САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-Механический факультет

Реализация читательского дневника
Отчет по проекту

Студент: Юферева Диана Дмитриевна
Преподаватель: Нестерчук Григорий Анатольевич

Группа: 23.Б12-мм.

Санкт-Петербург
2024

Содержание

1 Введение

При создании данного проекта я хотела, чтобы тема моей работы,хоть немного отражала меня. Поэтому за основу я взяла свое хобби.
Прочитав большое количество книг, я столкнулась с проблемой, что по мере прочитанного истории, названия произведений, авторы книг забываются или сливаются в абстрактное пятно.
Поэтому мной было принято решение сделать реализацию читательского дневника.
Его цель - возможность записывать и хранить уже прочитанные книги, так же их можно отсортировать по жанру/автору/ вашей оценке.Даже можно вывести впечатление от прочитанной книги, если это необходимо.
Благодаря читательскому дневнику, вы сможете найти прочитанные книги, чтобы насладиться ими вновь.

2 Работа над проектом

Во время работы над проектом возникли некотроые трудности :

- По началу также было неудобно писать на Visual Studio. До этого я использовала CodeBlocks. Но из-за того, что в Visual Studio намного удобнее писать проект с несколькими файлами, поэтому я сделала выбор в его пользу.
- Возникли большие трудности при сортировках с исходным файлом вывода.
  Поэтому был сделан файл "settings.txt".
- Из-за "громоздкости"сортировок, было принято решение использовать односвязный список.
  Также он упростил задачу по чтению данных из файла.

2.1 Выбор инструментов

Я работала в среде разработки "Visual Studio 2022". Используя компилятор Microsoft Visual C++ 19
В проекте я использовала следующие библиотеки, классы и файлы :

- iostream - часть стандартной библиотеки C++.
  Для ввода/вывода данных.
- fstream - часть стандартной библиотеки C++.
  Для работы с файлами, в данном случае для чтения из файла.

- string - класс из стандартной библиотеки C++.
  Для работы со строками.
- windows.h - заголовочный файл, предоставляет доступ к различным функциям Windows API.
- sstream - часть стандартной библиотеки C++.
  Для работы с потоками ввода/вывода.
- MyBook.h - заголовочный файл, предоставляет доступ к файлу, в котором описаны методы для работы с книгами.
- LinkedList.h - заголовочный файл, предоставляет доступ к файлу, в котором описаны методы работы односвязного списка.

2.2 Архитектура проекта

Описание функционала в данном случае будет происходить так, я кратко поясню зачем та или иная функция, а подробнее прочитаете в описании кода.

Функции MyBook

- void inputbook();
  Функция, через которую пользователь вводит данные, необходимые для книги.Эти данные сохраняются в соответсвующих строках.
- string TestRating(string rating);
  Функция проверяет корректность заполнения строки оценки, если все хорошо, возвращается строка.Иначе придется перезаписывать строку.
- `void writebook(std::string _file_name);\verb`
  Данная функция записывает книгу (т.е данные полученные из inputbook() в файл settings.txt.
- void LibraryOut();
  Данная функция записанные данные полученные из inputbook() в "чистовик"(т.е файл dlitbooks.txt), где это будут читать пользователи.
- void WriteSentence(string Impression);
  Данная функция делит строку на предложения.Ориентируется по точкам.Т.к мне было нужна возможность добавлять ?????? или !!!!!.
- `string split_line(string& input_string);`
  Данная функция разделяет строку на подстроки с применением символа-разделителя.Возвращает первую подстроку до разделителя.

- void readbook(std::ifstream input);
  Данная функция считывает из файла строку, при помощи функции `string split_line(string& input_string);` разбивает строку и возвращает каждому полю объекта новые значения (т.е например author(поле объекта) присвается новое значение, полученное из файла.

- `void PrintBookInfo(std::ostream & out_stream);`
  данная функция выводит на консоль информацию о книге (кроме поля впечатления)

- `void PrintBookInfo1(std::ostream & out_stream);`
  данная функция выводит на консоль информацию о книге (включая поле впечатления)

- `void PrintBookImp(std::ostream& out_stream);`
  данная функция выводит на консоль только поле впечатления.

Функции LinkedList Была создана структура Node, которая представляет узел в нашем списке (односвязном).
В поле data стоит указатель на объект Mybook(т.е данные берем оттуда).
На следующей строке в поле next стоит указатель на Node (т.е на следующий узел в списке)
Конструктор Node принимает данные указатели и инициализирует соответствующие поля.
Конструктор копирования Node соответственно копирует данные из существующих узлов.
Деструктор освобождает память, говоря что поля data и next пустые.
Далее создается класс LinkedList. В public создается конструктор LinkedList(), в котором инциализируется head (пустой строкой). Деструктор освобождает память с помощью функции dispose().(она объявлена в private)
Потом мы также объявляем функцию int Length(), которая считает длину собственно.
Потом мы пишем функцию bool IsEmpty(), которая возвращает значение true, если head равно nullptr.
Также прописывается функция bool IndexValid(int index), она проверяет вышло ли значение index за диапазон (он должен лежать в пределах от нуля (включительно) и до значения Length()). Если все хорошо, то он возвращает true.
Потом мы объявляем две функции void InsertNode(int index, Node* node) и Node* ExtractNode(int index), которые добавляют элемент и удаляют.
Далее мы сделали указатель на голову (head)
Как я уже писала ранее в private мы объявляем функцию dispose(), которая помогает очищать память.

Функции Pet Сначала мы создаем класс BoookLibrary. В public (там и есть только public). Мы прописываем конструктор и деструктор (но сразу

спойлер, они оба пустые).
Потом с помощью LinkedList мы создаем переменную books, которая будет односвязным списком.
Далее мы прописываем строку `library_file_name`, которая будет хранить название файла, с котором мы работаем.

- Функция void PrintHeader()
  Данная функция выводит на консоль название моей программы.Сделано чисто для красоты.

- Функция void printMenu()
  Данная функция выводит меню на экран.

- Функция void LoadLibraryFromFile()
  Данная функция загружает информацию о книгах из файлах.

- Функция int choice()
  Данная функция с помощью switch(choice) выбирает case исходя из того, какую цифру выбрал пользователь.

- void FindBookByAuthor(string Author)
  Данная функция ищет необходимое поле author в списке и выводит на консоль книги этих авторов.

- void FindBookByGenre(string Genre)
  Данная функция ищет необходимое поле genre в списке и выводит на консоль книги этого жанра.

