

Politechnika Śląska  
Wydział Informatyki, Elektroniki i Informatyki

## **Programowanie Komputerów 3**

System w Bibliotece

---

autor	Paulina Czapla
prowadzący	Grzegorz Kwiatkowski
rok akademicki	2020/2021
kierunek	informatyka
rodzaj studiów	rok II, semestr 3
termin oddania	10.11.2020

---

# 1. Temat zadania

Tematem zadania było napisanie programu System w Bibliotece. Program miał zawierać:

- system naliczania opłat za nieoddanie zasobu;
- zasoby biblioteki to książki, płyty CD i filmy;
- możliwość dodawania i usuwania zasobów;
- możliwość sprowadzania zasobów z innych bibliotek;
- możliwość pożyczania zasobów innym bibliotekom.

Oprócz tego:

- klasy;
- dziedziczenie;
- operatory;
- strumienie/pliki;
- dynamiczna alokacja pamięci.

## 2. Analiza zadania

Zagadnienie przedstawia problem przechowywania i organizacji dużej ilości danych w programie, korzystania z operacji wejścia/wyjścia, komunikacji z użytkownikiem, korzystania z dynamicznych struktur danych oraz reprezentacji daty w programie.

## 3. Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem obiektowym. W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji.

### 3.1. Opis klas

#### 3.1.1 class List<T>

```
template<typename T>
class List
{
    Node<T>* head;
    unsigned counter;

public:
    List();
    Node<T>* push_front(Node<T>*);
    Node<T>* delete_element(Node<T>*);
    void display_list();
    int size();
    Node<T>* get_head();
    Node<T>* get_element(std::string);
    Node<T>* get_element(T&);
    void set_counter(int);
    void increase_counter();
    T& get_data(Node<T>*);
    ~List();
};
```

Jest to klasa będąca listą dwukierunkową, wykorzystująca template.

Lista jest wykorzystywana do przechowywania danych typu:

Person (lista użytkowników biblioteki), Borrowed (lista wypożyczonych zasobów), CD, Book, Movie.

## Atrybuty klasy

Node<T>\* head - wskaźnik na głowę listy;  
unsigned counter - licznik, wskazujący liczbę do tworzenia ID.

## Metody

**Node<T>\* push\_front(Node<T>\*)** - dodaje element na początek listy;  
**Node<T>\* delete\_element(Node<T>\*)** - usuwa wybrany element;  
**void display\_list()** - wyświetlanie całej listy;  
**int size()** - zwraca counter;  
**Node<T>\* get\_head()** - zwraca wskaźnik na głowę listy;  
**Node<T>\* get\_element(std::string)** - wyszukiwanie elementu po ID, wykorzystuje operatory == klas, które są wykorzystywane do tworzenia listy;  
**Node<T>\* get\_element(T&)** - wyszukiwanie elementu po tytule, wykorzystuje operatory == klas, które są wykorzystywane do tworzenia listy (bardziej szczegółowo opisane przy konkretnych operatorach);  
**void set\_counter(int)** - ustawia wartość pola counter;  
**void increase\_counter()** - zwiększa wartość counter o 1;  
**T& get\_data(Node<T>\*)** - zwraca obiekt, który znajduje się w Node<T> (elementie listy).

### 3.1.2 class Node <T>

```
template<typename T>
class Node
{
    Node* next, *prev;
    T data;

public:
    Node(T);
    Node(Node&);
    Node(T, Node<T>*, Node<T>*);
    void display();
    void set_next(Node*);
    Node* get_next();
    void set_prev(Node*);
    Node* get_prev();
    T& get_current_data();
    Node* get_current();
    Node& operator=(const Node&);
    bool operator==(const std::string&);
    ~Node();
};
```

Jest to klasa reprezentująca element listy dwukierunkowej. Do jej stworzenia wykorzystano template.

## Atrybuty klasy

Node\* next, \*prev - wskaźniki na następny i poprzedni element listy.  
T data - obiekt, który przechowuje lista.

## Metody

Klasa posiada metody set i get, ustawiające i zwracające wskaźniki next i prev. Oprócz tego:

**void display()** - wyświetla element listy przy wykorzystaniu operatora << obiektu, który jest parametrem klasy Node;

**T& get\_current\_data()** - zwraca obiekt data poprzez referencję;

**Node\* get\_current()** - zwraca this.

### 3.1.3 class Resource

```
class Resource
{
    friend std::ostream& operator<< (std::ostream&, Resource&);
protected:
    std::string ID;
    std::string authors_name;
    std::string title;
    Date release_date;
    bool is_available;
public:
    Resource(std::string, std::string, Date, std::string = "", bool = true );
    Resource(std::string, Date, std::string = "", bool = true);
    Resource();
    void change_availability(bool);
    virtual void create_id(int)=0;
    bool operator==(const std::string&);
    bool operator==(const Resource&);
    std::string get_ID();
    std::string get_authors_name();
    std::string get_title();
    std::string get_year_string();
    Date get_date();
    bool get_availability();
    virtual ~Resource();
};
```

Klasa Resource jest klasą abstrakcyjną.

