ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра по компютърни науки и технологии

Компютърни системи и технологии

КУРСОВА РАБОТА

по

Обектно-Ориентирано Програмиране – Част 1

Изготвил:

Селин

Фак. №:

Група:

Зимен Семестър 2020/2021

Задание № 4

4-I. Да се състави клас Книга LibraryBook

Скрити член променливи: Име на книга- string; Автор – string; Заета/свободна - bool

Публични функции:

I.1. Конструктори - експлицитен (име), експлицитен (име, флаг – заета/свободна);

I.2. Заемане на книгата: takeBook()

I.3. Връщане на книга: returnBook()

I.4. Четене/запис

I.5. operator == сравнява по всички данни

I.6. operator< за сравнение за по-малко на подадения параметър обект с текущия по име на книгата

I.7. Преобразува / връща низ от членовете на класа

I.8. operator<< (извежда в указан изходен поток)

I.9. operator>> (чете от указан входен поток)

4-II. Да се състави клас Библиотека Library

Скрити член променливи: име на библиотеката - string m_strLibName;

брой стелажи; капацитет на стелаж;

multimap на разположение - двойки (стелаж, книга) - multimap<int,LibraryBook>;

map за търсене буква от име към номер стелаж - двойки (символ,стелаж) - map<char, int>

Публични функции:

II.1. Конструктори - експл. конструктор с име на файл. Формат на файла:

име_на_библиотека<SP>брой_стелажи<SP>капацитет_на_стелаж<SP>име_на_книга<SP>автор<SP>име_на_книга<SP>авто

р<EOF>

LibName 4 5 Book20 Aut20 Book12 Aut12 Book13 Aut13 Book11 Aut11 Book3 Aut3

Разпределя книгите по стелажите подреждайки ги по азбучен ред на имената на книгите, създава карта за търсене по буква

II.2. По зададен стелаж връща вектор от наличните книги

II.3. По зададен стелаж връща вектор от заети книги

II.4. Търси свободна книга по зададено име

II.5. Заема свободна книга по зададено име

II.6. Търси заета книга по зададено име

II.7. Връща книга по зададено име

4-III. Главна функция (main)

III.1. Създава обекти Library по експл. конструктор от файл

III.2. Създава вектор от свободни и заети книги

III.3. Извежда книгите по указан стелаж на конзолен изход

III.4. Заема книга по име, извежда на конзолен изход резултат

III.5. Връща книга, извежда на конзолен изход резултат

*Изискване - използване на алгоритми при решението

Обяснение

```
LibName
4
5
THE_GREAT_GATSBY FITZGERALD 0
THE_LITTLE_PRINCE ANTOINE 0
TO_KILL_A_MOCKINGBIRD LEE 1
ANIMAL_FARM ORWELL 0
PRIDE_AND_PREJUDICE AUSTEN 1
THE_GREAT_GATSBY2 FITZGERALD 1
THE_LITTLE_PRINCE2 ANTOINE 1
TO_KILL_A_MOCKINGBIRD2 LEE 1
ANIMAL_FARM_2 ORWELL 1
PRIDE_AND_PREJUDICE3 AUSTEN 1
THE_GREAT_GATSBY3 FITZGERALD 1
THE_LITTLE_PRINCE3 ANTOINE 1
TO_KILL_A_MOCKINGBIRD3 LEE 1
ANIMAL_FARM_3 ORWELL 1
PRIDE_AND_PREJUDICE3 AUSTEN 0
```

(file.txt)

Това са данните на Библиотеката съхранени в текстов файл. Тя се казва LibName, има 4 стелажа и капацитета на един стелаж е 5 - брой книги. Библиотеката има 15 книги, всяка книга си има собствено име – повтарят се имената на 5 книги по 3 пъти с разлика 2, 3 и заета/свободна (0/1).

Клас `LibraryBook:` Клас който е за данните на книгата - име, автор и дали книгата е заета/свободна.

Функциите `void TakeBook()` и `void ReturnBook()` служат за функциите `void getsBook(const string& name)` и `void returnsBook(const string& name)` от клас втори.