- void FindBookByAuthoroid OutImpression`(string Book_name, string Author)`
  Данная функция ищет необходимые поля author и `book_name` в списке и выводит на консоль впечатление об этих книгах.

- void FindBookByRating(string Rating)
  Данная функция ищет необходимое поле rating в списке и выводит на консоль книги с такой оценкой.

3 Документация по коду

3.1 Файл MyBook.h

```
1   #pragma once
2   #include <iostream>
3   #include <fstream>
4   #include <string>
5   #include <windows.h>
6   #include <sstream>
7   using namespace std;
```

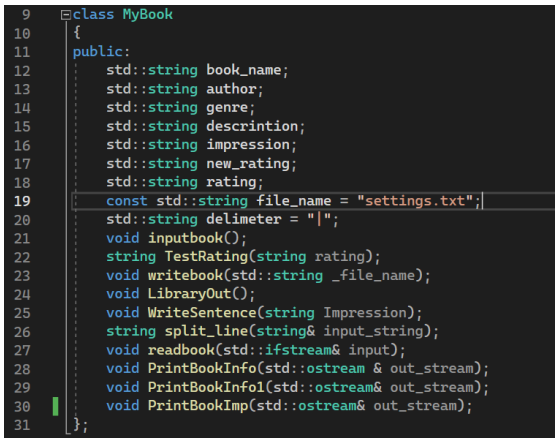
Рис. 1: Начало кода. Показывает какие библиотеки и классы подключены

В файле Mybook.h, который является заголовочным я подключила :

- iostream - часть стандартной библиотеки C++.
  Для ввода/вывода данных.
- fstream - часть стандартной библиотеки C++.
  Для работы с файлами, в данном случае для чтения из файла.
- string - класс из стандартной библиотеки C++.
  Для работы со строками.
- windows.h - заголовочный файл, предоставляет доступ к различным функциям Windows API.
- sstream - часть стандартной библиотеки C++.
  Для работы с потоками ввода/вывода.
- pragma once - во избежания двойного включения данного заголовочного файла в проекте.

Далее создается класс MyBook, где прописаны названия функций, которые подробно прописаны в файле MyBook.(рис.2)
Пропишем их и коротко поясним зачем они :

- с 12 - 18 строчку кода мы обьявили переменные (в нашем случае они вляются строками)
  Строка 17 специально добавлена для дальнейшей функции проверки.
- на 19 строчке уже объявлена постоянная строка, которая хранит название файла settings.txt
- на 20 строке хранится разделитель



```
class MyBook
{
public:
    std::string book_name;
    std::string author;
    std::string genre;
    std::string descrintion;
    std::string impression;
    std::string new_rating;
    std::string rating;
    const std::string file_name = "settings.txt";
    std::string delimeter = "|";
    void inputbook();
    string TestRating(string rating);
    void writebook(std::string _file_name);
    void LibraryOut();
    void WriteSentence(string Impression);
    string split_line(string& input_string);
    void readbook(std::ifstream& input);
    void PrintBookInfo(std::ostream & out_stream);
    void PrintBookInfo1(std::ostream& out_stream);
    void PrintBookImp(std::ostream& out_stream);
};
```

Рис. 2: class MyBook

- void inputbook();
  Функция, через которую пользователь вводит данные, необходимые для книги.Эти данные сохраняются в соответсвующих строках.

- string TestRating(string rating);
  Функция проверяет корректность заполнения строки оценки, если все хорошо, возвращается строка.Иначе придется перезаписывать строку.

- `void writebook(std::string _file_name);`
  Данная функция записывает книгу (т.е данные полученные из inputbook() в файл settings.txt.

- void LibraryOut();
  Данная функция записанные данные полученные из inputbook() в "читовик"(т.е файл dlitbooks.txt), где это будут читать пользователи.

- void WriteSentence(string Impression);
  Данная функция делит строку на предложения.Ориентируется по точкам.Т.к мне было нужна возможность добавлять ?????? или !!!!!.

- `string split_line(string& input_string);`
  Данная функция разделяет строку на подстроки с применением символа-разделителя.Возвращает первую подстроку до разделителя.

- void readbook(std::ifstream input);
  Данная функция считывает из файла строку, при помощи функции `string split_line(string& input_string);` разбивает строку и возвращает каждому полю объекта новые значения (т.е например author(поле объекта) присвается новое значение, полученное из файла.

- `void PrintBookInfo(std::ostream & out_stream);`
  данная функция выводит на консоль информацию о книге (кроме поля впечатления)
- `void PrintBookInfo1(std::ostream & out_stream);`
  данная функция выводит на консоль информацию о книге (включая поле впечатления)
- `void PrintBookImp(std::ostream& out_stream);`
  данная функция выводит на консоль только поле впечатления.

3.2 Файл MyBook.cpp

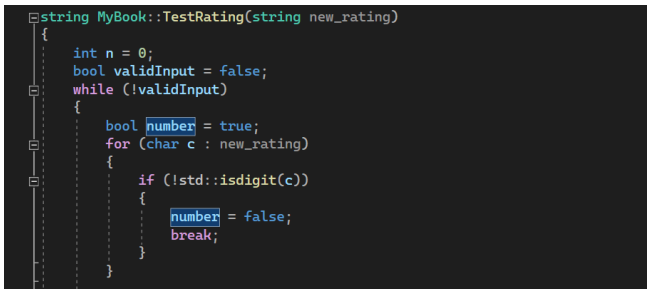
Теперь разберем подробнее работу каждой функции:

```
5   void MyBook::inputbook()
6   
7   {
8       std::cin.ignore();
9       std::cout << "Введите название книги: ";
10      getline(std::cin, book_name);
11      std::cout << "Введите автора: ";
12      getline(std::cin, author);
13      std::cout << "Введите жанр: ";
14      getline(std::cin, genre);
15      std::cout << "Введите впечатление: ";
16      getline(std::cin, impression);
17      std::cout << "Введите описание: ";
18      getline(std::cin, descrintion);
19      std::cout << "Оценка по десятибалльной шкале: ";
20      getline(std::cin, new_rating);
21      rating = TestRating(new_rating);
22  };
```

Рис. 3:

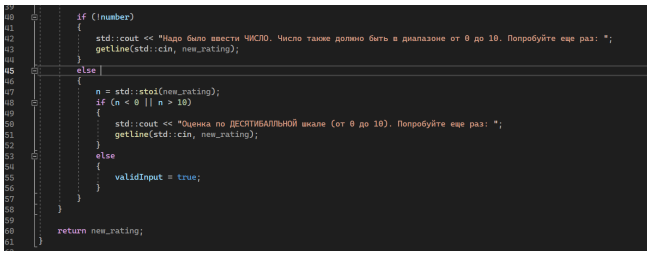
Функция void inputbook() Функция, через которую пользователь вводит данные, необходимые для книги.Эти данные сохраняются в соответсвующих строках.
Сначала пишем std::cin.ignore(), чтобы "очистить"все введенное раннее.(т.е игнорируем все до конца строки)
Далее мы просим пользователя ввести данные.Для их записи в определенные строки (author, genre и тд), пользуемся getline(std::cin, нужная нам строка), чтобы строка считывалась целиком. Для строки rating также прописывается дополнительная проверка.

