|  |  |
| --- | --- |
|  | **AKADEMIA GÓRNICZO-HUTNICZA** |

Dokumentacja do projektu

**Library Management System**

z przedmiotu

**Języki programowania obiektowego**

Elektronika, drugi rok

*Krzysztof Sikora*

Piątek 13:00

prowadzący: Rafał Frączek

28.12.2022r.

# Opis projektu

Library Management System to program napisany w języku C++ służący do zarządzania biblioteką uczelnianą. Posiada on bazę książek oraz użytkowników zapisaną w zewnętrznych plikach i pozwala studentom wypożyczać, zwracać oraz rezerwować książki poprzez prosty interfejs w konsoli. Administrator ma natomiast możliwość przeglądać listę użytkowników: sprawdzać jakie książki wypożyczyli a jakie zarezerwowali, a także listę książek. Możliwe jest również dodawanie do bazy danych nowych pozycji, a także generowanie raportu, w którym przedstawieni są użytkownicy wraz z książkami których nie oddali w terminie i kwotą którą muszą zapłacić.

# Project description

Library Management System is a program written in the C++ language whose primary task is to manage a university library. It makes use of a database of its users and books which is contained in two external files. It let’s the students check out, return and reserve various books through a simple console interface. The administrator has the ability to view a list of all the users: check which books they’ve checked out or reserved, as well as the list of all the books. It is possible to add new records to the database, be it new users or items. There is also an option to generate a simple report showing all the users with overdue books and fees they have to pay.

# Instrukcja użytkownika

Po uruchomieniu programu użytkownik proszony jest o wybór pomiędzy menu dla studenta administratora. Wyboru tego, tak jak w pozostałych submenu dokonuje się poprzez wprowadzenie odpowiedniej cyfry bądź kodu i potwierdzenie wyboru klawiszem Enter. W następnym oknie student proszony jest o podanie swojego numeru indeksu, a administrator hasła dostępu.

Menu użytkownika:

Student posiada możliwość skorzystania z trzech pod-menu: zwrot książek, wypożyczenie książki, rezerwacja książki. Ponownie wybór odbywa się poprzez wpisanie odpowiedniej cyfry przedstawionej w menu. Po wejściu jedną z tych opcji student dokonuje wyboru książki poprzez wpisanie wewnętrznego numeru książki będącego jej identyfikatorem w bibliotece. Po potwierdzeniu wyboru pokazany wyświetlony zostanie komunikat świadczący o zakończeniu procesu a użytkownik wróci do menu głównego programu.

Menu Administratora:

Administrator ze swojego menu ma możliwość: przejrzeć listę wszystkich użytkowników, bądź wszystkich książek, dodać nowego użytkownika do bazy danych, dodać nową książkę, stworzyć raport zawierający wszystkich użytkowników wraz z książkami których nie zwrócili w terminie i opłatami. Podczas przeglądania bazy danych można podejrzeć więcej informacji o poszczególnym rekordzie wpisując numer indeksu/ID książki. Podczas dodawania nowych użytkowników administrator będzie po kolei proszony o wpisywanie danych i potwierdzanie ich klawiszem Enter. Gdy proces wprowadzania danych się zakończy, administrator wróci do menu głównego a nowy rekord zostanie dodany do bazy danych.

Aby wrócić do menu głównego programu z innego pod-menu podczas wyboru należy wpisać „-1” i potwierdzić wybór klawiszem Enter. Stamtąd użytkownik może opuścić program wybierając opcję nr 3.

# Kompilacja

Kompilacja projektu przebiegała w środowisku visual studio 2022. Wykorzystano wersję 14 języka C++. Program testowany był na systemie operacyjnym windows 11 i niektóre jego opcje mogą nie działać w innych systemach, z racji wykorzystania systemowych funkcji obsługi konsoli. W przypadku zaistnienia potrzeby kompilacji na inne systemy, należy zmodyfikować program, zamieniając wyłącznie funkcje systemowe z biblioteki <windows.h>.

# Pliki źródłowe

Projekt składa się z następujących plików źródłowych:

* *Book.h, Book.cpp* – deklaracja oraz implementacja klasy Book, podklasa Lib\_Item
* *DateParser.h, DateParser.cpp –* deklaracja i implementacja klasy DateParser
* *fileReadWrite.*h, *fiileReadWrite.cpp –* deklaracja i implementacja klasy fileReadWrite
* *Lib\_Item.*h, *Lib\_Item.cpp –* deklaracja i implementacja klasy Lib\_Item
* *Library.*h, *Library.cpp –* deklaracja I implementacja klasy Library (głowna klasa przechowująca bazę danych biblioteki)
* *Student.h, Student.cpp –* deklaracja i implementacja klasy Student, poklasa User
* *User.*h, *User.cpp –* deklaracja i implementacja klasy User
* *User\_Interface.*h, *User\_Interface.cpp –* deklaracja I implementacja klasy User\_Interfejs (interfejs użytkownika w konsoli)
* *Library\_management\_system.cpp* – główny plik zawierający funkcję main.

# Zależności

Brak.

# Opis klas

W projekcie utworzono następujące klasy:

* User – klasa reprezentująca użytkownika biblioteki.
  + Virtual void GetInfo(std::map<std::string, std::string> & infoMap) {}; - wirtualna metoda która zwraca dane o użytkowniku w postaci mapy [string, string].

