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

Выполнил:
Левахин Лев Александрович
К3140

Проверил:
--

Санкт-Петербург
2024г

Содержание отчета

Задачи по варианту

Задача №1. Ввод-Вывод [N баллов]

Вам необходимо выполнить 4 следующих задачи:

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("Введите число a и через пробел число b: ").split())
if (-10**9<=a<=10**9) and (-10**9<=b<=10**9):
    print(a + b)
else:
    print("Числа должны удовлетворять условию |a|<=10**9, |b|<=10**")
```

Текстовое объяснение решения:
Считываем переменные a и b, сразу приводим их к инту.
Проверяем удовлетворяют ли числа условию задачи: если да – выводим сумму чисел, если нет – выводим ошибку.

Результат работы кода на примерах из текста задачи и результат работы кода в случае превышения хотя бы одним из чисел допустимого значения:

```
Введите число a и через пробел число b: 10000000001 1
Числа должны удовлетворять условию |a|<=10**9, |b|<=10**9
PS C:\Python> & c:/Users/name8/AppData/Local/Programs/Pyth
Введите число a и через пробел число b: 10 20
30
```

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("Введите число a и через пробел число b: ").split())
if (-10**9<=a<=10**9) and (-10**9<=b<=10**9):
    print(a + b**2)
else:
```

```
    print("Числа должны удовлетворять условию |a|<=109, |b|<=109")
```

Текстовое объяснение решения:
Считываем переменные a и b, сразу приводим их к инту.
Проверяем удовлетворяют ли числа условию задачи: если да – выводим результат по условию задачи, если нет – выводим ошибку.

Результат работы кода на примерах из текста задачи и результат работы кода в случае превышения хотя бы одним из чисел допустимого значения:

```
Введите число a и через пробел число b: 100000000001 1
Числа должны удовлетворять условию |a|<=109, |b|<=109
PS C:\Python> & c:/Users/name8/AppData/Local/Programs/
Введите число a и через пробел число b: 10 10
110
```

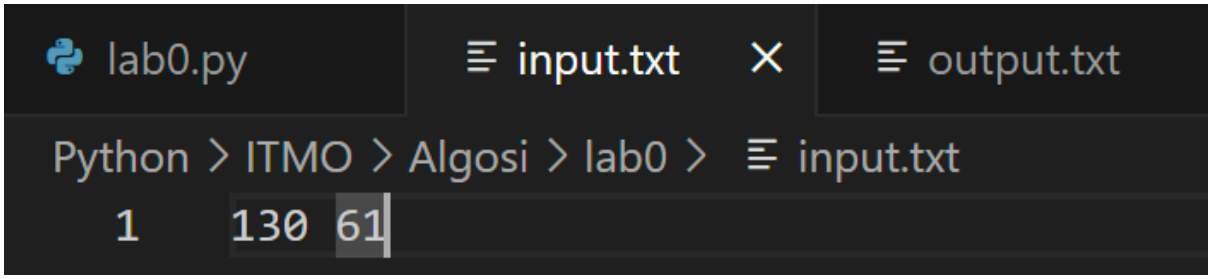
3. Выполните задачу a + b с использованием файлов.
• Имя входного файла: input.txt
• Имя выходного файла: output.txt
• Формат входного файла. Входной файл состоит из одной строки, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-10^{**}9 \le a, b \le 10^{**}9$.
• Формат выходного файла. Выходной файл единственное целое число — результат сложения a + b.

