САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №0

по курсу «Алгоритмы и структуры данных»

Тема: Введение. Работа с файлами, тестирование

Выполнил:

Буй Тхук Хуен

К3139

Проверила:

Афанасьев А.В.

Санкт-Петербург

2024 г.

Содержание отчета

Задачи по варианту

TASK 1: Ввод-вывод

1. Задача a + b. В данной задаче требуется вычислить сумму двух заданных чисел. Вход: одна строка, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-10^9 \le a, b \le 10^9$. Выход: единственное целое число — результат сложения a + b.

- Листинг кода

```
a,b = map(int, input().split())
if -10**9 <= a <= 10**9 and -10**9 <= b <= 10**9:
    print(a+b)
else:
    print('Ввод неверен')
```

- Текстовое объяснение решения

Функция *map()* применяет функцию *int()* к каждому элементу списка, возвращаемого *split()*. Проверяю полученные числа с помощью двойного неравенства и вывожу сумму a+b. В противном случае проверю введённые числа.

- Результат работы кода

```
24 53
77
```

2. Задача $a + b^2$. В данной задаче требуется вычислить значение $a + b^2$. Вход: одна строка, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-10^9 \le a, b \le 10^9$.

Выход: единственное целое число — результат сложения $a + b^2$.

- Листинг кода

```
a,b = map(int, input().split())
if -10**9 <= a <= 10**9 and -10**9 <= b <= 10**9:
    print(a+b*b)
else:
    print('Ввод неверен')
```

- Текстовое объяснение решения.

Функция *map()* применяет функцию *int()* к каждому элементу списка, возвращаемого *split()*. Проверяю полученные числа с помощью двойного неравенства и вывожу сумму a+b^2. В противном случае проверю введённые числа.

- Результат работы кода

```
3 6
39
```

3. Выполните задачу a + b с использованием файлов.

• Имя входного файла: input.txt

• Имя выходного файла: output.txt

• Формат входного файла. Входной файл состоит из одной строки, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-10^9 \le a, b \le 10^9$

• Формат выходного файла. Выходной файл единственное целое число — результат сложения a + b.

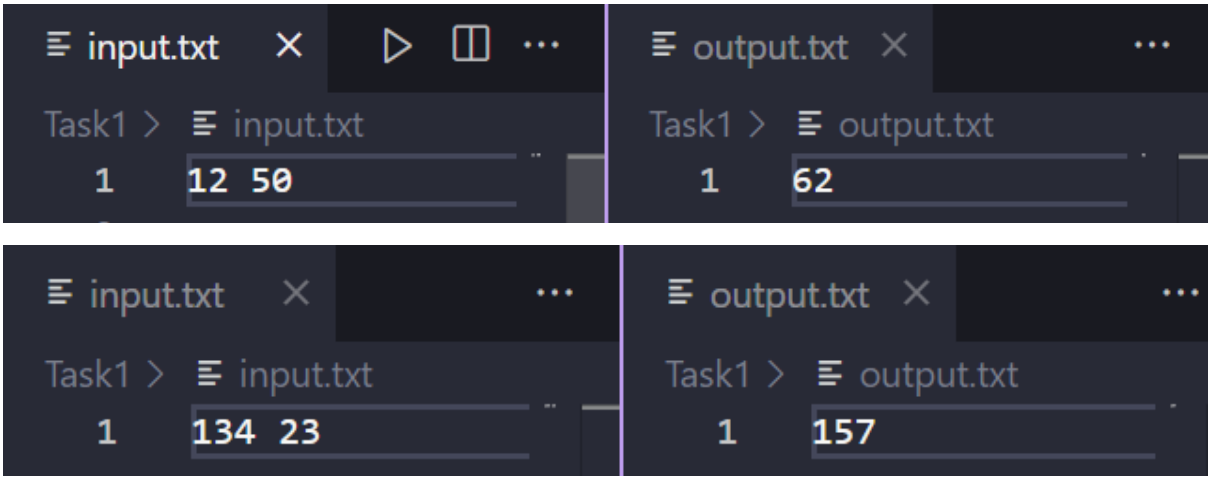
- Листинг кода

```
f1 = open('D:\\Python\\CTDL&GT\\Lab0\\Task1\\input.txt', 'r')
f2 = open('D:\\Python\\CTDL&GT\\Lab0\\Task1\\output.txt', 'w')
a, b = map(int, f1.readline().split())
if -10**9 <= a <= 10**9 and -10**9 <= b <= 10**9:
    f2.write(str(a+b))
else:
    print('Ввод неверен')
```

