# **1.По текстовому описанию задачи построена диаграмма прецедентов.**

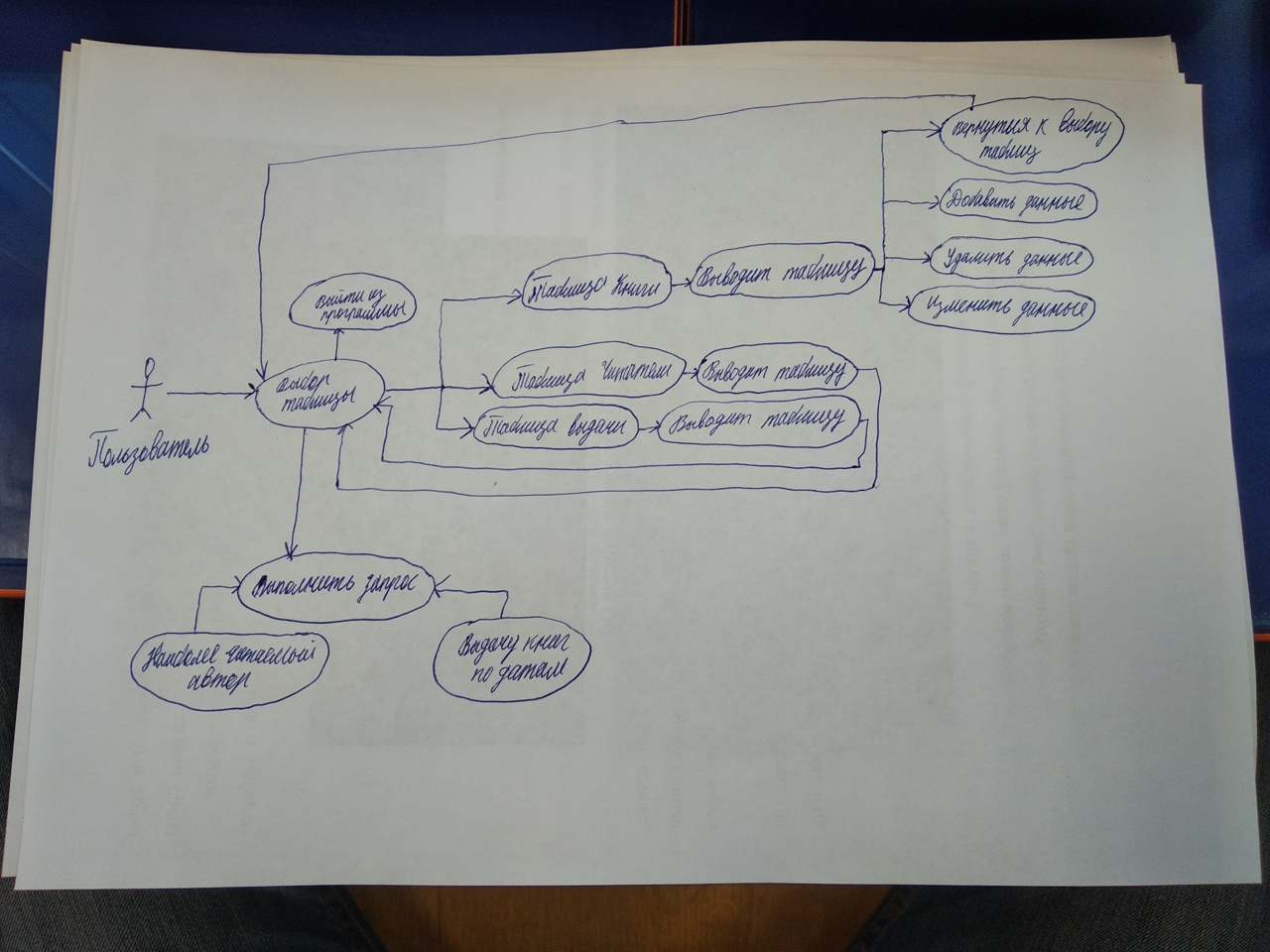


Рисунок 1