Функция string TestRating(string rating)



```
string MyBook::TestRating(string new_rating)
{
    int n = 0;
    bool validInput = false;
    while (!validInput)
    {
        bool number = true;
        for (char c : new_rating)
        {
            if (!std::isdigit(c))
            {
                number = false;
                break;
            }
        }
```

Рис. 4:



```
    if (!number)
    {
        std::cout << "Надо было ввести ЧИСЛО. Число также должно быть в диапазоне от 0 до 10. Попробуйте еще раз: ";
        getline(std::cin, new_rating);
    }
    else
    {
        n = std::stoi(new_rating);
        if (n < 0 || n > 10)
        {
            std::cout << "Оценка по ДЕСЯТИБАЛЛЬНОЙ шкале (от 0 до 10). Попробуйте еще раз: ";
            getline(std::cin, new_rating);
        }
        else
        {
            validInput = true;
        }
    }
}

return new_rating;
}
```

Рис. 5:

Функция проверяет корректность заполнения строки оценки, если все хорошо, возвращается строка.Иначе придется перезаписывать строку.
Как он проверяет все ли хорошо.Было выставлено предупреждение, что это должно быть числовое значение(От 0 до 10)
Сначала мы создаем две переменные n типа int и validInput типа bool изначально равную false.
Потом мы в цикле while (validInput равен true)
Сначала объявляем переменную number типа bool изначально равную true.
Потом циклом for перебираем каждый символ в строке `new_rating` и если оказывается так,что символ не цифра (!std::isdigit), то пишем что .
Сначала number равно false. И выходим из цикла
Потом прописываем, что при number равном false мы просим ввести другое значениею
Иначе мы говорим, что n равен значению того числа, которое находится в строке (с помощью функции atoi(), преобразует string в int)
Если данное число не находится в диапазоне от 0 до 10. Если это так, то мы просим ввести новые данные в эту строку.Также присваем значение новой строки числу n.
Иначе говорим, что validInput равен true.
В конце функции возвращаем строку return `new_rating`

Функция `string split_line(string& input_string)`

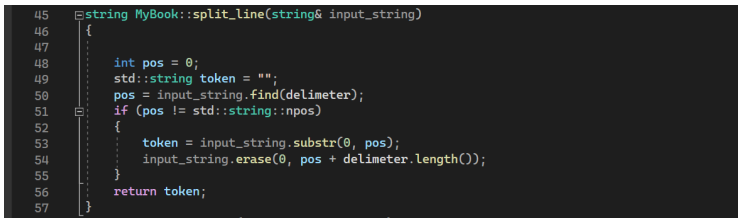


Рис. 6:

Данная функция разделяет строку на подстроки с применением символа-разделителя.
Сначала мы создаем переменную pos типа int, чтобы хранить позицию разделителя и переменую token типа string, чтобы хранить подстроку до разделителя.
Далее мы ищем позицию разделителя(delimeter)(с помощью метода find()). Если позиция оказалась найденной (т.е не равной std::string::npos)(если метод find() не находит нужную нам подстроку, то возвращает std::string::npos), то мы приравниваем переменную token к подстроке от входной строки, начиная с 0 индекса до индекса pos - 1 (т.е длиной pos). Потом мы у входной строки удаляем эту подстроку , но с длиной pos + delimeter.length() (т.е мы удаляем подстроку и разделитель).
Возвращается переменная token.

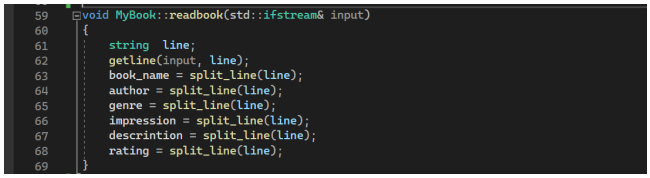
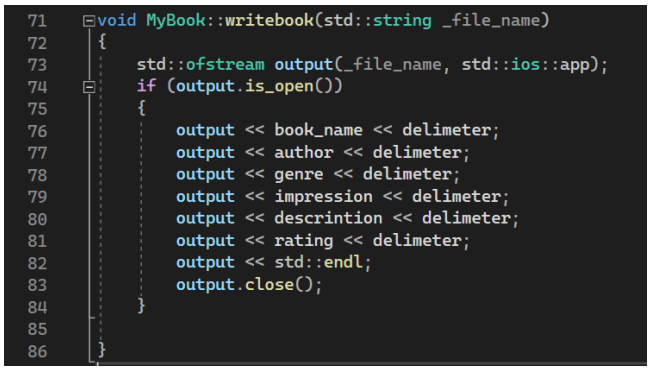


Рис. 7:

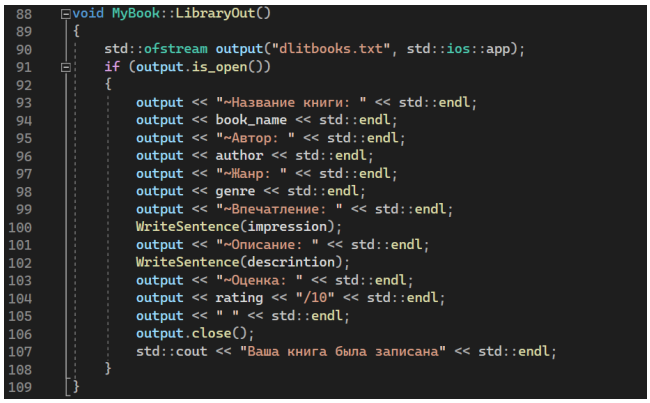
Функция void readbook(std::ifstream input) Данная функция считывает из файла строку, при помощи функции `string split_line(string& input_string);` разбивает строку и возвращает каждому полю объекта новые значения Сначала создаем переменную line типа string, в котором будет хранится строка, которую мы получаем из файла settings.txt.С помощью getline() мы считываем строку из файла.
Далее с помощью функции `string split_line(string& input_string)` каждой нашей переменной, относящейся к книге(`book_name`, author и тд) присваем значение возвращаемой подстроки от функции.Постепенно уменьшая строку.



```
71  void MyBook::writebook(std::string _file_name)
72  {
73      std::ofstream output(_file_name, std::ios::app);
74      if (output.is_open())
75      {
76          output << book_name << delimeter;
77          output << author << delimeter;
78          output << genre << delimeter;
79          output << impression << delimeter;
80          output << descrintion << delimeter;
81          output << rating << delimeter;
82          output << std::endl;
83          output.close();
84      }
85
86  }
```

Рис. 8:

Функция `void writebook(std::string _file_name)`
Данная функция записывает книгу (т.е данные полученные из inputbook() в файл settings.txt.
С помощью потока вывода мы в файл settings.txt.(дополнительно прописывая std::ios::app, чтобы не перезаписывался новый файл, а дописывать данные в конец) мы записываем в него данные(Если файл открыт для записи), после каждой строки мы добавляем разделитель. После этого закрываем файл.



```
88  void MyBook::LibraryOut()
89  {
90      std::ofstream output("dlitbooks.txt", std::ios::app);
91      if (output.is_open())
92      {
93          output << "~Название книги: " << std::endl;
94          output << book_name << std::endl;
95          output << "~Автор: " << std::endl;
96          output << author << std::endl;
97          output << "~Жанр: " << std::endl;
98          output << genre << std::endl;
99          output << "~Впечатление: " << std::endl;
100         WriteSentence(impression);
101         output << "~Описание: " << std::endl;
102         WriteSentence(descrintion);
103         output << "~Оценка: " << std::endl;
104         output << rating << "/10" << std::endl;
105         output << " " << std::endl;
106         output.close();
107         std::cout << "Ваша книга была записана" << std::endl;
108     }
109 }
```

Рис. 9:

Функция void LibraryOut() Данная функция записанные данные полученные из inputbook() в файл dlitbooks.txt.
С помощью потока вывода мы в файл dlitbooks.txt.(дополнительно прописывая std::ios::app, чтобы не перезаписывался новый файл, а дописывать данные в конец) записываем данные (Если файл открыт для записи), но в

более "читабельном "варианте.Также это достигается и при помощи функции void WriteSentence(string Impression)
После этого закрываем файл.После этого выводим на консоль "Ваша книга была записана "

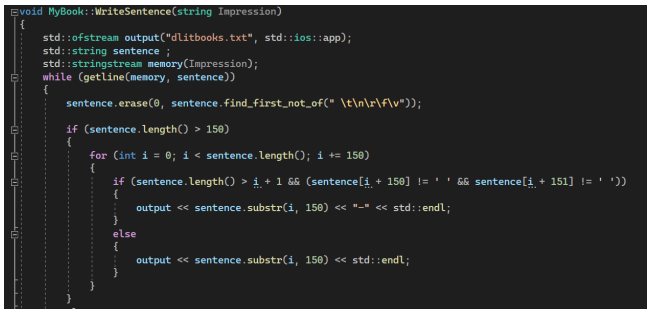


Рис. 10:

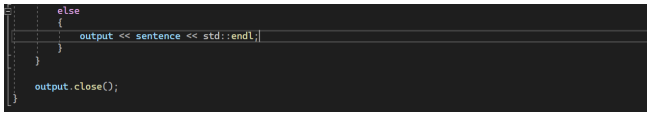
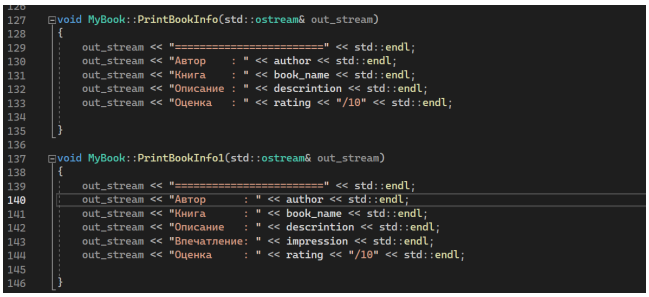


Рис. 11:

Функция void WriteSentence(string Impression) Данная функция делит строку на подстроки с определнным количеством символов.
С помощью потока вывода мы в файл dlitbooks.txt.(дополнительно прописывая std::ios::app, чтобы не перезаписывался новый файл, а дописывать данные в конец) записываем следующие данные
Для начала создаем пременную sentence типа string и объект memory типа stringstream (чтобы можно было работать с содержимым заданой строки)
Потом в цикле while пока возможно взять из объекта memory подстроку.Мы удаляем пробелы в начале каждого предложения (при помощи sentence.erase)
Потом мы проверям что длина строки больше 150,если это так ,то мы в цикле for с шагом 150 ее разбиваем на подстроки.
Более того мы проверяем при длине sentence больше i + 1 и символы стоящие на позициях 150 и 151 не равны пробелу, то мы ставим между ними тире Если условие if не выполняется,то мы записываем каждое предложение в файл и закрываем его.

Функции `void PrintBookInfo(std::ostream & out_stream)`
`void PrintBookInfo1(std::ostream & out_stream)`



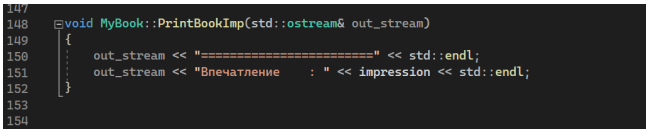
```
void MyBook::PrintBookInfo(std::ostream& out_stream)
{
    out_stream << "=======================" << std::endl;
    out_stream << "Автор     : " << author << std::endl;
    out_stream << "Книга     : " << book_name << std::endl;
    out_stream << "Описание  : " << descrintion << std::endl;
    out_stream << "Оценка    : " << rating << "/10" << std::endl;


}

void MyBook::PrintBookInfo1(std::ostream& out_stream)
{
    out_stream << "=======================" << std::endl;
    out_stream << "Автор       : " << author << std::endl;
    out_stream << "Книга       : " << book_name << std::endl;
    out_stream << "Описание    : " << descrintion << std::endl;
    out_stream << "Впечатление: " << impression << std::endl;
    out_stream << "Оценка      : " << rating << "/10" << std::endl;


}
```