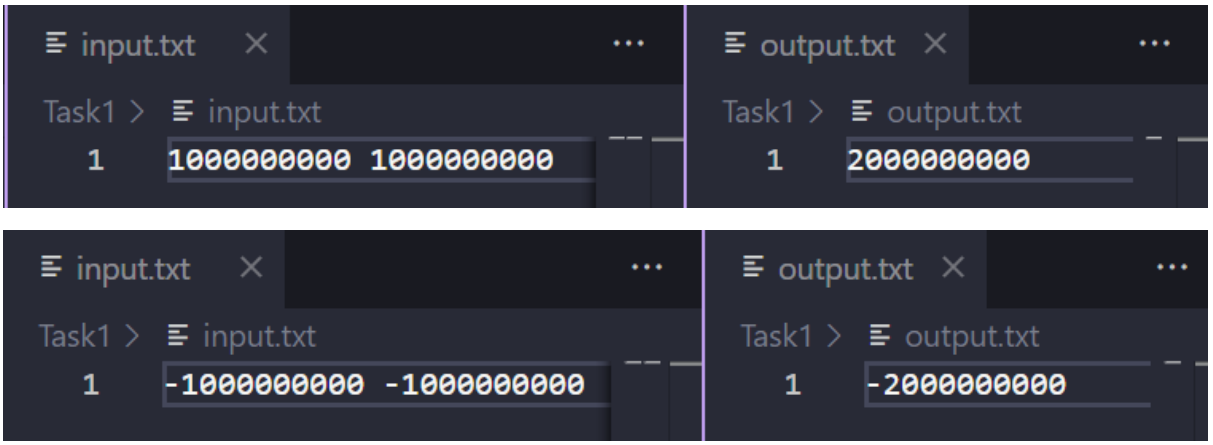
- Текстовое объяснение решения.

Открываю файлы для чтения (*input.txt*) и записи (*output.txt*) с помощью функции *open()*. Функция map() применяет функцию *int()* к каждому элементу списка, возвращаемого *split()*. Проверяю полученные числа с помощью двойного неравенства и записываю в файл сумму a+b. Закрываю файлы. В противном случае проверю введённые числа.

- Результат работы кода:



- Результат работы кода на максимальных и минимальных значениях:



Тест	Время выполнения (s)	Затраты памяти(bytes)
12 50	0.004897800041362643	84
134 23	0.0010449000401422381	84
-10^9 -10^9	0.0004544999683275819	88
10^9 10^9	0.005220899998676032	88

4. Выполните задачу a + b^2 с использованием файлов.

• Имя входного файла: input.txt

• Имя выходного файла: output.txt

• Формат входного файла. Входной файл состоит из одной строки, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-10^9 \le a, b \le 10^9$

• Формат выходного файла. Выходной файл единственное целое число — результат сложения $a + b^2$.