Jej polami są dane zasobu biblioteki:

std::string ID - unikalne ID zasobu;

std::string authors\_name - imię i nazwisko autora;

std::string title - tytuł zasobu;

Date release\_date - obiekt Date z datą wydania zasobu;

bool is\_available - informacja, czy książka jest aktualnie dostępna do wypożyczenia.

## Metody

Posiada metodę wirtualną **create\_id(int)** oraz metody get, zwracające odpowiedni atrybut.

## Operatory

Klasa posiada operator globalny << odpowiedzialny za wyświetlanie danych na ekranie, oraz dwa operatory ==.

**operator==(const std::string&)** - jest wykorzystywany przy wyszukiwaniu zasobów po ID w liście. Zwraca true, jeśli wartość std::string jest zgodna z atrybutem obiektu std::string ID.

**operator==(const Resource&)** - jest wykorzystywany przy wyszukiwaniu zasobów po tytule. W celu znalezienia zasobu z odpowiednim tytułem, tworzony jest tymczasowy obiekt, posiadający wszystkie pola domyślne, oprócz std::string title. Operator zwraca true, jeśli wartości title obiektów są takie same. Przy porównywaniu wielkość liter nie ma znaczenia.

### 3.1.4 class Book, class CD, class Movie

```
class Book :
{
    public Resource
public:
    Book(std::string, std::string, Date, std::string = "", bool = true);
    Book(int, std::string, Date, std::string = "", bool = true);
    Book();
    void create_id(int);
    Book& operator=(const Book&);
    ~Book();
};

class CD :
{
    public Resource
public:
    CD(std::string, std::string, Date, std::string = "", bool = true);
    CD(int, std::string, Date, std::string = "", bool = true);
    CD();
    void create_id(int);
    CD& operator=(const CD&);
    ~CD();
};

class Movie :
{
    public Resource
public:
    Movie(std::string, std::string, Date, std::string = "", bool = true);
    Movie(int, std::string, Date, std::string = "", bool = true);
    Movie();
    void create_id(int);
    Movie& operator=(const Movie&);
    ~Movie();
};
```

Klasy Book, Movie i CD są klasami pochodnymi klasy Resource. Każda z tych klas ma swoją metodę create\_id(int), która tworzy id dla nowego zasobu. Parametrem tej metody jest największy numer zasobu, powiększony o 1. ID jest tworzone według zasady:

książki: K-000001-GL1

płyty: C-...-

filmy: F-....-

Pierwsza litera – oznacza rodzaj zasobu (K – książki, C – płyty CD, F – filmy). Następny jest numer unikalny dla danej filii biblioteki (książki mogą mieć ten sam numer w różnych bibliotekach). Ostatnia jest informacją o mieście i numerze filii biblioteki, dzięki czemu każde ID jest unikalne.

### 3.1.5 class Person

```
class Person
{
    friend std::ostream& operator<<(std::ostream& output, Person& data);
    std::string name;
    Date birth_date;
    std::string address;
    std::string phone_number;
    std::string ID;

public:
    List<Borrowed> borrowed_resources;

    Person(std::string, std::string, Date, std::string, std::string, List<Borrowed>);
    Person(std::string, std::string, Date, std::string, std::string);
    Person();
    std::string get_ID();
    std::string get_name();
    Date get_birth_date();
    std::string get_string_date();
    std::string get_address();
    std::string get_phone_number();
    bool operator==(const std::string&);
    ~Person() {}
};
```

Klasa ta przechowuje dane osób korzystających z biblioteki.

#### Atrybuty klasy

std::string name - imię i nazwisko osoby

Date birth\_date - obiekt klasy Date z datą urodzenia osoby;

std::string address - adres zamieszkania;

std::string phone\_number - numer telefonu osoby;

std::string ID - unikalne ID, będące jednocześnie numerem PESEL;

List <Borrowed> borrowed\_resources - lista wypożyczonych zasobów przez danego użytkownika.

## Metody

Klasa posiada kilka metod get, zwracających wartość danego atrybutu.

## Operatory

Klasa posiada operator globalny << służący do wypisania danych na ekran, oraz operator ==.