Клас `Library:`

Name	Value	Type
obj	{alphabet={ size=15 } m_LibraryName="LibName" m_stelaj=4 ...}	Library
alphabet	{ size=15 }	std::vector<LibraryBook,st...
m_LibraryName	"LibName"	std::string
m_stelaj	4	int
m_CapacityOfStelaj	5	int
mmap_location	{ size=15 }	std::multimap<int,LibraryB...
[comparator]	less	std::_Compressed_pair<std...
[allocator]	allocator	std::_Compressed_pair<std...
[1]	{m_strName="ANIMAL_FARM" m_strAuthor="ORWELL" m_bSituation=false }	std::pair<int const ,Library...
[1]	{m_strName="ANIMAL_FARM_2" m_strAuthor="ORWELL" m_bSituation=true }	std::pair<int const ,Library...
[1]	{m_strName="ANIMAL_FARM_3" m_strAuthor="ORWELL" m_bSituation=true }	std::pair<int const ,Library...
[1]	{m_strName="PRIDE_AND_PREJUDICE" m_strAuthor="AUSTEN" m_bSituation=true }	std::pair<int const ,Library...
[1]	{m_strName="PRIDE_AND_PREJUDICE3" m_strAuthor="AUSTEN" m_bSituation=true }	std::pair<int const ,Library...
[2]	{m_strName="PRIDE_AND_PREJUDICE3" m_strAuthor="AUSTEN" m_bSituation=false }	std::pair<int const ,Library...
[2]	{m_strName="THE_GREAT_GATSBY" m_strAuthor="FITZGERALD" m_bSituation=false }	std::pair<int const ,Library...
[2]	{m_strName="THE_GREAT_GATSBY2" m_strAuthor="FITZGERALD" m_bSituation=true }	std::pair<int const ,Library...
[2]	{m_strName="THE_GREAT_GATSBY3" m_strAuthor="FITZGERALD" m_bSituation=true }	std::pair<int const ,Library...
[2]	{m_strName="THE_LITTLE_PRINCE" m_strAuthor="ANTOINE" m_bSituation=false }	std::pair<int const ,Library...
[3]	{m_strName="THE_LITTLE_PRINCE2" m_strAuthor="ANTOINE" m_bSituation=true }	std::pair<int const ,Library...
[3]	{m_strName="THE_LITTLE_PRINCE3" m_strAuthor="ANTOINE" m_bSituation=true }	std::pair<int const ,Library...
[3]	{m_strName="TO_KILL_A_MOCKINGBIRD" m_strAuthor="LEE" m_bSituation=true }	std::pair<int const ,Library...
[3]	{m_strName="TO_KILL_A_MOCKINGBIRD2" m_strAuthor="LEE" m_bSituation=true }	std::pair<int const ,Library...
[3]	{m_strName="TO_KILL_A_MOCKINGBIRD3" m_strAuthor="LEE" m_bSituation=true }	std::pair<int const ,Library...
[Raw View]	{...}	std::multimap<int,LibraryB...
map_search	{ size=0 }	std::map<char,int,std::less...

Стелаж 1-ви

Стелаж 2-ри

Стелаж 3-ти

В функцията `istream& Input(istream& in)`, с отделна функция наречена (`void SortVector()`), правя сортиране по азбучен ред на вектора, който съхранява книгите. С цикъл for, разпределям книгите по стелажите и с `insert` слагам книгите в multimap – в key се съхранява стелажа. Ако капацитета за стелажа се напълни, новата книга ще бъде сложена в другия стелаж като се направи i++ и нулира j.

Name	Value	Type
obj	{alphabet={ size=15 } m_LibraryName="LibName" m_stelaj=4 ...}	Library
v	{ size=4 }	std::vector<LibraryBook,st...
[capacity]	4	int
[allocator]	allocator	std::_Compressed_pair<std...
[0]	{m_strName="ANIMAL_FARM_2" m_strAuthor="ORWELL" m_bSituation=true }	LibraryBook
[1]	{m_strName="ANIMAL_FARM_3" m_strAuthor="ORWELL" m_bSituation=true }	LibraryBook
[2]	{m_strName="PRIDE_AND_PREJUDICE" m_strAuthor="AUSTEN" m_bSituation=true }	LibraryBook
[3]	{m_strName="PRIDE_AND_PREJUDICE3" m_strAuthor="AUSTEN" m_bSituation=true }	LibraryBook
[Raw View]	{_Mypair=allocator }	std::vector<LibraryBook,st...