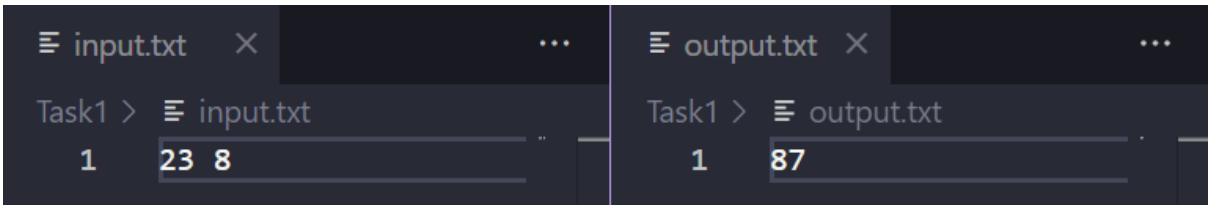
- Листинг кода

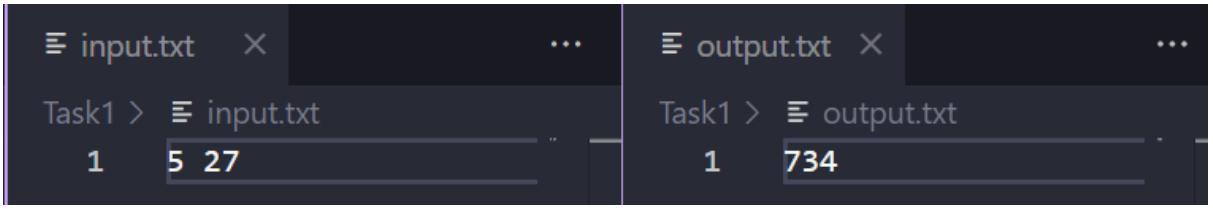
```
f1 = open('D:\\Python\\CTDL&GT\\Lab0\\Task1\\input.txt', 'r')
f2 = open('D:\\Python\\CTDL&GT\\Lab0\\Task1\\output.txt', 'w')
a, b = map(int, f1.readline().split())
if -10**9 <= a <= 10**9 and -10**9 <= b <= 10**9:
    f2.write(str(a+b*b))
else:
    print('Ввод неверен')
```

- Текстовое объяснение решения

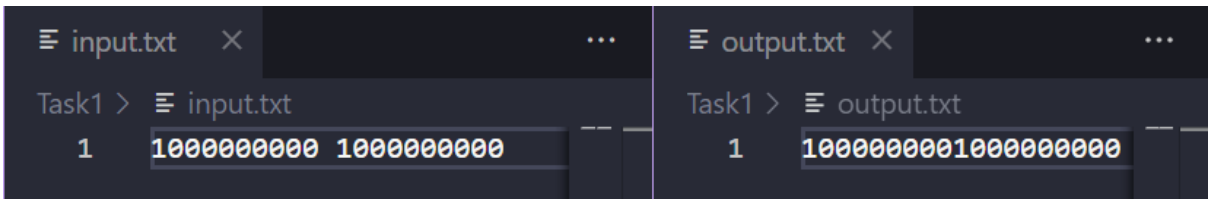
Открываю файлы для чтения (*input.txt*) и записи (*output.txt*) с помощью функции *open()*. Функция *map()* применяет функцию *int()* к каждому элементу списка, возвращаемого *split()*. Проверяю полученные числа с помощью двойного неравенства и записываю в файл сумму $a+b^2$. Закрываю файлы. В противном случае проверю введённые числа.

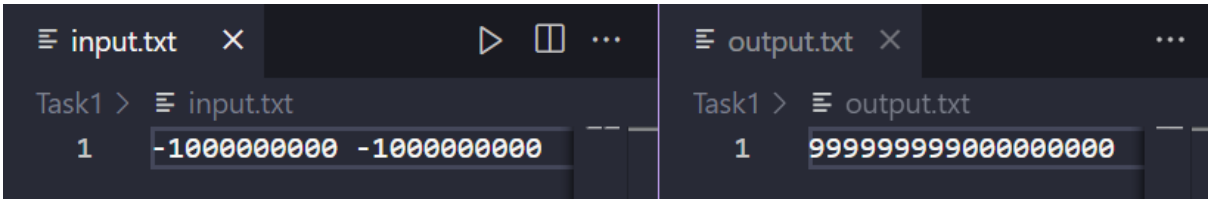
- Результат работы кода:





