "ne" || choice == "nE" || choice == "NE")

{

break;

}

else

{

cout << "Molq izberete da ili ne \n";

goto LOOP;

}

}

else cout << "Tazi kniga ne e nalichna \n\n";

}

}

choice.clear();

return Br\_Knigi;

}

int VurniKnigaPoIme(Book kniga[100], int& Br, string ImeNaKniga, string choice)

{

int Br\_Knigi = 0;

cout << "\tImeto na knigata: ";

cin.ignore();

getline(cin, ImeNaKniga);

for (int k = 0; k < Br; k++)

{

if (ImeNaKniga == kniga[k].Ime && kniga[k].Zaet == true)

{

Br\_Knigi++;

cout << "\tIskate li da vurnete kniga: " << kniga[k].Ime << " s avtor: " << kniga[k].Avtor << "\n";

cout << "\tDa/Ne \n";

cout << "\t";

LOOP:getline(cin, choice);

if (choice == "Da" || choice == "da" || choice == "dA" || choice == "DA")

{

cout << "\tVie vurnahte kniga: " << kniga[k].Ime << "\n\n";

kniga[k].Nalichnost++;

kniga[k].Zaet = false;

break;

}

else if (choice == "Ne" || choice == "ne" || choice == "nE" || choice == "NE")

{

break;

}

else

{

cout << "Molq izberete da ili ne \n";

goto LOOP;

}

}

else if (ImeNaKniga == kniga[k].Ime && kniga[k].Zaet == false)

{

cout << "\tNe ste zaeli tazi kniga \n\n";

Br\_Knigi++;

}

}

return Br\_Knigi;

}

int VurniKnigaPoNomer(Book kniga[100], int& Br, int NomerNaKniga, string choice)

{

int Br\_Knigi = 0;

cout << "\tKatalojen nomer na knigata: ";

cin >> NomerNaKniga;

for (int k = 0; k < Br; k++)

{

if (NomerNaKniga == kniga[k].K\_nomer && kniga[k].Zaet == true)

{

Br\_Knigi++;

cout << "\tIskate li da vurnete kniga: " << kniga[k].Ime << " s avtor: " << kniga[k].Avtor << "\n";

cout << "\tDa/Ne \n";

cout << "\t";

cin.ignore();

LOOP:getline(cin, choice);

if (choice == "Da" || choice == "da" || choice == "dA" || choice == "DA")

{

cout << "\tVie vurnahte kniga: " << kniga[k].Ime << "\n\n";

kniga[k].Nalichnost++;

kniga[k].Zaet = false;

break;

}

else if (choice == "Ne" || choice == "ne" || choice == "nE" || choice == "NE")

{

break;

}

else

{

cout << "Molq izberete da ili ne \n";

goto LOOP;

}

}

else if (NomerNaKniga == kniga[k].K\_nomer && kniga[k].Zaet == false)

{

cout << "\tNe ste zaeli tazi kniga \n\n";

Br\_Knigi++;

}

}

return Br\_Knigi;

}

void CreateFile(Book kniga[100],int& Br)

{

ofstream file;

if (Br == 0)

cout << " Nqma vavedeni knigi !\n";

else

{

file.open("knigi.txt", ios::out | ios::ate);

if (file.fail())

{

cout << "Faila ne moja da se sazdade !\n";

}

else

{

for (int k = 0; k < Br; k++)

{

file<<"Kniga ["<<k+1<<"]\n";

file<<"Ime: "<<kniga[k].Ime;

file<<"\nAvtor: "<<kniga[k].Avtor;

file<<"\nKatalojen nomer: "<<kniga[k].K\_nomer;

file<<"\nCena: "<<kniga[k].Cena;

file<<"\nBroi: "<<kniga[k].Br\_knigi;

file<<"\nBroi zaemaniq: "<<kniga[k].Br\_zaemaniq;

file<<"\nNalichnost: "<<kniga[k].Nalichnost;

file<<"\nData: "<<kniga[k].Data;

file<<endl;

}

cout<<"Knigite bqha zapisani!\n\n";

}

file.close();

}

}

void CreateBinaryFile(Book kniga[100], int& Br)

{

ofstream file;

if (Br == 0)

cout << " Nqma vavedeni knigi !\n";

else

{

file.open("knigi.dat",ios::binary | ios::out | ios::ate);

if (file.fail())

{

cout << "Faila ne moja da se sazdade !\n";

}

else

{

for (int k = 0; k < Br; k++)

{

file.write((char\*)&kniga[k].Ime,sizeof(string));

file<<endl;

file.write((char\*)&kniga[k].Avtor,sizeof(string));

file<<endl;

file.write((char\*)&kniga[k].K\_nomer,sizeof(int));

file<<endl;

file.write((char\*)&kniga[k].Cena,sizeof(double));

file<<endl;

file.write((char\*)&kniga[k].Br\_knigi,sizeof(int));

file<<endl;

file.write((char\*)&kniga[k].Br\_zaemaniq,sizeof(int));

file<<endl;

file.write((char\*)&kniga[k].Nalichnost,sizeof(int));

file<<endl;

file.write((char\*)&kniga[k].Data,sizeof(string));