Рис. 12:



```
void MyBook::PrintBookImp(std::ostream& out_stream)
{
    out_stream << "=======================" << std::endl;
    out_stream << "Впечатление     : " << impression << std::endl;
}
```

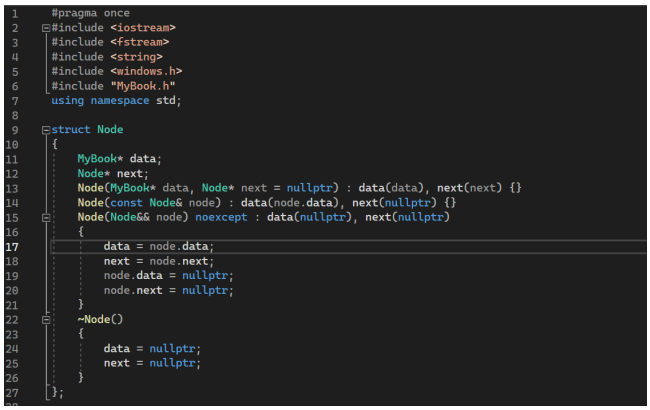
Рис. 13:

`void PrintBookImp(std::ostream& out_stream)`
Я решила написать 3 функции сразу, тк единственное их отличие - количество выводимых данных.
В `void PrintBookInfo(std::ostream & out_stream)` функция выводит на консоль информацию о книге(строки author, genre и тд)(кроме поля впечатления) (при помощи std::cout)
В `void PrintBookInfo1(std::ostream & out_stream)` функция выводит на консоль информацию о книге(строки author, genre и тд)(при помощи std::cout)
в `void PrintBookImp(std::ostream& out_stream)` функция выводит на консоль только строку впечатления (при помощи std::cout)

3.3 Файл LinkedList.h

[h] В файле LinkedList.h, который является заголовочным я подключила :

- iostream - часть стандартной библиотеки C++.
  Для ввода/вывода данных.
- fstream - часть стандартной библиотеки C++.
  Для работы с файлами, в данном случае для чтения из файла.
- string - класс из стандартной библиотеки C++.
  Для работы со строками.
- windows.h - заголовочный файл, предоставляет доступ к различным функциям Windows API.



```
#pragma once
#include <iostream>
#include <fstream>
#include <string>
#include <windows.h>
#include "MyBook.h"
using namespace std;

struct Node
{
    MyBook* data;
    Node* next;
    Node(MyBook* data, Node* next = nullptr) : data(data), next(next) {}
    Node(const Node& node) : data(node.data), next(nullptr) {}
    Node(Node&& node) noexcept : data(nullptr), next(nullptr)
    {
        data = node.data;
        next = node.next;
        node.data = nullptr;
        node.next = nullptr;
    }
    ~Node()
    {
        data = nullptr;
        next = nullptr;
    }
};
```

Рис. 14:

- MyBook.h - заголовочный файл, предоставляет доступ к файлу, в котором описаны методы для работы с книгами.

- pragma once - во избежания двойного включения данного заголовочного файла в проекте.

Потом была создана структура Node, которая представляет узел в нашем списке (односвязном).
В поле data стоит указатель на объект Mybook(т.е данные берем оттуда).
На следующей строке в поле next стоит указатель на Node (т.е на следующий узел в списке)
Конструктор Node принимает данные указатели и инициализирует соответствующие поля.
Конструктор копирования Node соответственно копирует данные из существующих узлов.
Конструктор перемещения Node перемещает указатели на данные и следующий узел из одного объекта в другой, а затем обнуляет указатели в переданном объекте.
Деструктор освобождает память, говоря что поля data и next пустые.

Рис. 15:

Далее создается класс LinkedList. В public создается конструктор LinkedList(), в котором инциализируется head (пустой строкой). Деструктор освобождает память с помощью функции dispose().(она объявлена в private)
Потом мы также объявляем функцию int Length(), которая считает длину собственно.
Потом мы пишем функцию bool IsEmpty(), которая возвращает значение true, если head равно nullptr.
Также прописывается функция bool IndexValid(int index), она проверяет вышло ли значение index за диапазон (он должен лежать в пределах от нуля (включительно) и до значения Length()). Если все хорошо, то он возвращает true.
Потом мы объявляем две функции void InsertNode(int index, Node* node) и Node* ExtractNode(int index), которые добавляют элемент и удаляют.
Далее мы сделали указатель на голову (head)
Как я уже писала ранее в private мы объявляем функцию dispose(), которая помогает очищать память.

3.4 Файл LinkedList.cpp

В данном файле мы поробно прописываем функции, которые мы объявляли в LinkedList.h.

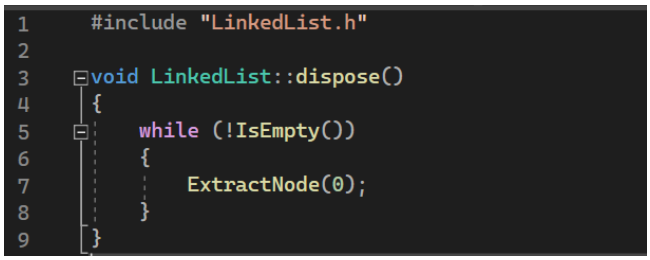


Рис. 16:

Функция void dispose() В данной фукции в цикле while (пока еще функция bool IsEmpty() не возвращает true), то с помощью ExtractNode(0), мы каждый раз из списка удаляем нулевой узел, пока он не очистится.

Рис. 17:

Функция void InsertNode(int index, Node* node) Данная функция добавляет узел в список на место указанного индекса.
Сначала мы должны проверить является ли голова списка пустой или индекс меньше или равне 0.Если это так, то мы говорим, что следующим узлом будет head, а на месте head мы добавляем наш узел.
Потом мы проверяем, что индекс находикся в обусловленном диапазоне (с помощью функции IndexValid(index)). Если это так, то сначала мы говорим, что указатель на временный узел tmp равен head. Потом в цикле for мы говорим, что пока не дойдем до указателя на узел index - 1 и пока следующий узел после временного не равно nullptr, мы смещаем указатель tmp на 1 вперед.
Потом мы говорим, что указатель на узел, следующий после узла, который

мы хотим вставить равен указателю на следующий узел после tmp. Потом мы говорим, что указатель на узел, следующий после tmp равен нашему узлу node. Теперь узел вставлен между tmp и tmp->next.
Иначе мы прописываем, что указатель на узел tmp равен голове. И в цикле while, пока указатель на следующий узел не равен nullptr, то мы смещаем tmp на 1 вперед.
Потом мы прописываем, что указатель на узел, следующий после tmp равен узлу node(который мы хотим вставить). А указатель на следующий узел после node равен nullptr.