- Результат работы кода на максимальных и минимальных значениях:





Тест	Время выполнения (s)	Затраты памяти(bytes)
5 27	0.0004285000031813979	84
23 8	0.00031889998354017735	84
-10^9 -10^9	0.0045185000053606 93	88
10^9 10^9	0.0002894999925047159	88

TASK 2: Число Фибоначчи

Определение последовательности Фибоначчи:
F0 = 0
F1 = 1
Fi = Fi−1 + Fi−2 для i ≥ 2.
Ваша цель – разработать эффективный алгоритм для подсчета чисел Фибоначчи.
• Имя входного файла: input.txt
• Имя выходного файла: output.txt
• Формат входного файла. Целое число n. 0 ≤ n ≤ 45.
• Формат выходного файла. Число Fn.

- Листинг кода

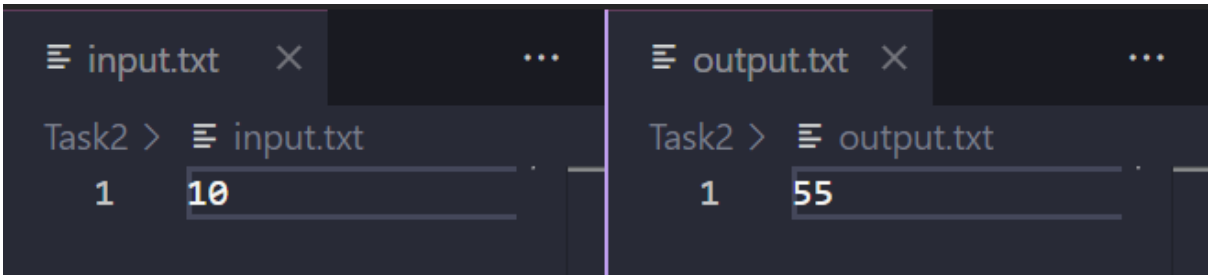
```
n = 0
a1 = open('D:\\Python\\CTDL&GT\\Lab0\\Task2\\input.txt',
'r')
a2 = open('D:\\Python\\CTDL&GT\\Lab0\\Task2\\output.txt',
'w')
```

```
n = int(a1.readline())
f0, f1 = 0, 1
if n<0 or n>45:
    print("Неправильный формат данных.")
elif n==0:
    f1=f0=0
else:
    for _ in range (2, n+1):
        f0, f1 = f1, f0+f1
a2.write(str(f1))
```

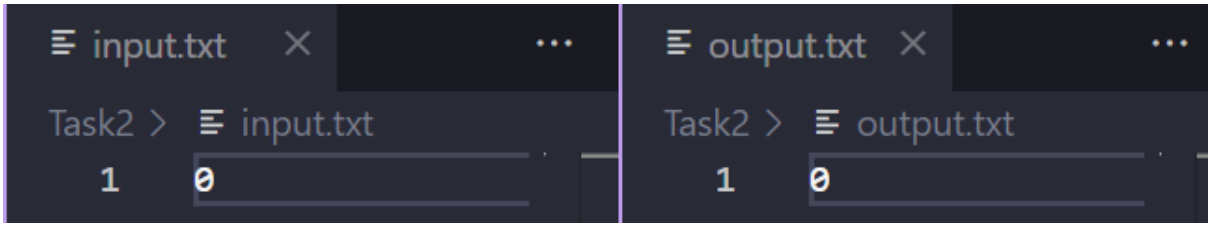
- Текстовое объяснение решения.