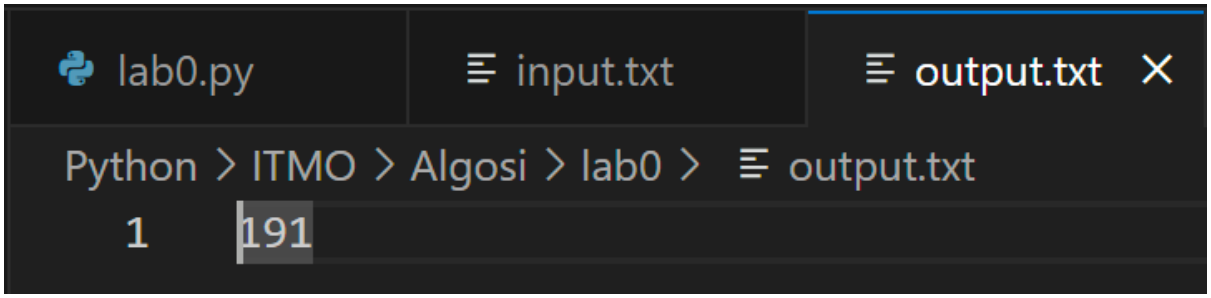
Листинг кода

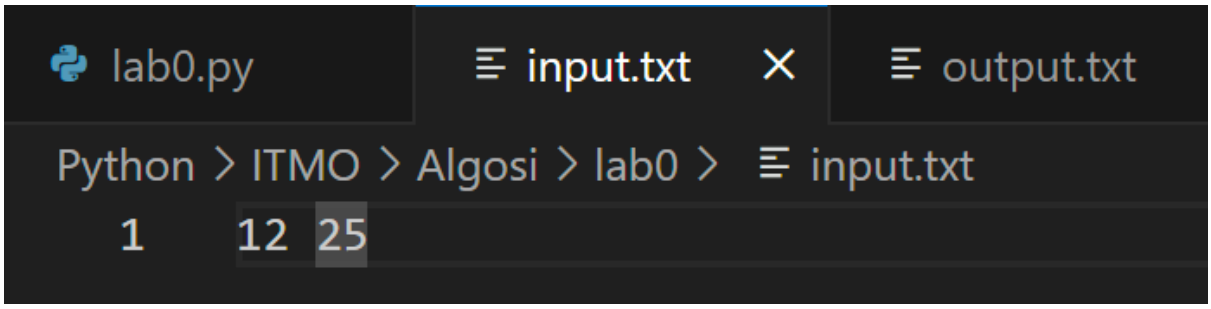
```
input = open(r"C:\Python\ITMO\Algosi\lab0\input.txt")
output = open(r"C:\Python\ITMO\Algosi\lab0\output.txt", "w")

a, b = map(int, input.readline().split())
if (-10**9<=a<=10**9) and (-10**9<=b<=10**9):
    output.write(str(a+b))
    output.close()
else:
    print("Числа должны удовлетворять условию |a|<=109, |b|<=109")
```

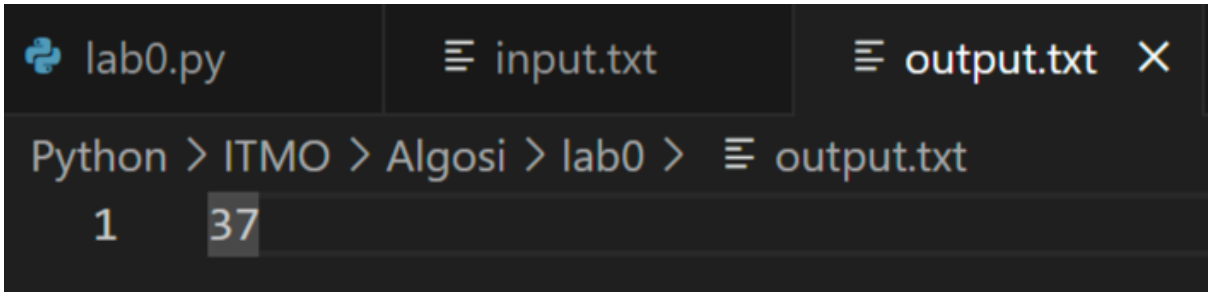
Текстовое объяснение решения:
Открываем файлы: файл инпут для чтения, файл оутпут для записи. Считываем переменные a и b, сразу приводим их к инту. Проверяем удовлетворяют ли числа условию задачи: если да – записываем результат в файл оутпут, закрываем файл. Если нет – выводим ошибку.

Результат работы кода на примерах из текста задачи:

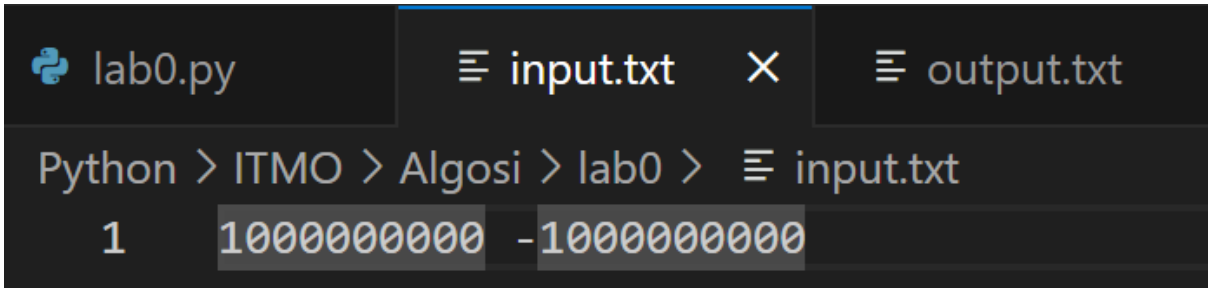


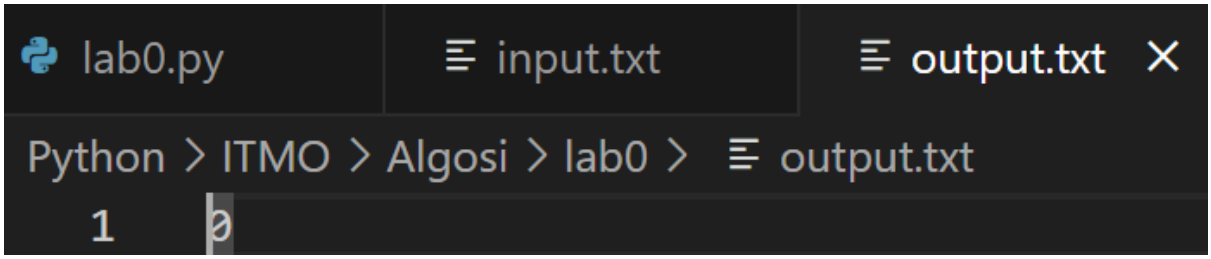






Результат работы кода на максимальных и минимальных значениях:





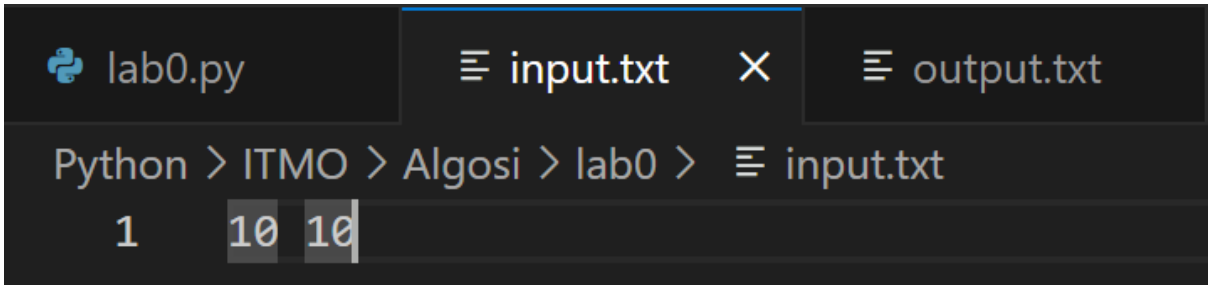
4. Выполните задачу a+b**2 с использованием файлов аналогично предыдущему пункту.

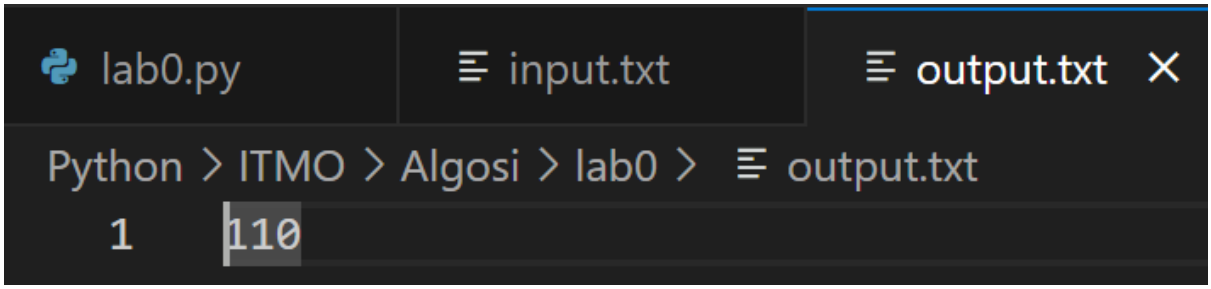
Листинг кода:

```
input = open(r"C:\Python\ITMO\Algosi\lab0\input.txt")
output = open(r"C:\Python\ITMO\Algosi\lab0\output.txt", "w")

a, b = map(int, input.readline().split())
if (-10**9<=a<=10**9) and (-10**9<=b<=10**9):
    output.write(str(a+b**2))
    output.close()
else:
    print("Числа должны удовлетворять условию |a|<=109, |b|<=109")
```