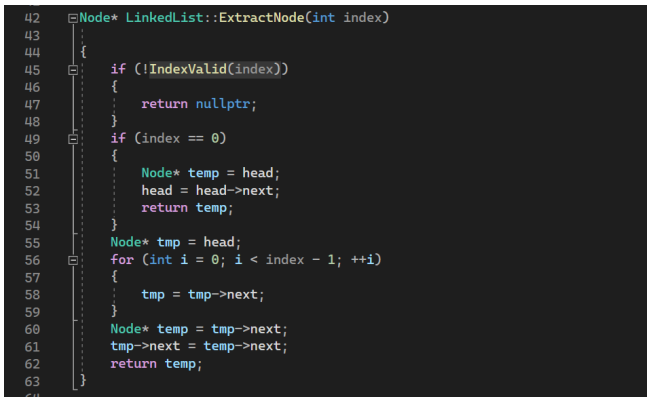
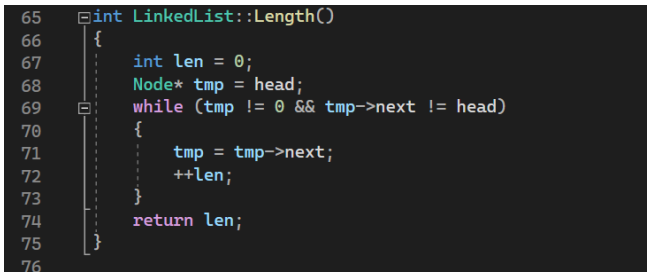


Рис. 18:

функция Node* ExtractNode(int index) Данная функция удаляет из списка узел, индекса который мы хотим, и возвращает его. Для начала мы проверяем является ли индекс,не лежащим в нашем диапазоне. Если это так, то мы возвращаем nullptr.
Потом если индекс равен 0, то мы пишем , что указатель на temp равен head.Новый head будет равен указателю на следующий узел после head. Потом мы возвращаем значение temp.
Если первые два условия оказываются не подходят, то указатель на временный узел tmp равен head. Потом в цикле for мы говорим, что пока не дойдем до указателя на узел index - 1 и пока следующий узел после временного не равно nullptr, мы смещаем указатель tmp на 1 вперед.
Потом мы пишем, что указатель temp равен указателю на узел следующий после tmp. Потом данный указатель(tmp->next) перенаправляем на указатель на узел, следующий после temp и возвращаем temp.



```
65  int LinkedList::Length()
66  {
67      int len = 0;
68      Node* tmp = head;
69      while (tmp != 0 && tmp->next != head)
70      {
71          tmp = tmp->next;
72          ++len;
73      }
74      return len;
75  }
76
```

Рис. 19:

Функция int Length() Функция, которая считает длину списка.
Сначала мы объявляем новую переменную len типа int равную 0.Указатель на временный узел tmp равен head. Потом в цикле while (пока tmp не равен нулю или указатель на следующий после tmp не равен head) смещаем указатель tmp на 1 и прибавляем к len + 1. После цикла возвращаем значение len.

3.5 Файл Pet.cpp

Это основной файл, благодаря котрому мы можем пользоваться приложением. Тут мы подключаем такие классы и библиотеки:

- iostream - часть стандартной библиотеки C++.
  Для ввода/вывода данных.
- fstream - часть стандартной библиотеки C++.
  Для работы с файлами, в данном случае для чтения из файла.
- string - класс из стандартной библиотеки C++.
  Для работы со строками.
- windows.h - заголовочный файл, предоставляет доступ к различным функциям Windows API.
- LinkedList.h - заголовочный файл, предоставляет доступ к файлу, в которой описаны методы работы односвязного списка.

Сначала мы создаем класс BoookLibrary. В public (там и есть только public). Потом с помощью LinkedList мы создаем переменную books, которая будет односвязным списком.
Далее мы прописываем строку `library_file_name`, которая будет хранить название файла, с котором мы работаем.

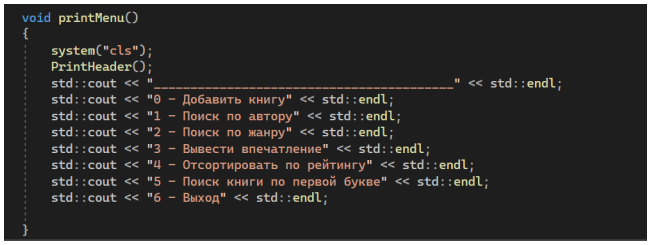
```
class BoookLibrary
{
public:
    LinkedList books;
    string library_file_name = "settings.txt";

    void PrintHeader()
    {
        std::cout << "######  #       #  ########      " << std::endl;
        std::cout << "#     # #       #     #  " << std::endl;
        std::cout << "#     # #       #     # " << std::endl;
        std::cout << "#     # #       #     # " << std::endl;
        std::cout << "#     # #     # #     #  " << std::endl;
        std::cout << "#     # #     # #     #  " << std::endl;
        std::cout << "######  ####### #     # " << std::endl;
        std::cout << " " << std::endl;
        std::cout << "Book Library " << std::endl;


    }
```

Рис. 20:

Функция void PrintHeader() Данная функция выводит на консоль название моей программы.Сделано чисто для красоты.



```
void printMenu()
{
    system("cls");
    PrintHeader();
    std::cout << "_____________________________________" << std::endl;
    std::cout << "0 - Добавить книгу" << std::endl;
    std::cout << "1 - Поиск по автору" << std::endl;
    std::cout << "2 - Поиск по жанру" << std::endl;
    std::cout << "3 - Вывести впечатление" << std::endl;
    std::cout << "4 - Отсортировать по рейтингу" << std::endl;
    std::cout << "5 - Поиск книги по первой букве" << std::endl;
    std::cout << "6 - Выход" << std::endl;

}
```

Рис. 21:

Функция void printMenu() Данная функция выводит меню на экран. Сначала, с помощью функции system("cls") происходит очистка консоли перед выводом нового меню.Потом с помощью функции void PrintHeader() выводим логотип.Затем мы выводим объяснения пользователю, что будет, если нажать ту или иную клавишу.