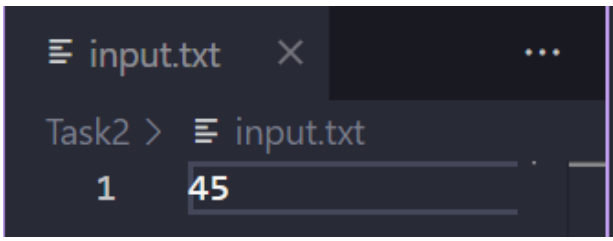
Открываю файлы для чтения (*input.txt*) и записи (*output.txt*) с помощью функции *open()*. Функция *map()* применяет функцию *int()* к каждому элементу списка, возвращаемого *split()*. Проверяю полученные числа с помощью двойного неравенства. Значения переменных *f0, f1* записываются после каждого цикла. Когда цикл завершается, значение числа Фибоначчи записывается в файл *output.txt*.

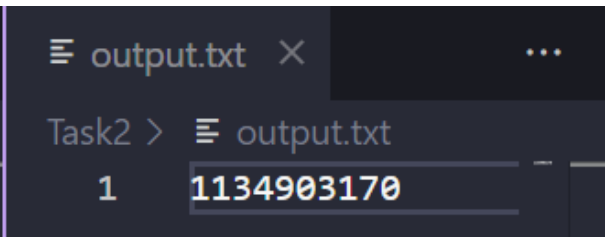
- Результат работы кода на примерах из текста задачи:



- Результат работы кода на максимальных и минимальных значениях:







Тест	Время выполнения (s)	Затраты памяти(bytes)
10	0.00078980000185377896	84
0	0.00029539999792225659	84
45	0.00031679996754974127	88

TASK 3: Еще про числа Фибоначчи

Определение последней цифры большого числа Фибоначчи. Числа Фибоначчи растут экспоненциально. Например,

F200 = 280571172992510140037611932413038677189525

Хранить такие суммы в массиве, и при этом подсчитывать сумму, будет достаточно долго. Найти последнюю цифру любого числа достаточно просто: F mod 10.

• Имя входного файла: input.txt

• Имя выходного файла: output.txt

• Формат входного файла. Целое число n. 0 ≤ n ≤ 107

• Формат выходного файла. Одна последняя цифра числа Fn.

- Листинг кода

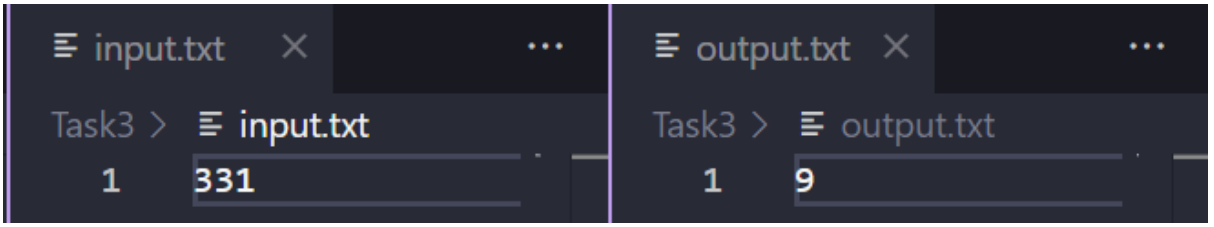
```
n = 0
a1  =  open('D:\\Python\\CTDL&GT\\Lab0\\Task3\\input.txt',
'r')
a2  =  open('D:\\Python\\CTDL&GT\\Lab0\\Task3\\output.txt',
'w')
n = int(a1.read())
if n<0 or n>10**7:
    print("Неправильный формат данных.")
elif n==0:
    f1=f0=0
else:
    f0, f1 = 0, 1
    for _ in range(2, n+1):
        f0, f1 = f1, (f0 + f1) % 10
a2.write(str(f1 % 10))
```