**operator==(const std::string&)** - jest wykorzystywany przy wyszukiwaniu osób po ID w liście. Zwraca true, jeśli wartość std::string jest zgodna z atrybutem obiektu std::string ID.

### 3.1.6 class Date

```
class Date
{
    friend std::ostream& operator<< (std::ostream& output, Date const& date);

protected:
    short unsigned int day;
    short unsigned int month;
    short unsigned int year;

public:
    Date(short unsigned int, short unsigned int, short unsigned int);
    Date(std::string);
    Date();
    void set_day(short unsigned int);
    void set_month(short unsigned int);
    void set_year(short unsigned int);
    void set_date_from_string(std::string);
    std::string date_to_string();
    std::string year_to_string();
    bool validate();
    bool check_string(std::string);
    int operator- (const Date& const);
    Date operator+ (const int& const);
    Date& operator=(const Date&);
    ~Date() {};
};
```

Stworzenie tej klasy było wymaganiem w projekcie. Atrybutami tej klasy są wartości typu short unsigned int, wyrażające kolejno dzień, miesiąc i rok.

## Metody

Klasa posiada kilka metod set, w tym set\_date\_from\_string(std::string), której parametrem jest data w formie ciągu znaków i która konwertuje datę podaną jako ciąg znaków w trzy wartości short unsigned int. Oprócz tego posiada metody:

**std::string date\_to\_string()** - konwertuje datę podaną jako trzy wartości short unsigned int na ciąg znaków;

**std::string year\_to\_string()** - konwertuje pole year (short unsigned int) na ciąg znaków.

**bool validate()** - sprawdza czy wartości nie przekraczają zakresu (rok i miesiąc);

**bool check\_string(std::string)** - sprawdza czy parametr std::string przechowuje datę (cyfry oddzielone odpowiednim znakiem).

Konstruktor domyślny ustawia aktualną datę.

## Operatory

Klasa posiada operator globalny << odpowiedzialny za wyświetlanie danych na ekranie.  
**int operator- (const Date&) const** - operator odejmowania dwóch dat od siebie, zwraca różnicę wyrażoną w dniach;

**Date operator+ (const int&) const** - operator dodawania liczby dni do daty;

**Date& operator=(const Date&)** - operator przypisania.

### 3.1.7 class File

```
template<typename T>
class File
{
protected:
    std::string filename;

public:
    File(std::string);
    virtual void write(T&) =0;
    virtual void read(T&) =0;
    bool check(std::fstream&);
};
```

Jest to klasa abstrakcyjna, wykorzystująca template. Jej atrybutem jest std::string filename, przechowujące nazwę pliku.

Posiada dwie metody wirtualne, przyjmujące jako parametr T, które jest odpowiednim typem, z którego obiektu mają być odczytywane/wpisywane dane.

Metoda write wpisuje odpowiednie dane do pliku, natomiast metoda read odczytuje.

Metoda bool check(std::fstream&) sprawdza, czy plik otworzył się prawidłowo.

### 3.1.8 class FileLibrary, class FileBorrowed, class FilePersonalData

```
class FileLibrary :
    public File<Library>
{
public:
    FileLibrary(std::string);
    void write(Library&);
    void read(Library&);
    ~FileLibrary() {};
};
```

```
class FileBorrowed :
    public File<List<Person>>
{
public:
    FileBorrowed(std::string);
    void write(List<Person>&);
    void read(List<Person>&);
    ~FileBorrowed() {};
};
```

```
class FilePersonalData :
    public File< List<Person>>
{
public:
    FilePersonalData(std::string);
    void write(List<Person>&);
    void read(List<Person>&);
    ~FilePersonalData() {};
};
```

Są to klasy pochodne klasy File. Klasa FileLibrary dziedziczy po klasie File z parametrem Library - dane odczytywane/ wpisywane do pliku znajdują się w obiekcie typu Library.

Klasy FileBorrowed oraz FilePersonalData dziedziczą po klasie File z parametrem List<Person>.



### 3.1.9 class Borrowed

```
class Borrowed
{
    friend std::ostream& operator<< (std::ostream&, Borrowed&);
    std::string resource_ID;
    Date borrow_date;
    Date deadline;

public:
    Borrowed(std::string, Date, Date);
    Borrowed();
    void display(Library*);
    std::string get_resource_ID();
    Date get_borrow_date();
    Date get_deadline();
    int count_fine();
    bool operator==(const std::string&);
};
```

Jest to klasa, która zawiera informacje o wypożyczeniach. Jej atrybutami są ID zasobu, data wypożyczenia oraz data zwrotu (deadline).