Settery I gettery podstawowych parametrów:

* void setName(const std::string& input);
* void setSurname(const std::string& input);
* std::string getName() const;
* std::string getSurname() const;
* Student – podklasa klasy User, reprezentująca studenta. Rozszerza klasę User o następujące settery i gettery:
  + void setUniID(const std::string& input);
  + void setAge(const int& input);
  + void addBook(const Book& input);
  + void removeBook(const int& bID);
  + void addReserve(const Book& input);
  + std::string getUniID() const;
  + int getAge() const;
  + std::vector<Book> getCheckedOutItems() const; - zwraca wektor wypożyczonych książek
  + std::vector<Book> getReservedItems() const; - zwraca wektor książek zarezerwowanych
* Lib\_Item – klasa reprezentująca pojedynczy rekord w zbiorze biblioteki. Do jej metod publicznych należą:
  + virtual void getInfo(std::map<std::string, std::string>& infoMap) = 0; - zwraca wszystkie informacje o przedmiocie – podobnie jak w klasie User

Poniżej znajdują się gettery i settery wszystkich parametrów klasy:

* + void setTitle(const std::string& input);
  + void setInternalID(const int& input);
  + void setReserved(const bool& input);
  + void setChecked(const bool& input);
  + void setReturnDate(const std::string& input);
  + void setReservedDate(const std::string& input);
  + std::string getTitle() const;
  + int getInternalID() const;
  + bool getReserved() const;
  + bool getChecked() const;
  + std::string getReturnDate() const;
  + std::string getReservedTILL() const;
* Book – klasa rozszerzająca klasę Lib\_Item, reprezentująca książkę. Jej dodatkowe metody to jedynie gettery i settery:
  + void setISBN(const std::string& input);
  + void= setPageCount(const int& input);
  + std::string getISBN() const;
  + int getPageCount() const;
* DateParser – klasa posiadająca jedynie statyczne metody, służąca do operacjach na dacie, która przechowywana jest w programie w postaci zmiennej typu string.
  + static std::string getCurrentDate(); - zwraca aktualną date
  + static std::string getReturnDate(); - zwraca date zwrotu książki (miesiąc po wypożyczeniu)
  + static std::string toDate(std::string year, std::string month, std::string day); - dokonuje konwersji daty z trzech paramatrów na jeden typu string
  + static int dateDiff(std::string date1, std::string date2); - oblicza I zwraca różnicę pomiędzy dwiema datami w postaci ilości dni.
* fileReadWrite – klasa wykorzystywana przez obiekt Library do odczytu i zapisu danych do i z plików stanowiących bazę danych programu.
  + static void readUserDatabase(std::vector<Student>& students); - czyta dane z pliku LibraryUserDatabase I zapisuje dane do otrzymanego wektora
  + static void readBookDatabase(std::vector<Book>& books); - czyta dane z pliku LibraryBookDatabase I zapisuje dane do otrzymanego wektora
  + static void saveUserDatabase(const std::vector<Student>& students); - zapisuje otrzymane dane o użytkownikach do pliku
  + static void saveItemDatabase(const std::vector<Book>& books); - zapisuje otrzymane dane o książkach do pliku
* User\_Interface – klasa typu singleton, obsługująca konsolowy interfejs użytkownika. Posiada liczne metody prywatne, każda z nich reprezentująca osobne pod-menu oraz dwie metody publiczne:
  + void start(); - główne menu programu, metoda ta wywoływana jest na początku działania programu i z niej wywoływane są pozostałem pod-menu.
  + static User\_Interface& GetInstance(); - zwraca instancję klasy

# Zasoby

W projekcie wykorzystywane są następujące pliki zasobów:

* LibraryUserDatabase.txt – plik zawierający informacje o użytkownikach biblioteki, każda linia zawiera następujące informacje oddzielone znakiem „;”:
  + Imię
  + Nazwisko
  + Wiek
  + Numer indeksu
  + ID książek wypożyczonych, każda oddzielona znakiem „;”;
  + ID książek zarezerwowanych, oddzielone znakiem „;”;

Książki wypożyczone i zarezerwowane oddzielone są pomiędzy sobą znakami „\_;”.

* LibraryItemDatabase – plik zawierający informację o książkach znajdujących się w bibliotece, ponownie jedna linia określa dane jednej pozycji, oddzielając dane znakiem „;”:
  + Tytuł
  + Wewnętrzne ID książki w bibliotece
  + Wypożyczona – 0/1
  + Zarezerwowana – 0/1
  + Data wypożyczenia – format yyyy\_mm\_dd
  + Data rezerwacji – format jak wyżej
  + Numer ISBN
  + Liczba stron

# Dalszy rozwój i ulepszenia

Dalszy rozwój programu zakłada wykorzystanie SQL do przechowywania bazy danych. Możliwe jest również utworzenie nowych funkcji dla administratora, pozwalających na modyfikację istniejących już rekordów, czy też dodanie nowych klas takich jak np.: magazyn, czy film które mogłyby być przechowywane jako obiekt w bibliotece. Finalnym usprawnieniem byłoby dodanie graficznego interfejsu użytkownika.

# Inne

Brak.