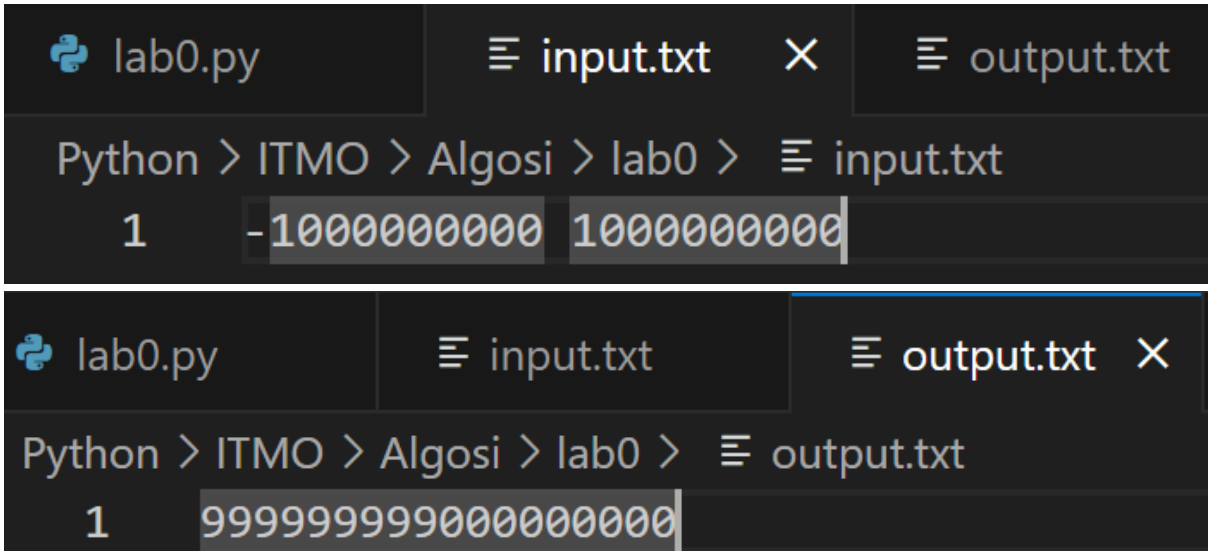
Текстовое объяснение решения:
Открываем файлы: файл инпут для чтения, файл оутпут для записи. Считываем переменные a и b, сразу приводим их к инту. Проверяем удовлетворяют ли числа условию задачи: если да – записываем результат в файл оутпут, закрываем файл. Если нет – выводим ошибку.

Результат работы кода на примере:





Результат работы кода на максимальных и минимальных значениях:



Вывод по задаче 1:

1) Чем меньше числа на вход, тем быстрее выполняется алгоритм
2) Работать с файлами не всегда удобнее, чем в консоли

Задача №2. Число Фибоначчи [N баллов]

Определение последовательности Фибоначчи:

F0 = 0 (1)

F1 = 1

Fi = Fi−1 + Fi−2 для i ≥ 2.

Таким образом, каждое число Фибоначчи представляет собой сумму двух предыдущих, что дает последовательность 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... 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.
• Пример.

input.txt 10

output.txt 55

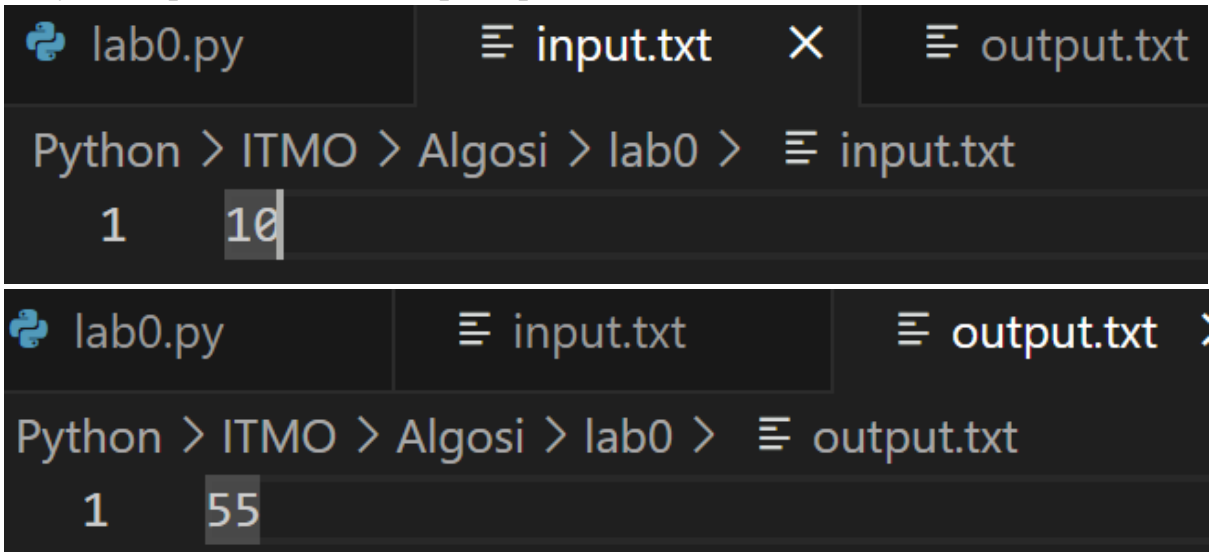
Листинг кода

```
input = open(r"C:\Python\ITMO\Algosi\lab0\input.txt")
output = open(r"C:\Python\ITMO\Algosi\lab0\output.txt", "w")
```