Deadline tworzony jest poprzez dodanie do borrow\_date odpowiedniej stałej. W programie jest to 21.

```
#define BORROW_TIME 21
```

#### Metody

Klasa posiada trzy metody get, zwracające resource\_ID, borrow\_date oraz deadline. Klasa posiada również metodę int count\_fine(), która zlicza jak wysoką opłatę należy uiścić przy zwrocie zasobu. Jeśli użytkownik nie spóźnił się z oddaniem, funkcja zwraca wartość 0. Koszt jednego dnia jest określony jako stała:

```
#define PRICE 2
```

### 3.1.10 class Library

```
class Library
{
public:
    Library();
    List<Book> books;
    List<CD> cds;
    List<Movie> movies;
    ~Library() {};
};
```

Klasa reprezentująca bibliotekę. Jej polami są trzy listy - lista z książkami, płytami oraz filmami. W listach znajdują się zasoby danej biblioteki.

### 3.1.11 class Menu

```
class Menu : public LibraryUserActions, public LibraryActions
{
public:
    void main_menu(Library&, Library&, Library&, List<Person>&);
    void resources_menu(Library&);
    void display_library_menu(Library&);
    void add_resource_menu(Library&);
    void external_libraries_menu(Library&, Library&, Library&);
    void users_menu(List<Person>& );
    void libraries_menu(Library&, Library&, Library&, short int);
};
```



Klasa składająca się z samych metod. Metody wyświetlają odpowiednie menu, następnie wychwytyją wybór użytkownika i wywołują odpowiednią funkcję (inne menu, albo dziedziczona metoda). Zastosowano dziedziczenie wielobazowe.

### 3.1.12 class LibraryActions

```
class LibraryActions :
    public ResourceActions
{
public:
    LibraryActions() {};
    void borrow(Library&, List<Person>&);
    void make_a_return(Library&, List<Person>&);
    bool search_library_IDs(Library&);
    bool search_library_titles(Library&);
    void show_imported_resources(Library&, std::string&);
    void import_resource(Library&, Library&);
    ~LibraryActions() {};
};
```

Klasa posiadająca metody do zarządzania bibliotekami. Dziedziczy po klasie ResourceActions. Funkcje programu, które wykonują metody to:

- wypożyczenie zasobu użytkownikowi;
- oddanie zasobu do biblioteki;
- przeszukanie biblioteki w celu znalezienia zasobu o określonym ID/tytule;
- wyświetlanie importowanych zasobów biblioteki;
- wypożyczanie zasobów innej bibliotece/sprowadzanie zasobów z biblioteki zewnętrznej.

### 3.1.13 class ResourceActions

```
class ResourceActions : public Text
{
public:
    ResourceActions() {};

    void add_new_book(Library&);
    void add_new_cd(Library&);
    void add_new_movie(Library&);
    void delete_resource(Library&);
    Node<Book>* find_book_byID(List<Book>&, std::string);
    Node<CD>* find_cd_byID(List<CD>&, std::string);
    Node<Movie>* find_movie_byID(List<Movie>&, std::string);
    Node<Book>* find_book_byTitle(List<Book>&, Book&);
    Node<CD>* find_cd_byTitle(List<CD>&, CD&);
    Node<Movie>* find_movie_byTitle(List<Movie>&, Movie&);
};
```

Klasa posiadająca metody zarządzające zasobami biblioteki. Funkcje programu, które wykonują metody to:

- dodawanie nowego zasobu;
- usuwanie zasobu;
- wyszukiwanie w liście zasobu po ID/tytule.

### 3.1.14 class LibraryUserActions

```
class LibraryUserActions
{
    Text text;
public:
    LibraryUserActions() {};
    Node<Person>* find_user_byID(List<Person>&);
    void add_new_user(List<Person>&);
    void delete_user(List<Person>&);
    ~LibraryUserActions() {};
};
```

Klasa posiadająca metody zarządzające użytkownikami biblioteki. Funkcje programu, które wykonują metody to:

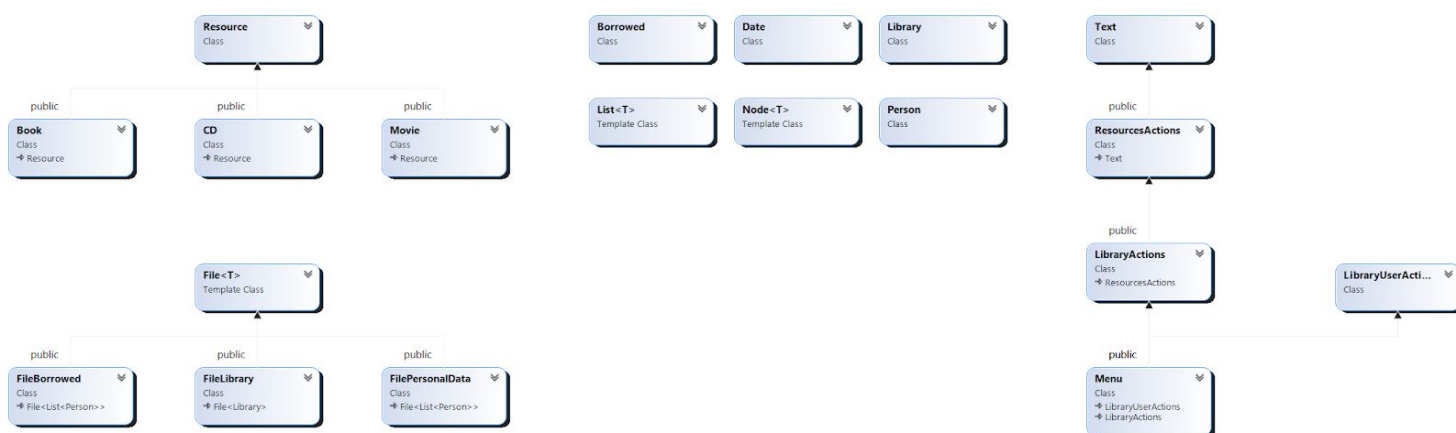
- dodawanie nowego użytkownika;
- usuwanie użytkownika z bazy;
- wyszukiwanie użytkownika po ID.

### 3.1.15 class Text

```
class Text
{
public:
    void display_green(std::string);
    void display_red(std::string);
};
```

Klasa z metodami wyświetlającymi tekst w kolorze zielonym i czerwonym.

## 3.2 Diagram klas



### 3.3 Obsługa plików

. Program odczytuje i zapisuje dane do plików. Są to pliki:

- `personal_data.txt` - plik z danymi użytkowników biblioteki

Format pliku z danymi osobowymi:

ID(PESEL)	Imię i nazwisko	Data urodzenia	Numer telefonu	Adres
-----------	-----------------	----------------	----------------	-------

- `library.txt`, `library2.txt`, `library3.txt` - pliki z zasobami bibliotek. Plik `library.txt` zawiera zasoby biblioteki głównej, natomiast pozostałe pliki, reprezentują biblioteki zewnętrzne

Format pliku z zasobami:

ID zasobu	Autor	Rok wydania	Czy dostępne	Tytuł
-----------	-------	-------------	--------------	-------

- `borrowed_resources.txt` - plik z wypożyczeniami

Format pliku z wypożyczeniami:

ID osoby	ID zasobu	Data wypożyczenia	Termin zwrotu
----------	-----------	-------------------	---------------