С функцията `vector<LibraryBook> AvailableBooks(const int& stelaj_no)`, първо създавам един вектор наречен (`vector<LibraryBook> available`), който ще действа за връщането на резултата. После търся и намирам всичките книги от зададения стелаж. Ако търсената книга е свободна, тя се записва в вектора създаден в началото, и този вектор се връща като резултат.

```
Available books in stelaj 1:
ANIMAL_FARM_2 ORWELL 1
ANIMAL_FARM_3 ORWELL 1
PRIDE_AND_PREJUDICE AUSTEN 1
PRIDE_AND_PREJUDICE3 AUSTEN 1
```

Name	Value	Type
obj	{alphabet={ size=15 } m_LibraryName="LibName" m_stelaj=4 ...}	Library
v1	{ size=3 }	std::vector<LibraryBook,st...
[capacity]	3	int
[allocator]	allocator	std::_Compressed_pair<std...
[0]	{m_strName="PRIDE_AND_PREJUDICE3" m_strAuthor="AUSTEN" m_bSituation=false }	LibraryBook
[1]	{m_strName="THE_GREAT_GATSBY" m_strAuthor="FITZGERALD" m_bSituation=false }	LibraryBook
[2]	{m_strName="THE_LITTLE_PRINCE" m_strAuthor="ANTOINE" m_bSituation=false }	LibraryBook
[Raw View]	{_Mypair=allocator }	std::vector<LibraryBook,st...

С функцията `vector<LibraryBook> UnavailableBooks(const int& stelaj_no)`, първо създавам един вектор наречен (`vector<LibraryBook> available`), който ще действа за връщането на резултата. После търся и намирам всичките книги от зададения стелаж. Ако търсената книга не е свободна, тя се записва в вектора създаден в началото, и този вектор се връща като резултат.

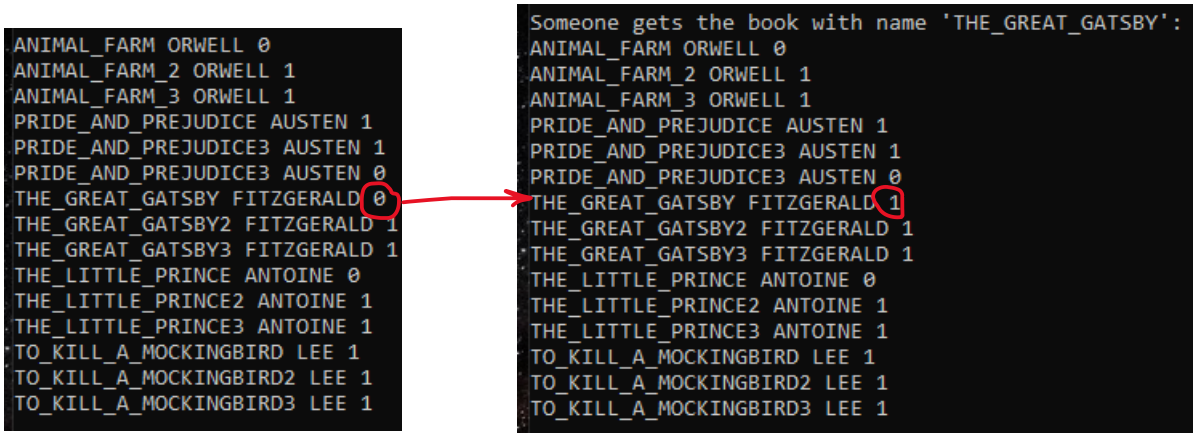
```
Unavailable books in stelaj 2:
PRIDE_AND_PREJUDICE3 AUSTEN 0
THE_GREAT_GATSBY FITZGERALD 0
THE_LITTLE_PRINCE ANTOINE 0
```

