Лабораторная работа №6.

Текст программы к заданию 2:

```
1.  #include <iostream>
2.  #include <string>
3.  #include <vector>
4.
5.  struct Book {
6.      unsigned long int _URN;
7.      std::string _title;
8.      std::string _author;
9.      unsigned  int _year;
10.     unsigned  int _count;
11.     Book() {}
12.     Book(unsigned long int URN, std::string title, std::string author, unsigned int year,
    unsigned int count) {
13          _URN = URN;
14.         if (title.length() <= 50) {
15.             _title = title;
16.         } else {
17.             std::cout << "Title is too long";
18.             throw new std::exception();
19.         }
20.         if (author.length() <= 20) {
21.             _author = author;
22.         } else {
23.             std::cout << "Author's name is too long";
24.             throw new std::exception();
25.         }
26.         _year = year;
27.         _count = count;
28.     }
29.     int input() {
30.         std::cout << "Input book's united registration number:\n";
31          std::cin >> _URN;
32..        std::cout << "Input book's title:\n";
33.         std::cin.ignore();
34.         std::getline(std::cin, _title, '\n');
35.         if (_title.length() > 50) {
36.             std::cout << "Error, try again!\n";
37.             return -1;
38.         }
39.         std::cout << "Input book's author:\n";
40.         std::getline(std::cin, _author, '\n');
41.         if (_author.length() > 20) {
42.             std::cout << "Error, try again!\n";
```

```
43.             return -1;
44.         }
45.         std::cout << "Input book's year:\n";
46.         std::cin >> _year;
47.         std::cout << "Input book's count:\n";
48.         std::cin >> _count;
 49         return 0;
50.     }
51.     void output() {
52.         std::cout << "Current URN: " << _URN << '\n';
53.         std::cout << "Current title: " << _title << '\n';
54.         std::cout << "Current author: " << _author << '\n';
55.         std::cout << "Current year: " << _year << '\n';
56.         std::cout << "Current count: " << _count << '\n';
57.     }
58. };
59.
60. struct Library {
61.     std::vector<Book> _container;
62.     int _N;
63.     Library() {}
64.     Library (int N) {
65.         _container = std::vector<Book>(N);
66.         for (int i = 0; i < N; ++i) {
67.             int result = _container.at(i).input();
 68             if (result == -1) {
69.                 --i;
70.             }
71.         }
72.         _N = N;
73.     }
74.     bool increase(const unsigned long int &URN) {
75.         for (int index = 0; index < _N; ++index) {
76.             if (_container[index]._URN == URN) {
77.                 _container[index]._count += 100;
78.                 break;
79.             }
80.         }
81.     }
82.     void print() {
83.         for (auto it = _container.begin(); it != _container.end(); ++it) {
84.             std::cout << "=========================================\n";
85.             it->output();
86.             std::cout << "=========================================\n";
87.         }
88.     }
89. };
90.
```

```
91.  int main() {
92.      int num;
93.      std::cin >> num;
94.      Library lib(num);
95.      system("cls");
96.      lib.print();
97.      lib.increase(1);
98.      std::cout << "After:\n";
99.      lib.print();
100.     return 0;
101. }
102.
```

Оценка характеристик программы

В программе представлено 2 класса: Book и Library.

Book:

- Book();
- Book(…);
- input();
- output();

Library:

- Library();
- Library(int N);
- increase();
- print();

Итого: по 4 метода в каждом классе,

$$WMC_{Book} = 0 + 15 + 20 + 5 = 40,$$
$$WMC_{Library} = 0 + 8 + 6 + 5 = 19$$

Отсюда видно, что основным классом является класс Book, а Library- это лишь «обёртка» для него с некоторыми функциями.

Теперь определим значение метрики NM для каждого класса:

$$NM_{Book} = 4;$$

$$NM_{Library} = 4;$$

По числу методов нельзя сказать какой класс сложнее.

Определим связность между классами объектов CBO, которая численно определяется по количеству классов, с которыми связан анализируемый класс.

$$CBO_{Book} = 8;$$

$$CBO_{Library} = 7;$$

Из анализа значений CBO, класс Library менее чувствителен к изменениям, чем класс Book, то есть его проще изменить, добавив или удалив что-то.

Теперь определим количество откликов на класс RFC:

$$RFC_{Book} = 4 + 5 = 9,$$

$$(4 \text{ метода и в них вызывается } 5 \text{ методов других классов})$$

$$RFC_{Library} = 4 + 7 = 11.$$

RFC_{Book} и $RFC_{Library}$ не сильно различаются, а потому их тестирование не особо будут различаться по сложности.

Теперь посчитаем значение метрики LCOM (отсутствия сцепления в методах класса).

Для Book:

- Book() не делает абсолютно ничего;
- Book(…) обращается ко всем полям класса(5 штук);
- input() обращается ко всем полям класса(5 штук);
- output() обращается ко всем полям класса(5 штук);

Таким образом для класса Book нет ни одной пары методов, которые не обращались бы к общех полей класса, и насчитывается 4 метода, которые обращаются с общим полям (всем):
$LCOM_{Book} = 0 - 4 = 0$, т.к. при $LCOM < 0$, мы приравниваем его к 0.

Для Library:

- Library() не делает ничего;
- Library(int N) обращается ко всем полям Library;
- increase() обращается ко всем полям Library;
- print() обращается ко всем полям Library;

Получаем такую же ситуацию, как и с классом Book, что говорит о том, что все классы (как же их много…) в программе тесно связаны друг с другом

Подводя итог можно сказать, что классы довольно просты, неплохо масштабируемы, слабо чувствительны к изменениям(сломаются, только если постараться), несложно тестируются и тесно связаны друг с другом