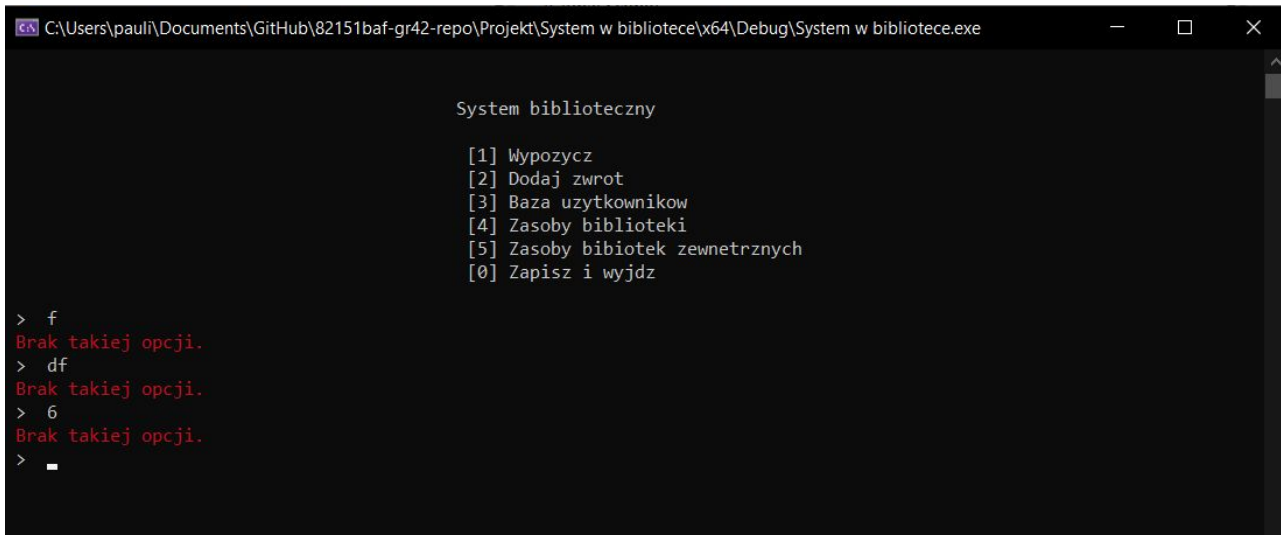
### 3.4 Ogólny opis działania programu

Na początku działania programu tworzone są obiekty `FilePersonalData`, `FileLibrary`, `FileBorrowed` oraz puste listy odpowiedniego typu. Następnie wywoływane są metody `read(T&)` obiektów obsługujących pliki z odpowiednim parametrem. Dane są wczytywane do list. W przypadku `FileBorrowed` tworzona jest lista list. Po prawidłowym wczytaniu danych z plików nie zostaje wyświetlony żaden komunikat. Zostaje stworzony obiekt typu `Menu` i zostaje wywołana metoda `main_menu` z odpowiednimi parametrami. Od tego momentu użytkownik wybiera funkcje w programie. Program jest zabezpieczony przed wprowadzaniem błędnych danych. Każde prawidłowo zakończone zadanie zasygnalizowane jest zielonym komunikatem. W przypadku błędu pojawia się czerwony komunikat.

Gdy użytkownik wykona już wszystkie zamierzone akcje, może zapisać i wyłączyć program. W tym celu wywoływane są metody `write(T&)` obiektów `FilePersonalData`, `FileLibrary`, `FileBorrowed`. Przy zapisie danych pojawiają się zielone komunikaty.

## 4. Instrukcja obsługi dla użytkownika

Po uruchomieniu programu i prawidłowym wczytaniu danych z pliku, wyświetlone zostaje menu główne. Użytkownik wybiera odpowiednią opcję poprzez wpisanie numeru. Jeśli użytkownik wpisze błędny znak, zostanie wyświetlony stosowny komunikat. Taka sytuacja zachodzi na zdjęciu poniżej.



```
C:\Users\pauli\Documents\GitHub\82151baf-gr42-repo\Projekt\System w bibliotece\x64\Debug\System w bibliotece.exe

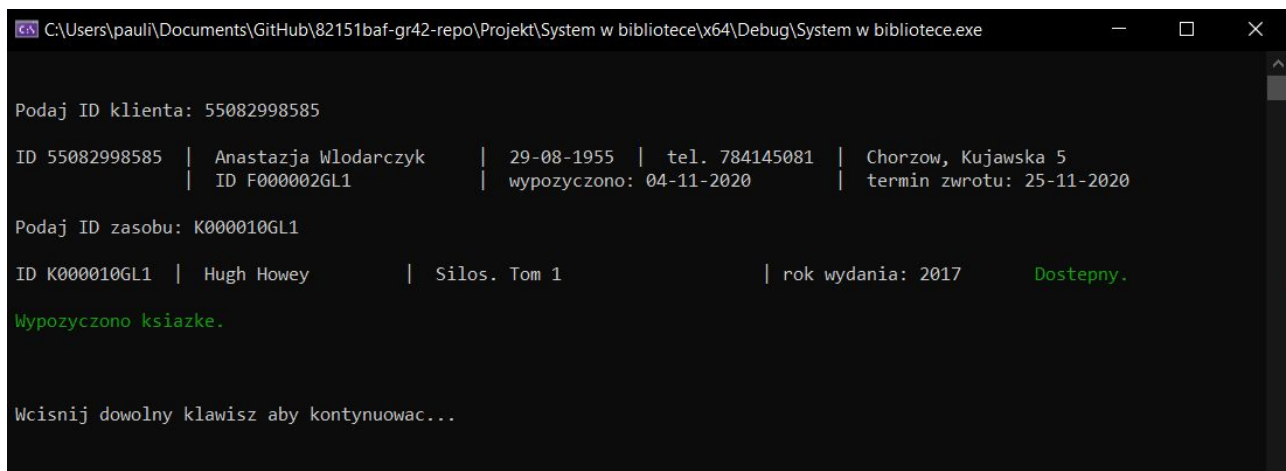
System biblioteczny

[1] Wypożycz
[2] Dodaj zwrot
[3] Baza uzytkownikow
[4] Zasoby biblioteki
[5] Zasoby bibiotek zewnetrznych
[0] Zapisz i wyjdz

> f
Brak takiej opcji.
> df
Brak takiej opcji.
> 6
Brak takiej opcji.
> _
```