Функция void LoadLibraryFromFile() Сначала мы объявляем обьект std::ifstream in для работы с файлом. Потом мы открывем файл settings.txt . Потом в цикле while (пока файл отрыт и пока он не закончился)
Потом в цикле мы создаем массив объекта MyBook под именем book. Потом мы создаем узел списка под названием `book_node` с указателем на обьект book.
Потом для обьекта book мы вызываем функцию readbook(in), чтобы он прочитал информацию о книге из нашего файла
Далее мы проверяем пуста ли поле author, если ее нет, мы удаляем обьект

```
46  void LoadLibraryFromFile()
47  {
48      std::ifstream in;
49      in.open(library_file_name);
50      while (in.is_open() && !in.eof())
51      {
52          MyBook* book = new MyBook();
53          Node* book_node = new Node(book);
54          book->readbook(in);
55          if (book->author.empty())
56          {
57
58              delete book;
59          }
60          else
61          {
62              books.InsertNode(0, book_node);
63          }
64      }
65      in.close();
66  }
```

Рис. 22:

book.
Если же поле author не пусто, то с помощью функции InsertNode(0, `book_node`) то мы вставляем в список book узел `book_node` на позицию 0.
Далее закрываем файл.

```
68  int choice(int choice)
69  {
70      switch (choice)
71      {
72      case 0:
73      {
74          MyBook* book = new MyBook();
75          Node* book_node = new Node(book);
76          book->inputbook();
77          books.InsertNode(0, book_node);
78          book->writebook(library_file_name);
79          book->LibraryOut();
80          std::cout << "Книги были записаны в файл dlitbooks.txt";
81          break;
82      }
83      case 1:
84      {
85          std::string author;
86          std::cout << "Введите автора : ";
87          std::cin.ignore();
88          getline(std::cin, author);
89          FindBookByAuthor(author);
90          break;
91      }
```

Рис. 23:

Функция int choice() Она достаточно длинная, поэтому изображения я поделю на части.
Внутри есть оператор switch (choice) и в зависимости от переменной choice. Прописываются case, которые работают в зависимости от выбора пользователя (т.е какую цифру из меню пользователь выберет)
Если пользователь выберет case 0, то:
мы создаем массив объекта MyBook под именем book. Потом мы создаем

узел списка под названием `book_node` с указателем на обьект book.
Потом для обьекта book вызываем функцию inputbook()(Функция, через которую пользователь вводит данные, необходимые для книги).
Далее помощью функции InsertNode(0, `book_node`) то мы вставляем в список book узел `book_node` на позицию 0.
Далее с помощью функций `writebook(library_file_name) и LibraryOut()` мы записываем данные этого узла в соответствующие файлы.
Если пользователь выбрал case 1:
Мы создаем строку author и просим ввести туда данные. Далее с помощью функции FindBookByAuthor(author) (она прописана ниже) мы в файле находим данного автора и выводим на консоль его книги.
Для выхода пишем break.
Если пользователь выбрал case 2:

```
92          case 2:
93          {
94              std::string genre;
95              std::cout << "Введите жанр: ";
96              std::cin.ignore();
97              getline(std::cin, genre);
98              FindBookByGenre(genre);
99              break;
100         }
101
102         case 3:
103         {
104             std::string book_name;
105             std::string author;
106             std::cin.ignore();
107             std::cout << "Введите название книги: ";
108             getline(std::cin, book_name);
109             std::cout << "Введите автора : ";
110             getline(std::cin, author);
111             OutImpression(book_name, author);
112             break;
113         }
```

Рис. 24:

Мы создаем строку genre и просим ввести туда данные. Далее с помощью функции FindBookByGenre(genre) (она прописана ниже) мы в файле находим данный жанр и выводим на консоль книги этого жанра.
Для выхода пишем break.
Если пользователь выбрал case 3:
Мы создаем строки `book_name`,author и просим ввести туда данные. Далее с помощью функции `OutImpression(book_name, author)` (она прописана ниже) мы в файле находим данного автора и книгу.Выводим на консоль впечатление.
Для выхода пишем break.
Если пользователь выбрал case 4:
Мы создаем строку rating и просим ввести туда данные. Далее с помощью функции FindBookByRating(rating) (она прописана ниже) мы в файле находим данного рейтинг и выводим на консоль книги с такой оценкой.
Для выхода пишем break.

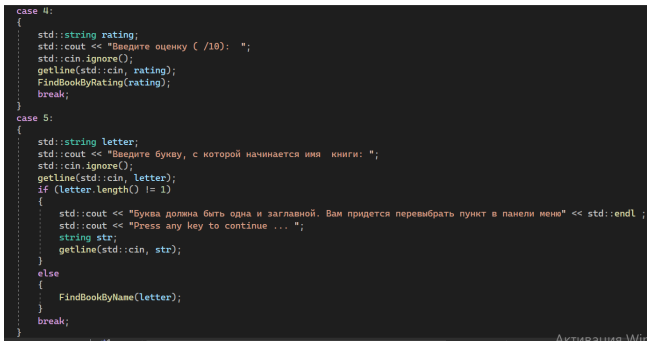


Рис. 25:

Если пользователь выбрал case 5:
Мы создаем строку letter И просим ввести туда данные.Усли длиина строки не равна 1 ,то говорим,что данные введены неверно.Иначе с помощью функции FindBookByName(letter) в файле находим книги с такой буквой и выводим их.
Если пользователь выбрал case 6 возвращаем 0.

```
case 6:
    return 0;
}
```

Рис. 26:

```
131  void FindBookByAuthor(string Author)
132  {
133
134      ostream& objOstream = cout;
135      Node* tmp = books.head;
136      while (tmp != nullptr)
137      {
138          if (tmp->data->author.find(Author) != std::string::npos)
139          {
140              tmp->data->PrintBookInfo(objOstream);
141          }
142          tmp = tmp->next;
143      }
144
145      std::cout << "Press any key to continue ... ";
146      string str;
147      getline(std::cin, str);
148  }
149
```

Рис. 27:

Функция void FindBookByAuthor(string Author) Сначала создаем ссылку на стандартный поток вывода.
Потом мы создаем указатель tmp на голову списка. Далее в цикле, по-