# **2. По описанию и диаграмме прецедентов построить диаграмму классов.**

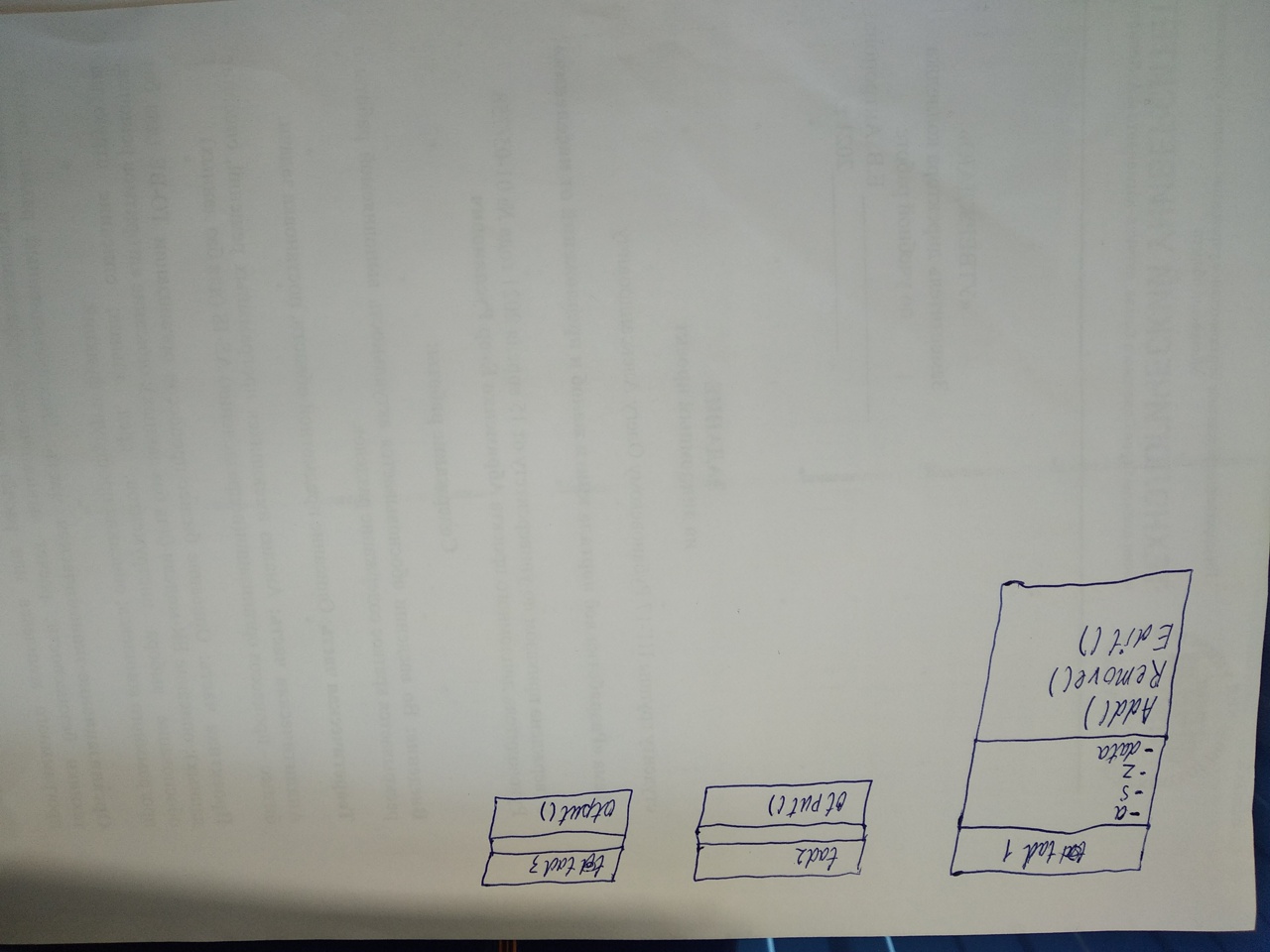


Рисунок 2.