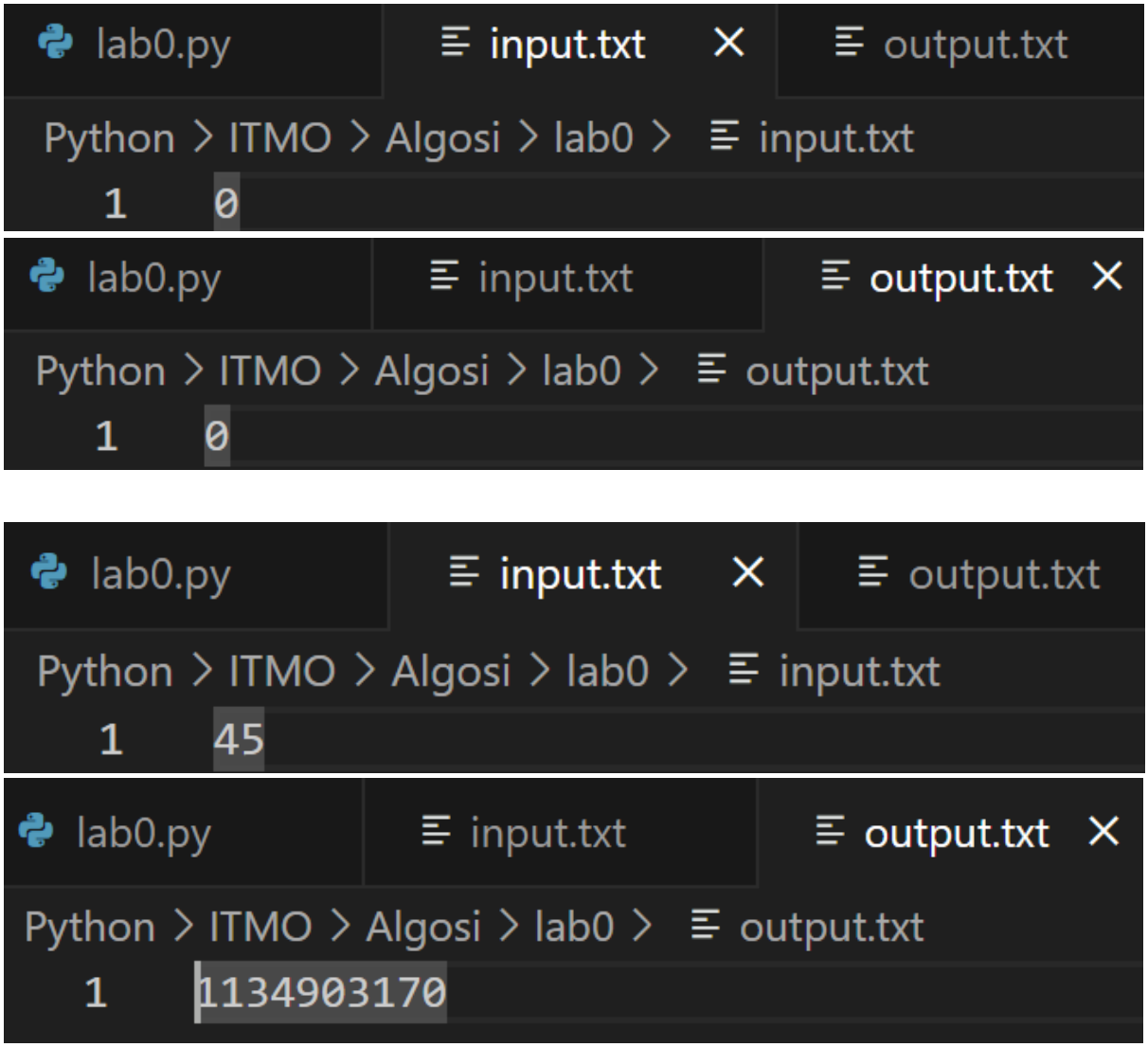
```python
import time
t_start = time.perf_counter()

n = int(input.readline())
if (0<=n<=45):
    fib = [0,1]
    for i in range(2,46):
        fib.append(fib[i-2]+fib[i-1])

    output.write(str(fib[n]))
    output.close()
else:
    print("Введённый элемент не должен превышать 45")

print("Время работы: %s секунд" % (time.perf_counter() - t_start))
```

Текстовое объяснение решения:
Открываем файл инпут(для чтения) и файл оутпут(для записи).
Импортируем библиотеку тайм для того, чтобы сделать таймер (задача 4)
Вводим переменную таймер. Обозначаем переменную n, в которой будем принимать значение из консоли. Проверяем условие, чтобы н было от 0 до 45. Если условие выполняется: задаём список чисел фибоначчи, в который добавляем два первых элемента 0 и 1, перебираем элементы со второго до 45 включительно. Добавляем в список чисел фибоначчи каждый элемент, по рассчитываемый по формуле ряда чисел фибоначчи. Далее в файл оутпут записываем результат из списка, по номеру элемента, который ввёл пользователь и закрываем файл. Если условие не выполняется(н не в рамках условия задачи) то – выводим ошибку. В конце выводим в консоль время выполнения алгоритма(для 4 задачи).