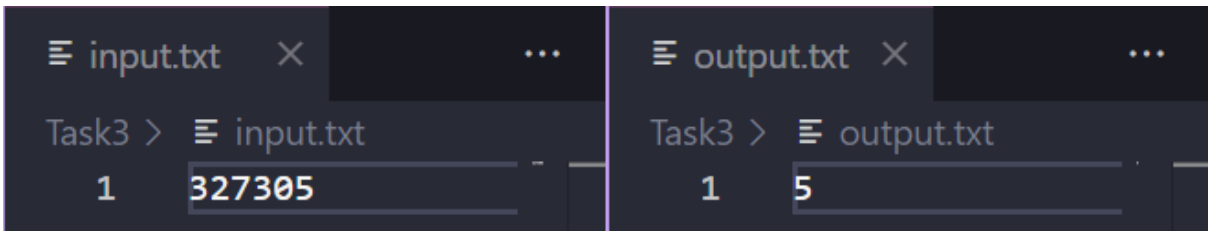
- Текстовое объяснение решения

Открываю файлы для чтения (*input.txt*) и записи (*output.txt*) с помощью функции *open()*. Функция *map()* применяет функцию *int()* к каждому элементу списка, возвращаемого *split()*. Проверяю полученные числа с помощью двойного неравенства. Значения переменных *f0, f1* записываются

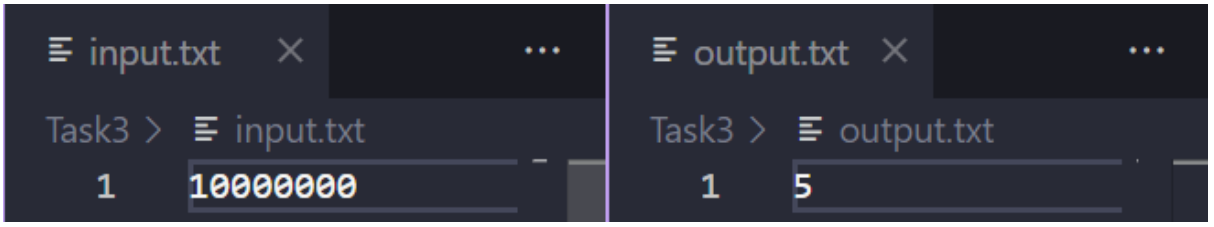
после каждого цикла. Когда цикл завершается, я использую остаточное деление «%», чтобы найти последнюю цифру. Значение записывается в файл *output.txt*.

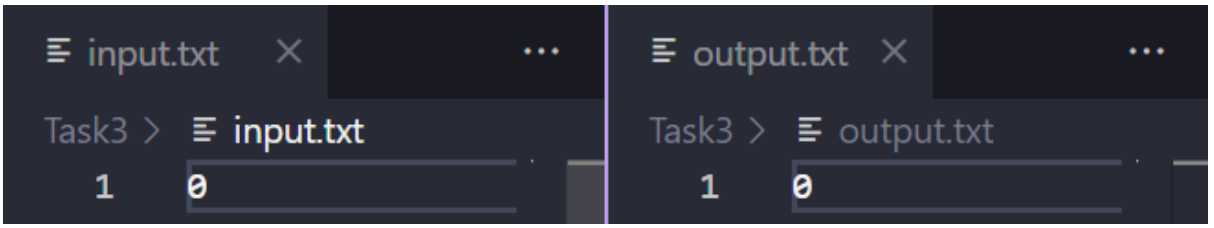
- Результат работы кода:





- Результат работы кода на максимальных и минимальных значениях:





Тест	Время выполнения (s)	Затраты памяти(bytes)
331	0.00036289996933192015	84
327305	0.030454499999626726	84
0	0.00028529996052384377	84
10000000	0.8877892000018619	84

TASK 4: Тестирование ваших алгоритмов.

Задача: Вам необходимо протестировать время выполнения вашего алгоритма в task 2 и task 3.

Дополнительно: вы можете протестировать объем используемой памяти при выполнении вашего алгоритма.