### 4.1 Wypożycz

Po wyborze opcji nr 1, użytkownik zostanie zapytany o ID osoby wypożyczającej oraz ID zasobu. Jeśli któraś z tych danych jest nieprawidłowa (nie istnieje taki zasób/użytkownik, zasób jest już wypożyczony) zostanie wyświetlony komunikat.



```
C:\Users\pauli\Documents\GitHub\82151baf-gr42-repo\Projekt\System w bibliotece\x64\Debug\System w bibliotece.exe

Podaj ID klienta: 55082998585

ID 55082998585 | Anastazja Wlodarczyk | 29-08-1955 | tel. 784145081 | Chorzow, Kujawska 5
                | ID F000002GL1          | wypożyczono: 04-11-2020 | termin zwrotu: 25-11-2020

Podaj ID zasobu: K000010GL1

ID K000010GL1 | Hugh Howey | Silos. Tom 1 | rok wydania: 2017 | Dostępny.

Wypożyczono książkę.

Wcisnij dowolny klawisz aby kontynuowac...
```

### 4.2 Dodaj zwrot

Po wyborze opcji nr 2 z menu głównego, użytkownik zostanie poproszony o podanie ID klienta oraz ID zasobu do zwrotu. Jeśli minął termin zwrotu, zostanie naliczona opłata.

Użytkownik ma wybór, czy chce dokonać zwrotu, czy anulować (w przypadku, gdyby osoba np. nie miała pieniędzy).

```
C:\Users\pauli\Documents\GitHub\82151baf-gr42-repo\Projekt\System w bibliotece\x64\Debug\System w bibliotece.exe

Podaj ID klienta: 88020410896

ID 88020410896 | Bolesaw Jozwiak | 04-02-1988 | tel. 515186227 | Gliwice, Akcyjowa 13/2
                | ID K000017GL1 | wypożyczono: 12-10-2020 | termin zwrotu: 02-11-2020

Podaj ID zasobu: K000017GL1

Opóźnienie w oddaniu: 8 dni. Do zapłaty: 16 zł.

Wcisnij '1', aby potwierdzić otrzymanie zapłaty.
Wcisnij dowolny klawisz, aby anulować zwrot.
> 1

Dokonano zwrotu.

Wcisnij dowolny klawisz aby kontynuować...
```

#### 4.3 Baza użytkowników

Po wybraniu opcji nr 3 w menu głównym, zostanie wyświetlone menu zarządzania bazą użytkowników.

```
C:\Users\pauli\Documents\GitHub\82151baf-gr42-repo\Projekt\System w bibliotece\x64\Debug\System w bibliotece.exe

                        Uzytkownicy

[1] Wyświetl wszystkich uzytkownikow
[2] Znajdz uzytkownika po ID
[3] Dodaj nowego uzytkownika
[4] Usun uzytkownika
[0] Wyjdz

> _
```

Opcja nr 1 - wyświetlenie wszystkich użytkowników wraz z informacją o wypożyczonych przez nich książkach;

Opcja nr 2 - użytkownik zostanie poproszony o wpisanie ID osoby, jeśli taka osoba znajduje się w bazie, to jej dane zostaną wyświetlone;

Opcja nr 3 - użytkownik zostanie poproszony o dane nowego klienta biblioteki. Jeśli dane będą prawidłowe (np. numer telefonu, data), do bazy zostanie dodany nowy użytkownik.

Opcja nr 4 - po podaniu ID osoby do usunięcia z bazy, użytkownik będzie musiał potwierdzić swoją intencję.

## 4.4 Zasoby biblioteki

Po wyborze opcji nr 4 z menu głównego, zostanie wyświetlone menu zasobów.

```
C:\Users\pauli\Documents\GitHub\82151baf-gr42-repo\Projekt\System w bibliotece\x64\Debug\System w bibliotece.exe

Zasoby biblioteki

[1] Wyświetl zasoby
[2] Dodaj nowe zasoby
[3] Usun zasob
[0] Wyjdz

>
```

Opcja nr 1 wywoła menu wyświetlania zasobów.

```
C:\Users\pauli\Documents\GitHub\82151baf-gr42-repo\Projekt\System w bibliotece\x64\Debug\System w bibliotece.exe

Wyświetl zasoby

[1] Wyświetl wszystkie książki
[2] Wyświetl wszystkie płyty
[3] Wyświetl wszystkie filmy
[4] Wyświetl po ID
[5] Wyświetl po tytule
[0] Wyjdz

> _
```