Результат работы кода на примерах из текста задачи:



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения
Нижняя граница диапазона значений входных данных из текста задачи	0.00012429998605512083 секунд
Пример из задачи	0.00014250000822357833 секунд
Пример из задачи	0.00014429999282583594 секунд
Верхняя граница диапазона значений входных данных из текста задачи	0.0002690000110305846 секунд

Вывод по задаче:

1) Чем больше число фибоначчи (н подаваемое на вход), тем дольше работает алгоритм, так как скорость O(n).

Задача №3. Больше про числа Фибоначчи [N баллов]

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.

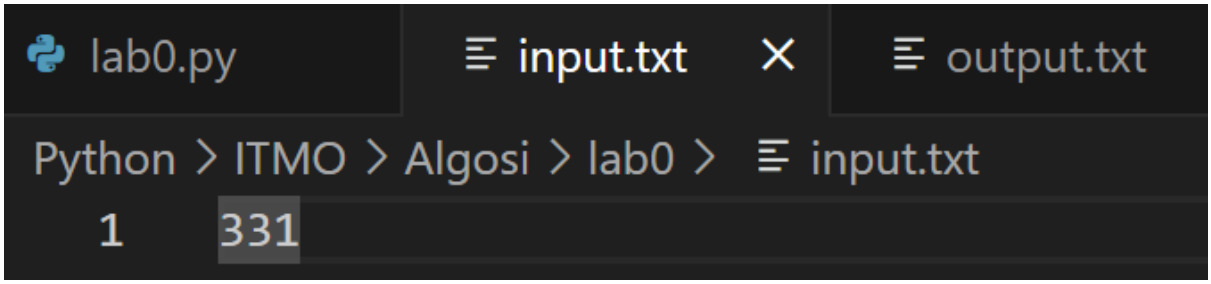
• Имя входного файла: input.txt
• Имя выходного файла: output.txt
• Формат входного файла. Целое число n. 0 ≤ n ≤ 10**7 .
• Формат выходного файла. Одна последняя цифра числа Fn.
•Пример 1.
input.txt 331
output.txt9
F331=668996615388005031531000081241745415306766517246774551964595292186469.
• Пример 2.
input.txt 327305
output.txt 5
Это число не влезет в страницу, но оканчивается действительно на 5.
• Ограничение по времени: 5сек.
• Ограничение по памяти: 512 мб.