# **3. Разработка определения классов по диаграмме.**

Класс tad1 создан для таблицы Книги. В нем созданы функции для записи, обработки и удаления данных в этой таблице.

Класс tad2 создан для таблицы Читатели. В нем создана функция для вывода данных их этой таблице.

Класс tad3 создан для таблицы Читатели. В нем создана функция для вывода данных их этой таблице.

# **4. Проект.**

Листинг 1.

from statistics import mode

class tad1:

def Add():

print('Введите данные через пробел')

a = input()

s = '\n'+a

f.write(s)

def Edit():

print('Введите индекс строки которую хотите редактировать')

z = int(input())

print("Напишите данные которыми хотите заменить существующие")

a = input()+'\n'

data = 0

with open('tad.txt','r') as f:

data = f.readlines()

with open('tad.txt', 'w') as f:

for line in data:

if line.strip('\n') != data[z-1].strip('\n'):

f.write(line)

else:

f.write(a)

def Remowe():

print('Введите индекс строки которую хотите удалить')

z = int(input())

data = 0

with open('tad.txt','r') as f:

data = f.readlines()

with open('tad.txt', 'w') as f:

for line in data:

if line.strip('\n') != data[z-1].strip('\n'):

f.write(line)

class tad2:

def otput():

print("код читателя|фамилия|имя|отчество|№ читательского билета")

f = open('tad1.txt', 'r')

print(\*f)

f.close()

class tad3:

def otput():

print("код читателя|книги|дата выдачи")

f = open('tad2.txt', 'r')

print(\*f)

f.close()

while True:

print('Выбирите таблицу\nКниги - 1\nЧитатели - 2\nВыдача - 3\nВыбирите запрос\nНаиболее читаемый автор - 4\n'

'Выдачу книг по датам - 5\nВыход - 6')

tad = int(input())

if tad == 1:

while True:

print('код книги|название|автор|жанр|год издания')

f = open('tad.txt','r+')

print(\*f)

print('Добавить - 1 \nРедактировать - 2 \nУдалить - 3 \nВыход - 4')

c = int(input())

if c == 1:

tad1.Add()

elif c == 2:

tad1.Edit()

elif c == 3:

tad1.Remowe()

elif c == 4:

break

elif tad == 2:

tad2.otput()

elif tad == 3:

tad3.otput()

elif tad == 4:

data = 0

z = []

x = []

with open('tad.txt', 'r') as f:

data = f.readlines()

for i in data:

z.append(i.split(' '))

for i in z:

x.append(i[2])

print('Наиболее читаемый автор = ', mode(x),'\n')

elif tad == 5:

print('Выдача книг по дадам')

data = 0

z = []

x = []

c = []

with open('tad2.txt', 'r') as f:

data = f.readlines()

for i in data:

z.append(i.split(' '))

for i in z:

x.append(i[1])

v = i[2]

x.append(v[0:-1])

if v[-1] == '\n':

print(i[1], v[0:-1])

else:

print(i[1], v)

print('\n')

elif tad == 6:

break

# **5. Показать отладку.**

1. Перейдем в программе на интересующую нас строку, начиная с которой будет начала отладки.
2. В главном меню выбираем Run -> Toggle Line Breakpoint или Run -> Toggle Temporary Line Breakpoint:

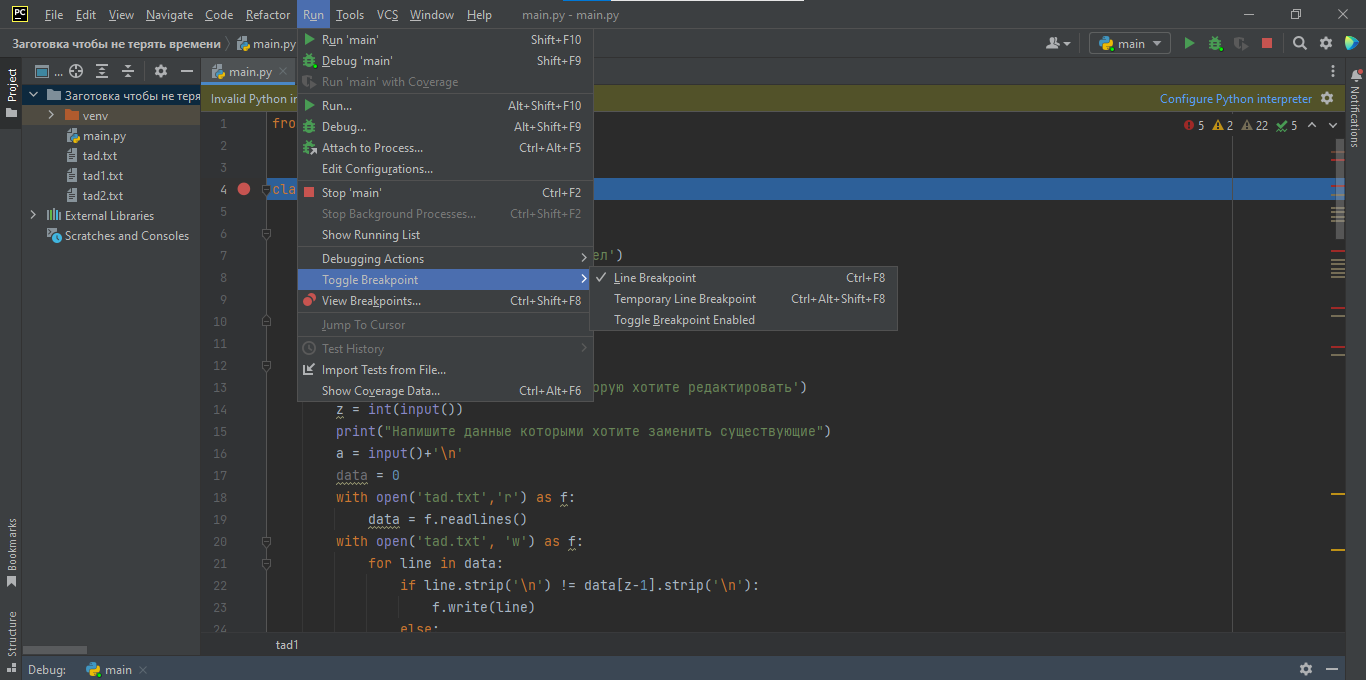


Рисунок 4. Отладка программы

1. Теперь рядом со строкой появилась красная жирная точка. Это точка останова. Теперь при выполнении программы в режиме отладки среда остановит её в этом месте, и можно будет узнать состояние программы.
2. Нажмем правой кнопкой на название файла с программой и выберем Debug:

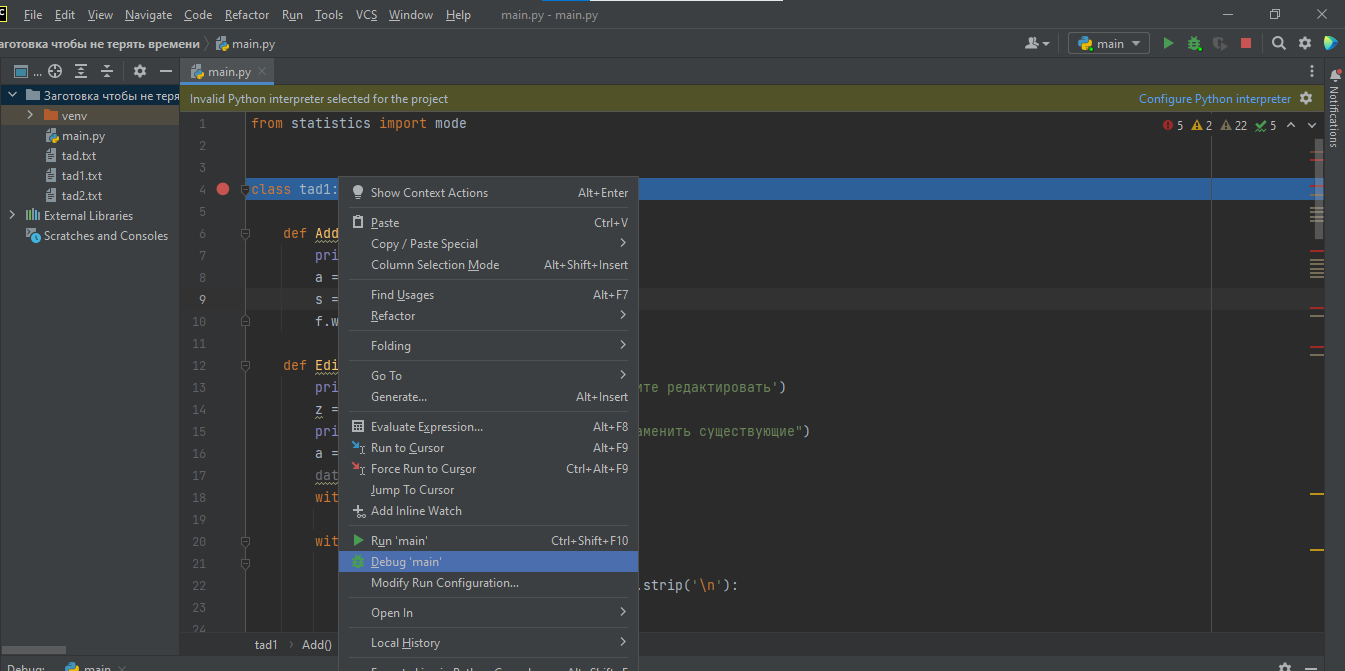


Рисунок 4. Использование функции Debug

1. Теперь программа запустилась и остановилась на указанной строке. Текущее положение интерпретатора Python в программе отмечается синей строкой. В нижней части экрана появилась вкладка отладки. Там виден список переменных, доступных из данной точки программы.
2. Для того, чтобы проследовать вслед за интерпретатором Python в класс (на строку 18), в главном меню выберем Run -> Debugging Actions-> Step Into, или нажмем клавишу F7.