}

cout<<"Knigite bqha zapisani!\n";

}

file.close();

}

}

int main()

{

Book kniga[100];

int decision = 0, N = 0, Br = 0, Br\_Avtor = 0, Br\_Knigi = 0, NomerNaKniga = 0;

float MinCena = 0;

string ImeNaAvtor, ImeNaKniga, MinIme, MinAvtor, choice, izbor;

do

{

cout << "\t \*\*\*Menu\*\*\* \t \n";

cout << ">1. Dobavqne \n";

cout << ">2. izvejdane \n";

cout << ">3. Zaemi kniga \n";

cout << ">4. Vurni kniga \n";

cout << ">5. Zapishi knigite vuv fail \n";

cout << ">6. Zapishi knigite v dvoichen fail \n";

cout << ">7. Izhod \n";

cout << "Vashiqt izbor: ";

cin >> decision; cout << "\n";

switch (decision)

{

case 1:

decision = 0;

cout << "\t>1. Dobavi edna kniga \n";

cout << "\t>2. Dobavi mnogo knigi \n";

cout << "\tVashiqt izbor: ";

cin >> decision; cout << "\n";

while (decision < 1 || decision>2)

{

cout << "\tMolq izberete 1 ili 2 \n";

cout << "\tVashiqt izbor: ";

cin >> decision; cout << "\n";

}

switch (decision)

{

case 1:

DobaviKniga(kniga, Br);

do

{

cout<<"Iskate li da dobavite oshte edna kniga ?\n";

cout<<"Da/Ne \n";

cout<<"Vashiqt izbor: ";

cin>>izbor;

if(izbor=="Da" || izbor=="da")

{

DobaviKniga(kniga, Br);

}

else

break;

}while(izbor=="Da");

decision = 0;

break;

case 2:

DobaviMnogoKnigi(kniga, Br, N);

decision = 0;

break;

}

break;

case 2:

decision = 0;

cout << "\t>1. Izvedi nai-evtinite knigi \n";

cout << "\t>2. Izvedi knigite na avtor \n";

cout << "\t>3. Izvedi knigite sortirani po avtor \n";

cout << "\t>4. Izvedi knigite sortirani po data na izdavane \n";

cout << "\t>5. Izvedi knigite sortirani po nai-mnogo zaemaniq i avtor \n";

cout << "\t>6. Izvedi knigite sortirani po nai-malko zaemaniq i data \n";

cout << "\tVashiqt izbor: ";

cin >> decision; cout << "\n";

while (decision < 1 || decision>6)

{

cout << "\t\nMolq proverete izbora si \n";

cout << "\tVashiqt izbor: ";

cin >> decision; cout << "\n";

}

switch (decision)

{

case 1:

BubbleSort(kniga, Br, MinCena, MinIme);

decision = 0;

break;

case 2:

cin.ignore();

cout << "\tIzberete avtor: ";

getline(cin, ImeNaAvtor);

Br\_Avtor = IzvediPoAvtor(kniga, Br, ImeNaAvtor);

if (Br\_Avtor == 0)

cout << "Nqma takuv avtor \n\n";

decision = 0;

break;

case 3:

SortPoAvtor(kniga, Br);

decision = 0;

break;

case 4:

SortPoData(kniga, Br);

decision = 0;

break;

case 5:

SortPoBrZaemaniqIAvtor(kniga, Br);

decision = 0;

break;

case 6:

SortPoBrZaemaniqIData(kniga, Br);

decision = 0;

break;

}

break;

case 3:

decision = 0;

Br\_Knigi = ZaemiKniga(kniga, Br, ImeNaKniga, choice);

if (Br\_Knigi == 0)

cout << "Nqma takava kniga \n\n";

decision = 0;

break;

case 4:

decision = 0;

cout << "\t>1. Po ime \n";

cout << "\t>2. Po katalojen nomer \n";

cout << "\tVashiqt izbor: ";

cin >> decision; cout << "\n";

while (decision < 1 || decision>2)

{

cout << "\t\nMolq izberete 1 ili 2 \n";

cout << "\tVashiqt izbor: ";

cin >> decision; cout << "\n";

}

switch (decision)

{

case 1:

Br\_Knigi = VurniKnigaPoIme(kniga, Br, ImeNaKniga, choice);

if (Br\_Knigi == 0)

cout << "\tNqma takava kniga \n\n";

decision = 0;

break;

case 2:

Br\_Knigi = VurniKnigaPoNomer(kniga, Br, NomerNaKniga, choice);

if (Br\_Knigi == 0)

cout << "\tNqma takava kniga \n\n";

decision = 0;

break;

}

break;

case 5:

CreateFile(kniga,Br);

decision=0;

break;

case 6:

CreateBinaryFile(kniga,Br);

break;

case 7:

cout << "Krai na programata \n";

break;

}

} while (decision !=7);

return 0;

}