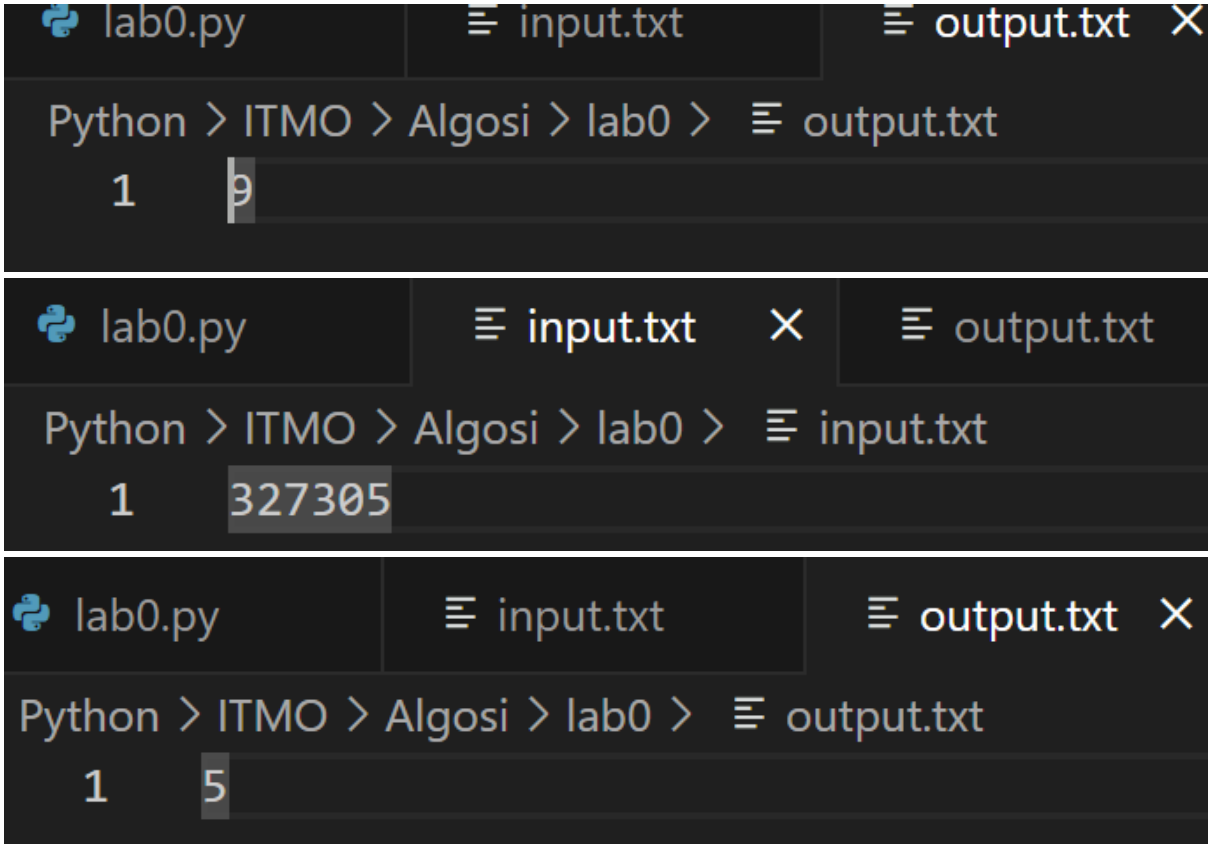
```
input = open(r"C:\Python\ITMO\Algosi\lab0\input.txt")
output = open(r"C:\Python\ITMO\Algosi\lab0\output.txt", "w")

t_start = time.perf_counter()

n = int(input.readline())

def fib(n):
    if n<=1:
        return n
    f1, f2 = 0,1
    for i in range(2,n+1):
        f1, f2 = (f2%10), (f1+f2)%10
    return f2
```

```
if (0<=n<=10**7):
    output.write(str(fib(n)%10))
    output.close()
else:
    print("Время работы: %s секунд" % (time.perf_counter() - t_start))

print("Время работы: %s секунд" % (time.perf_counter() - t_start))
```