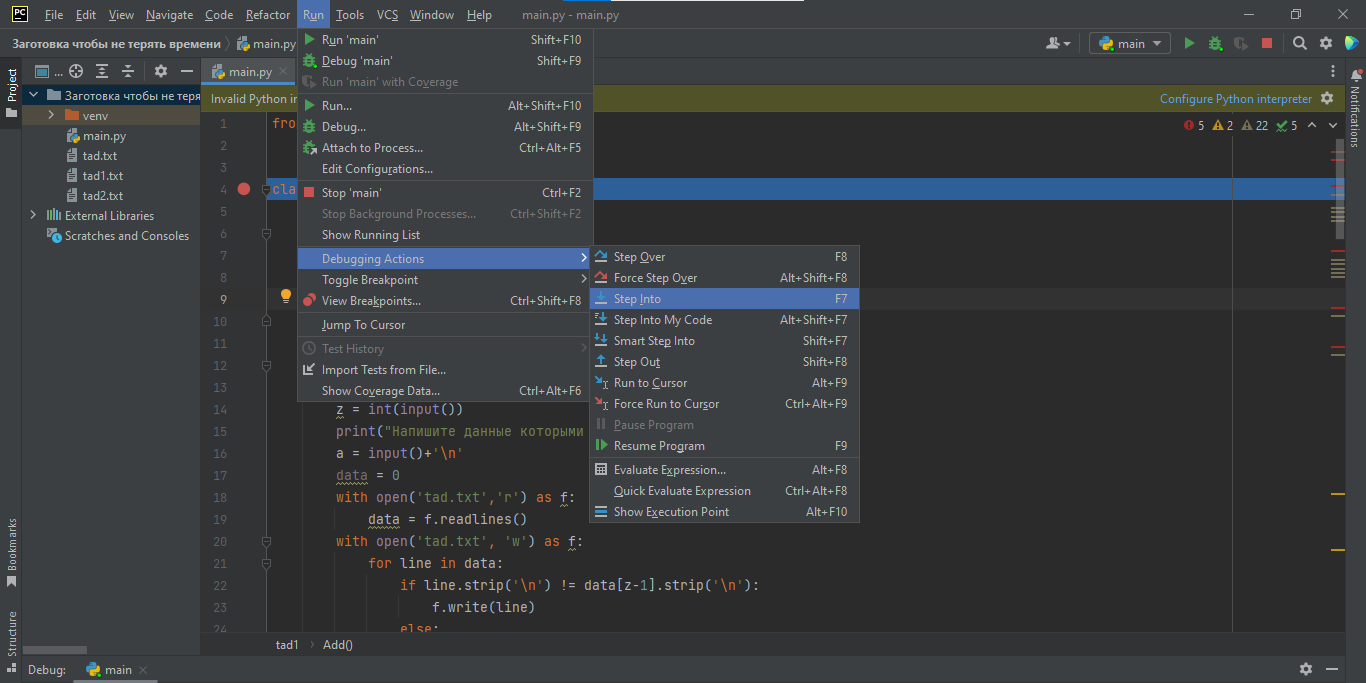


Рисунок 5. Использование функции Step Into

1. Как и ожидалось, интерпретатор переместил фокус своего внимания на строку 4, в классе tad1:
2. Чтобы сделать шаг вперёд, не заглядывая в используемые функции, используv команду Run -> Debugging Actions-> Step Over, или клавишу F8:

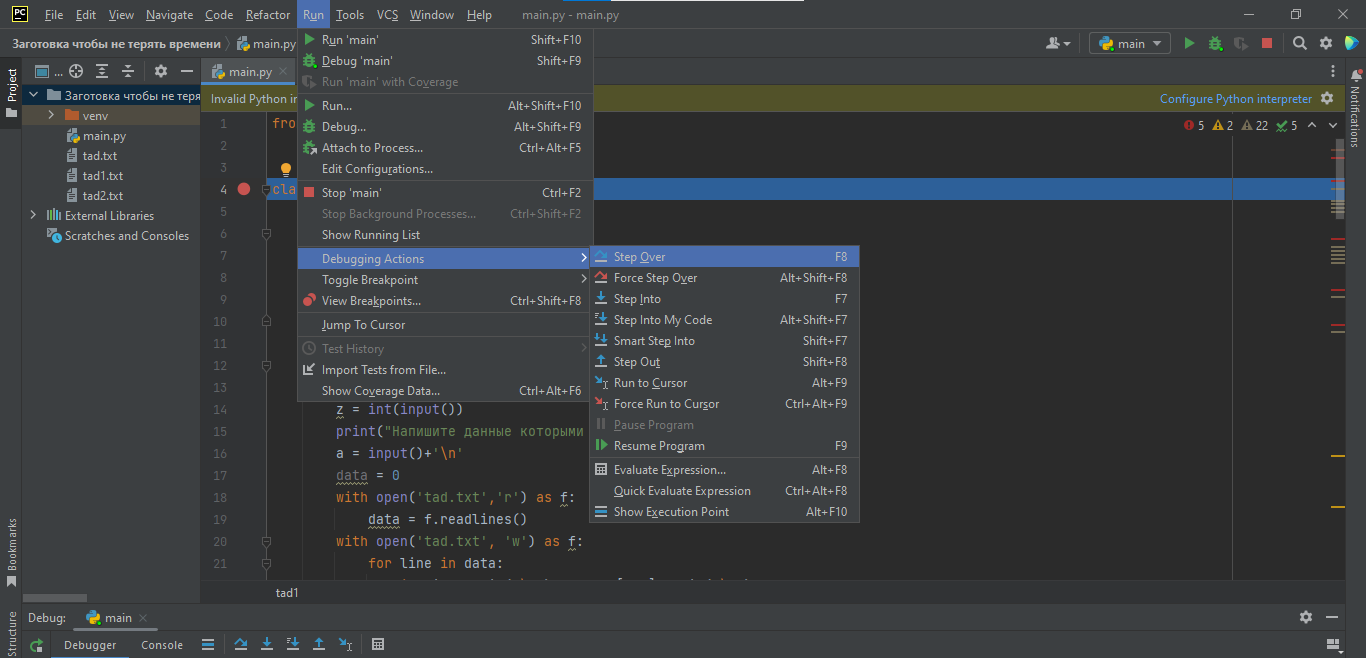


Рисунок 6. Использование функции Step Over

# **6.Тестовый набор и тестирование одного из модулей.**

1. Запускаем код, программа функционирует.