- Листинг кода

```
import time
t_start = time.perf_counter()
n = 0
a1 = open('D:\\Python\\CTDL&GT\\Lab0\\Task4\\input.txt', 'r')
a2 = open('D:\\Python\\CTDL&GT\\Lab0\\Task4\\output.txt', 'w')
n = int(a1.read())
f0, f1 = 0, 1
if n<0 or n>45:
    print("Неправильный формат данных.")
elif n==0:
    f1=f0=0
else:
    for _ in range (2, n+1):
        f0, f1 = f1, f0+f1
t_stop = time.perf_counter()
print('время работы', t_stop-t_start)
a2.write(str(f1))
```

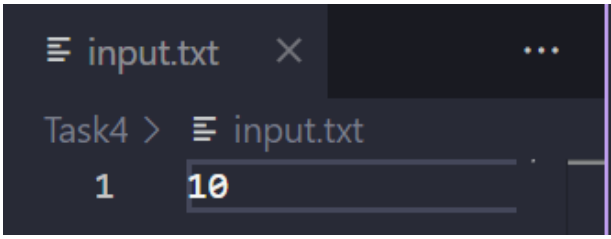
```
import time
import sys
t_start = time.perf_counter()
n = 0
a1 = open('D:\\Python\\CTDL&GT\\Lab0\\Task4\\input.txt', 'r')
a2 = open('D:\\Python\\CTDL&GT\\Lab0\\Task4\\output.txt', 'w')
n = int(a1.read())
if n<0 or n>10**7:
    print("Неправильный формат данных.")
elif n==0:
    f1=f0=0
else:
    f0, f1 = 0, 1
```

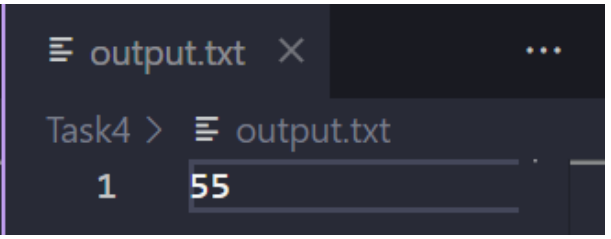
```
    for _ in range(2, n+1):
        f0, f1 = f1, (f0 + f1) % 10
a2.write(str(f1 % 10))
t_stop = time.perf_counter()
print('время работы', t_stop-t_start)
```

- Текстовое объяснение решения.

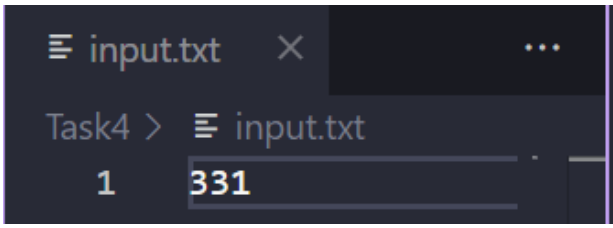
Подключаем библиотеки *time, sys*. Вычисляю соответственно затраченное время и память каждого теста с помощью методов *time.perf_ counter()* и *sys.getsizeof()*

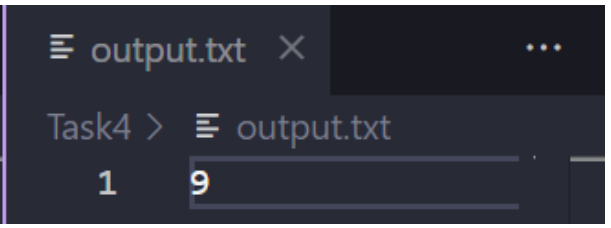
- Результат работы кода на тестах:






```
время работы 0.00032180000056222081
memory usage: 84 bytes
```






```
время работы 0.0003763999557122588
memory usage: 84 bytes
```

Вывод :

1. Выполняйте сложения и квадраты в пределах
2. −10^9 ≤ n 10^9 быстро
3. Чтение и запись данных в файл экономит время по сравнению с обычным вводом данных.
4. Сложность операции O(1)
5. Затраты времени и памяти алгоритма можно рассчитать с помощью методов time.perf_counter() и sys.getsizeof().