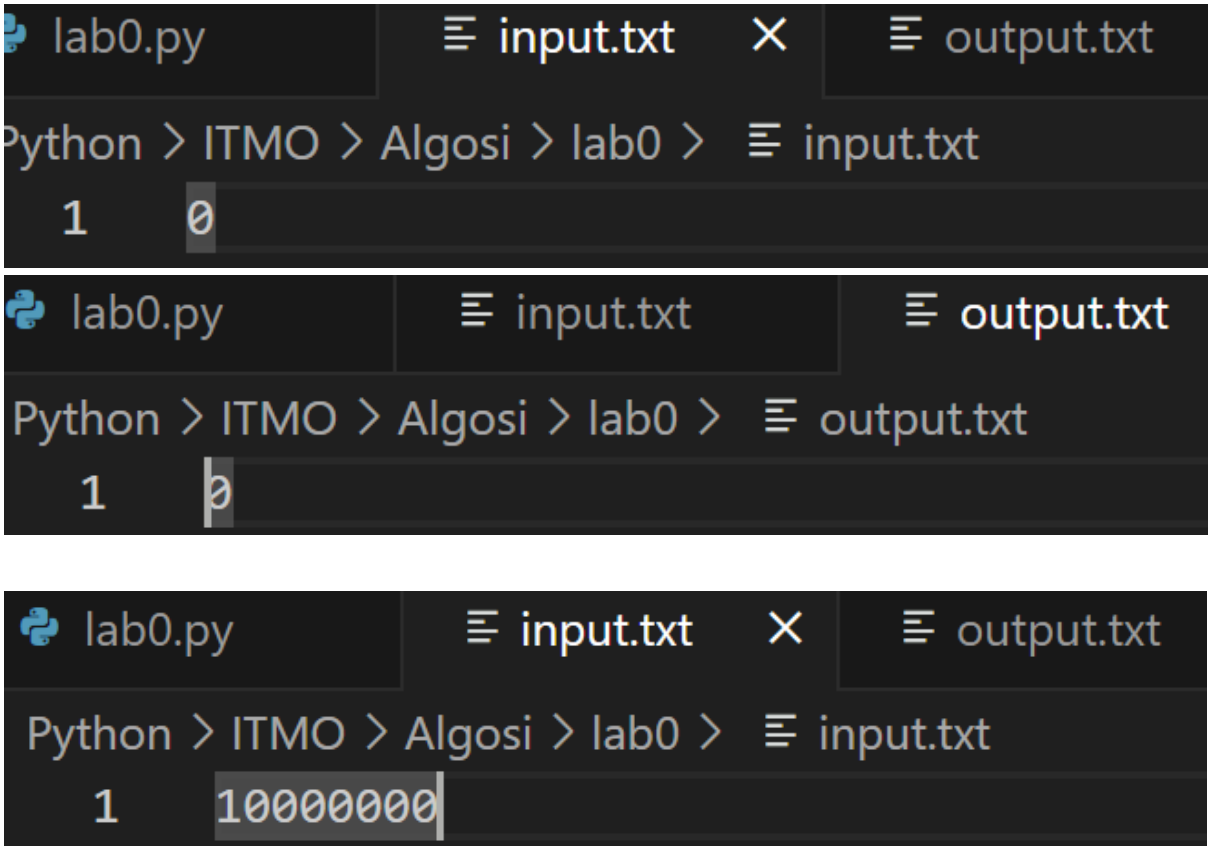
Текстовое объяснение решения.
Открываем файл инпут(для чтения) и файл оутпут(для записи).
Импортируем библиотеку тайм для того, чтобы сделать таймер (Я сделал это в прошлой задаче) (задача 4)
Вводим переменную таймер. Обозначаем переменную n, в которой будем принимать значение из консоли. Создаём функцию fib, которая принимает на вход переменную н – номер искомого элемента последовательности Фибоначчи. Если переменная н меньше или равна одному(соответствует значениям 0 или 1), то возвращаем н(аналогично 0 или 1, так как это и есть первые два элемента последовательности). Задаём первые два элемента последовательности, чтобы в дальнейшем можно было производить расчёты отталкиваясь от них. Начинаем перебор значений(от 2 до н(включительно)). Чтобы подсчитать элемент последовательности по номеру пользователя, переприсваеваем значения для переменных ф1 и ф2 согласно формуле чисел фибоначчи. Чтобы оптимизировать алгоритм, уже на этом этапе будем брать только последнюю цифру обоих чисел. Так, доходим до элемента последовательности, номер которого задаёт пользователь, и возвращаем его.
Проверяем условие, чтобы н было от 0 до 10**7. Если условие выполняется: в файл оутпут записываем результат выполнения функции от н и закрываем файл. Если условие не выполняется(н не в рамках условия задачи) то – выводим ошибку. В конце выводим в консоль время выполнения алгоритма(для 4 задачи).

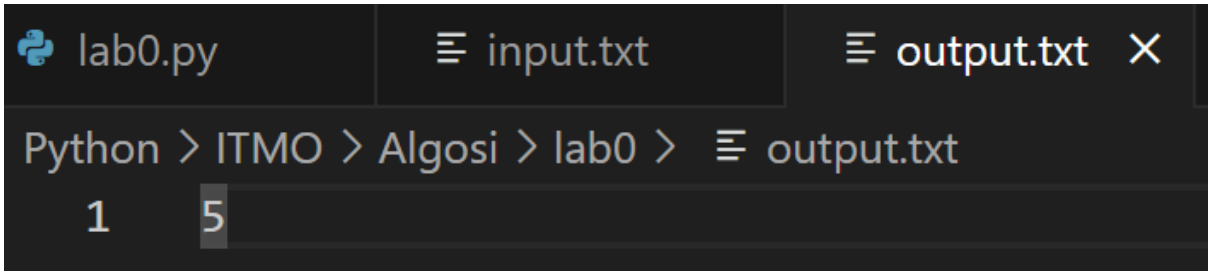
Результат работы кода на примерах из текста задачи:





Результат работы кода на максимальных и минимальных значениях:





	Время выполнения
Нижняя граница диапазона значений входных данных из текста задачи	0.00013669999316334724 секунд
Пример из задачи	0.0001306000049225986 секунд
Пример из задачи	0.014792900008615106 секунд
Верхняя граница диапазона значений входных данных из текста задачи	0.4272225000022445 секунд

Вывод по задаче:

1) Чтобы алгоритм выполнялся за нормальное время(особенно при использовании больших чисел), нужно использовать максимально оптимизированный алгоритм.

Задача №4. Тестирование ваших алгоритмов[N баллов]

Задача: вам необходимо протестировать время выполнения вашего алгоритма в Задании 2 и Задании 3.

Время выполнения алгоритмов и код прописано выше

Вывод

1) Выполнение простейших арифметических операций занимает немного ресурсов и времени.
2) Можно оптимизировать даже оптимизированный алгоритм.