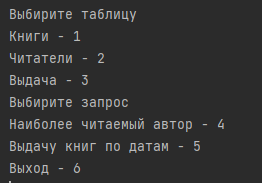


Рисунок 7. Главное меню

2. Добавление, изменение и удаление данных работает.

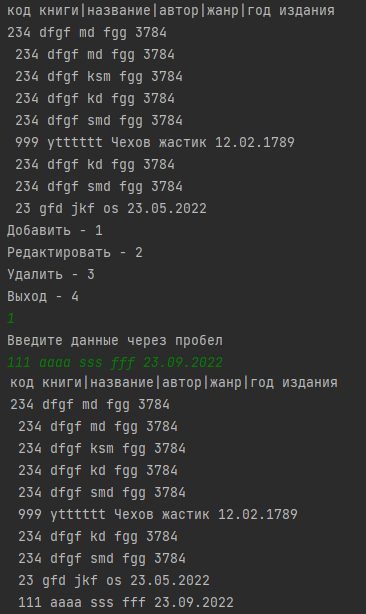


Рисунок 8. Добавление

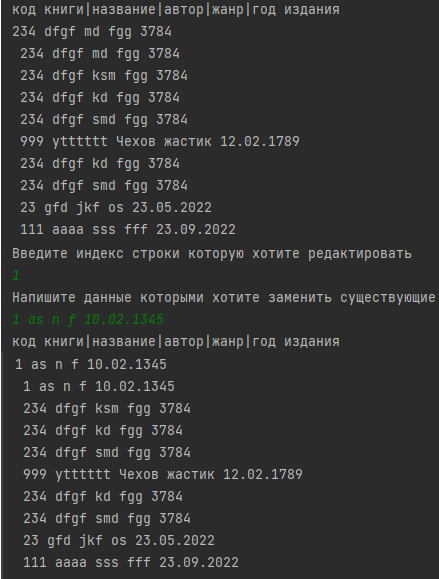


Рисунок 9. Редактирование

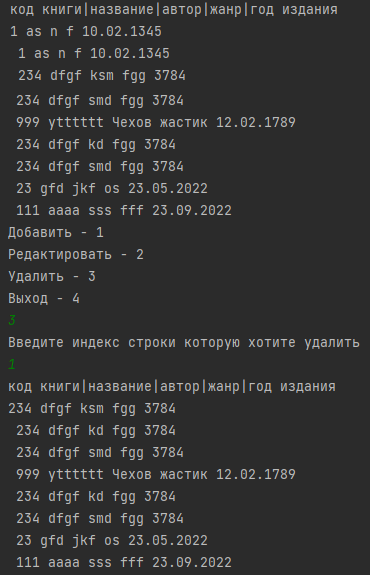


Рисунок 10. Удаление

3. Данные записываются в файл.

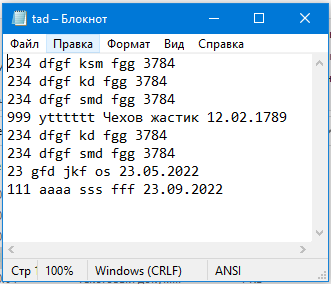


Рисунок 11. tad.txt

# **7. Определить значение временной сложности О для одного метода.**

Листинг 2. Функция по которой рассчитывается О.

def Edit():

print('Введите индекс строки которую хотите редактировать')

z = int(input())

print("Напишите данные которыми хотите заменить существующие")

a = input()+'\n'

data = 0

with open('tad.txt','r') as f:

data = f.readlines()

with open('tad.txt', 'w') as f:

for line in data:

if line.strip('\n') != data[z-1].strip('\n'):

f.write(line)

else:

f.write(a)

O(N)\*O(N)=O(N^2)