Po wyborze opcji nr 2 z menu zasobów, użytkownik musi wybrać, jaki tym zasobu chce dodać. Następnie podaje dane zasobu.

```
C:\Users\pauli\Documents\GitHub\82151baf-gr42-repo\Projekt\System w bibliotece\x64\Debug\System w bibliotece.exe

Dodawanie zasobow

[1] Dodaj ksiazke
[2] Dodaj plyte
[3] Dodaj film
[0] Wyjdz

> 1

Podaj imie i nazwisko autora:
> Adam Mickiewicz

Podaj tytul ksiazki:
> Pan Tadeusz

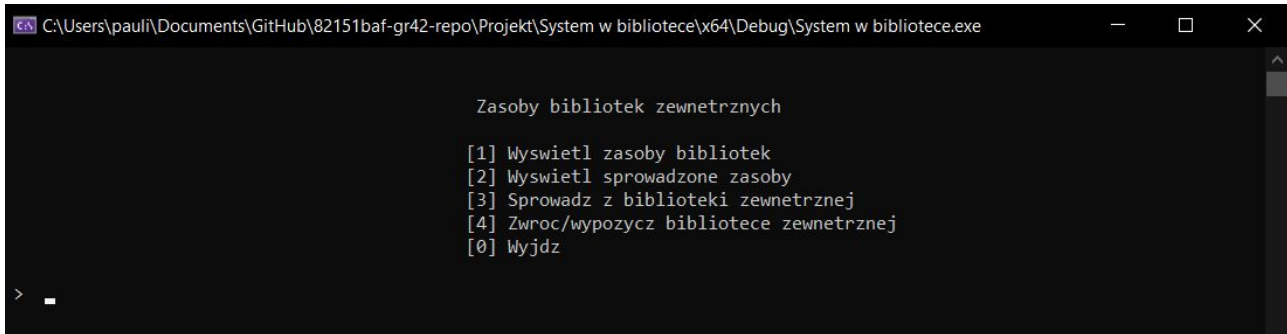
Podaj rok wydania:
> 2000
Ksiazka zostala dodana.

Wcisnij dowolny klawisz aby kontynuowac...
```

Po wyborze opcji nr 3 z menu zasobów, użytkownik zostanie poproszony o podanie ID zasobu do usunięcia. Jeśli ID jest prawidłowe i zasób jest dostępny, zostanie on usunięty z bazy.

#### 4.5 Zasoby bibliotek zewnętrznych

Po wyborze opcji nr 5 z menu głównego, zostanie wyświetlone menu zasobów bibliotek zewnętrznych.



```
C:\Users\pauli\Documents\GitHub\82151baf-gr42-repo\Projekt\System w bibliotece\x64\Debug\System w bibliotece.exe

Zasoby bibliotek zewnętrznych

[1] Wyświetl zasoby bibliotek
[2] Wyświetl sprowadzone zasoby
[3] Sprowadz z biblioteki zewnętrznej
[4] Zwroc/wypożycz bibliotece zewnętrznej
[0] Wyjdź

> 
```

Opcja nr 1 - należy wybrać bibliotekę zewnętrzną, której zasoby użytkownik chce wyświetlić. Następnie wyświetlane jest menu wyświetlania zasobów i użytkownik może wybrać odpowiednią opcję.

Opcja nr 2 - wyświetla wszystkie zasoby sprowadzone z zewnętrznej biblioteki.

Opcja nr 3 - sprowadzenie zasobu z biblioteki zewnętrznej poprzez podanie jego ID. Aby sprowadzenie było możliwe, zasób musi być dostępny.

Opcja nr 4 - zwrot/wypożyczenie zasobu bibliotece zewnętrznej. Użytkownik musi wybrać, której bibliotece chce zwrócić zasób. Aby było to możliwe, zasób musi być dostępny.

## 5. Wnioski

Program “System w bibliotece” wymagał obmyślenia odpowiedniej dynamicznej struktury danych, która by pozwoliła na przechowywanie informacji w programie w sposób zorganizowany. Przez specyfikę programu, który posiada listę dwukierunkową wykorzystującą template, wymagana była implementacja wielu operatorów, w szczególności operatorów porównania i wyświetlania. Oprócz tego zaimplementowano odpowiednie operatory dla klasy Date. Program musiał zostać zaprojektowany zgodnie z paradygmatem obiektowym, z wykorzystaniem informacji poznanych na laboratoriach. Program był dobrą okazją do zmierzenia się z programowaniem obiektowym po raz pierwszy w większym projekcie.

Link do repozytorium z projektem:  
<https://github.com/polsl-aei-pk3/82151baf-gr42-repo.git>