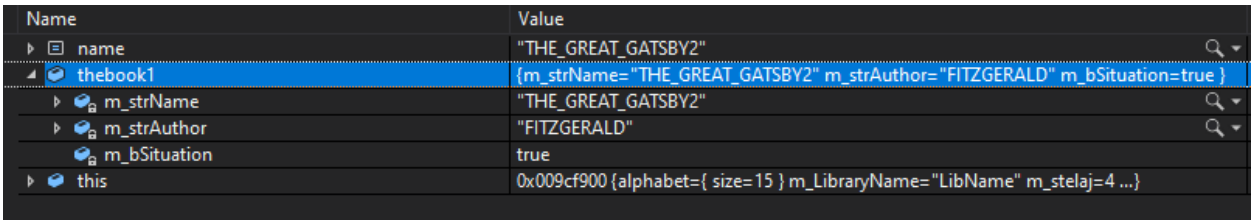
Name	Value
name	"THE_GREAT_GATSBY"
thebook	{m_strName="THE_GREAT_GATSBY" m_strAuthor="FITZGERALD" m_bSituation=false }
m_strName	"THE_GREAT_GATSBY"
m_strAuthor	"FITZGERALD"
m_bSituation	false
this	0x009ffd68 {alphabet={ size=15 } m_LibraryName="LibName" m_stelaj=4 ...}

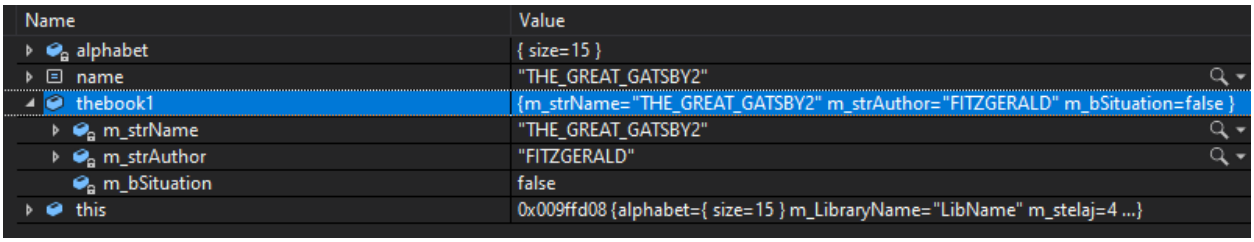
Name	Value
alphabet	{ size=15 }
name	"THE_GREAT_GATSBY"
thebook	{m_strName="THE_GREAT_GATSBY" m_strAuthor="FITZGERALD" m_bSituation=true }
m_strName	"THE_GREAT_GATSBY"
m_strAuthor	"FITZGERALD"
m_bSituation	true
this	0x005bfc88 {alphabet={ size=15 } m_LibraryName="LibName" m_stelaj=4 ...}

Функцията `LibraryBook findAvailable(const string& name)`, създава временно една книга, и търси свободна книга по подаденото име. Ако има такава, от втората стойност на мултимап който съдържа данните на тази книга се записват в временно създадената книга и данните се връщат като резултат.

Функцията `void getsBook(const string& name)`, заема свободната книга по зададеното име като създава една временна книга и й дава данните получени като резултат от предишната функция `LibraryBook findAvailable(const string& name)`. С функцията `TakeBook()`, създадена в клас `LibraryBook` , се заема книгата като от състояние свободно прави книгата в състояние заета. С replace() се отбелязва и променя в database'а, че книгата вече е заета.







Функцията `LibraryBook findUnavailable(const string& name)`, създава временно една книга, и търси заетата книга по подаденото име. Ако има такава, от втората стойност на мултимап който съдържа данните на тази книга се записват в временно създадената книга и данните се връщат като резултат.

Функцията `void returnsBook(const string& name)`, връща свободната книга по зададеното име като създава една временна книга и й дава данните получени като резултат от предишната функция `LibraryBook findUnvailable(const string& name)`. С функцията `ReturnBook()`, създадена в клас `LibraryBook` , се връща книгата като от състояние заета прави книгата в състояние свободна. С replace() се отбелязва и променя в database'а, че книгата вече е свободна.