ка указатель не равен nullptr, если мы с помощью функции find() находим поле author и оно не равно std::string::npos, мы вызываем функцию PrintBookInfo(objOstream). Далее выходим из проверки if и перемещаем указатель на следующий узел.
После цикла выводим сообщение о продолжении и ожидаем нажатие клавиши для продолжения выполнения программы.

```
150    void FindBookByGenre(string Genre)
151    {
152
153        ostream& objOstream = cout;
154        Node* tmp = books.head;
155        while (tmp != nullptr)
156        {
157            if (tmp->data->genre.find(Genre) != std::string::npos)
158            {
159                tmp->data->PrintBookInfo(objOstream);
160            }
161            tmp = tmp->next;
162        }
163
164        std::cout << "Press any key to continue ... ";
165        string str;
166        getline(std::cin, str);
167    }
168
```

Рис. 28:

Функция void FindBookByGenre(string Genre) Сначала создаем ссылку на стандартный поток вывода.
Потом мы создаем указатель tmp на голову списка. Далее в цикле, пока указатель не равен nullptr, если мы с помощью функции find() находим поле genre и оно не равно std::string::npos, мы вызываем функцию PrintBookInfo(objOstream). Далее выходим из проверки if и перемещаем указатель на следующий узел.
После цикла выводим сообщение о продолжении и ожидаем нажатие клавиши для продолжения выполнения программы.

```
169    void OutImpression(string Book_name, string Author)
170    {
171        ostream& objOstream = cout;
172        Node* tmp = books.head;
173        while (tmp != nullptr)
174        {
175            if (tmp->data->book_name.find(Book_name) != std::string::npos && tmp->data->author.find(Author) != std::string::npos)
176            {
177                tmp->data->PrintBookImp(objOstream);
178            }
179            tmp = tmp->next;
180        }
181
182        std::cout << "Press any key to continue ... ";
183        string str;
184        getline(std::cin, str);
185    }
```

Рис. 29:

`Функция void OutImpression(string Book_name, string Author)` Сначала создаем ссылку на стандартный поток вывода.
Потом мы создаем указатель tmp на голову списка. Далее в цикле, пока

указатель не равен nullptr, если мы с помощью функции find() находим поля author и genre и они не равны std::string::npos, мы вызываем функцию PrintBookInfo(objOstream). Далее выходим из проверки if и перемещаем указатель на следующий узел.
После цикла выводим сообщение о продолжении и ожидаем нажатие клавиши для продолжения выполнения программы.

```
187     void FindBookByRating(string Rating)
188     {
189
190         ostream& objOstream = cout;
191         Node* tmp = books.head;
192         while (tmp != nullptr)
193         {
194             if (tmp->data->rating.find(Rating) != std::string::npos)
195             {
196                 tmp->data->PrintBookInfo1(objOstream);
197             }
198             tmp = tmp->next;
199         }
200
201         std::cout << "Press any key to continue ... ";
202         string str;
203         getline(std::cin, str);
204     }
```

Рис. 30:

Функция void FindBookByRating(string Rating) Сначала создаем ссылку на стандартный поток вывода.
Потом мы создаем указатель tmp на голову списка. Далее в цикле, пока указатель не равен nullptr, если мы с помощью функции find() находим поле rating и оно не равно std::string::npos, мы вызываем функцию PrintBookInfo(objOstream). Далее выходим из проверки if и перемещаем указатель на следующий узел.
После цикла выводим сообщение о продолжении и ожидаем нажатие клавиши для продолжения выполнения программы.

```
void FindBookByName(string Letter)
{
    ostream& objOstream = cout;
    Node* tmp = books.head;
    while (tmp != nullptr)
    {
        if (tmp->data->book_name.find(Letter) == 0)
        {
            tmp->data->PrintBookInfo1(objOstream);
        }
        tmp = tmp->next;
    }

    std::cout << "Press any key to continue ... ";
    string str;
    getline(std::cin, str);
}
```

Рис. 31:

Функция void FindBookByName(string Letter) Сначала создаем ссылку на стандартный поток вывода.
Потом мы создаем указатель tmp на голову списка. Далее в цикле, пока указатель не равен nullptr, если мы с помощью функции find() в поле `book_name` находим поле Letter и оно равно 0, мы вызываем функцию PrintBookInfo(objOstream). Далее выходим из проверки if и перемещаем указатель на следующий узел.
После цикла выводим сообщение о продолжении и ожидаем нажатие клавиши для продолжения выполнения программы.

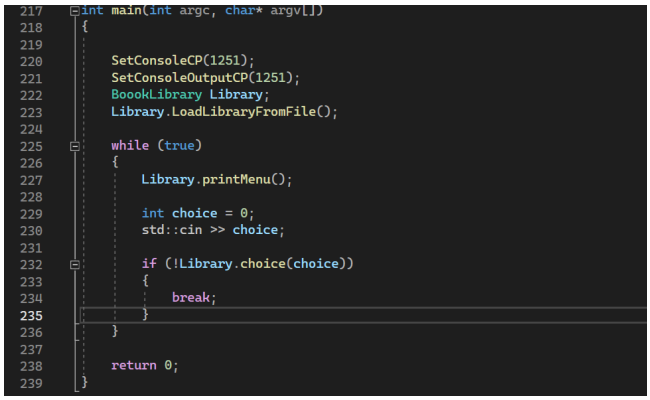


Рис. 32:

int main(int argc, char* argv[]) Сначала, я прописываю SetConsoleCP(1251), SetConsoleOutputCP(1251), чтобы у программы не было проблем с русскими буквами.
Создаем объект класса BoookLibrary,чтобы работать с библиотекой книг.
Далее вызываем функцию LoadLibraryFromFile() объекта Library,чтобы загружать информацию о книгах из нашего файла.
Потом в цикле while (пока не закончим программу) вызываем функцию printMenu().
Там же создаем переменную choice типа int и просим пользователя ввести его значения. Если функция choice() не работает (возвращает false), мы пишем break и выходим из цикла.

4 Вывод

Благодаря моему проекту, я намного лучше поняла работу односвязного списка.Узнала о новых функциях, таких как erase и find.Намного лучше

усвоила работу с файлами и поняла преимущество ссылок на поля и обьекты.
Еще я очень рада, что данный проект будет использован мной, для прочитанных книг.Это поможет мне лучше излагать мысли, но это побочный эффект.