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:
Федоров М.О.
К3140

Проверила:
Артамонова В.Е.

Санкт-Петербург
2023 г.

Содержание отчета

Задачи по варианту

Задача №1. Ввод-вывод

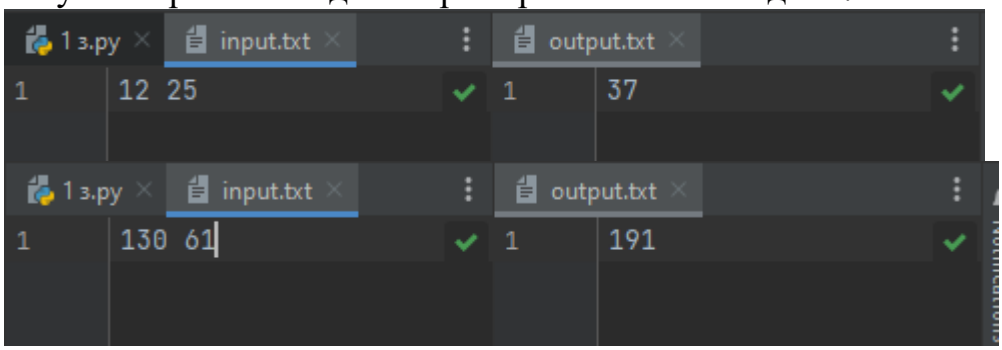
3. Выполните задачу a + b с использованием файлов.
• Имя входного файла: input.txt
• Имя выходного файла: output.txt

• 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.

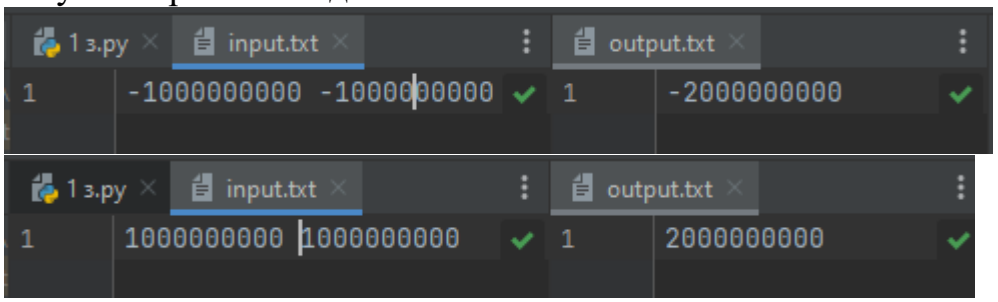
```
with open('input.txt', 'r') as file_input:
    a, b = map(int, file_input.read().split())
with open('output.txt', 'w') as file_output:
    file_output.write(str(a + b))
```

Текстовое объяснение решения.
Считываем из файла input.txt два целых числа, после чего выводим в файл output.txt сумму этих чисел.

Результат работы кода на примерах из текста задачи:



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0033437 с	14.0 Мб
Пример из задачи	0.002373 с	13.94140625 Мб
Пример из задачи	0.0024664 с	13.88671875 Мб
Верхняя граница диапазона значений входных данных из текста задачи	0.0022975 с	14.01953125 Мб

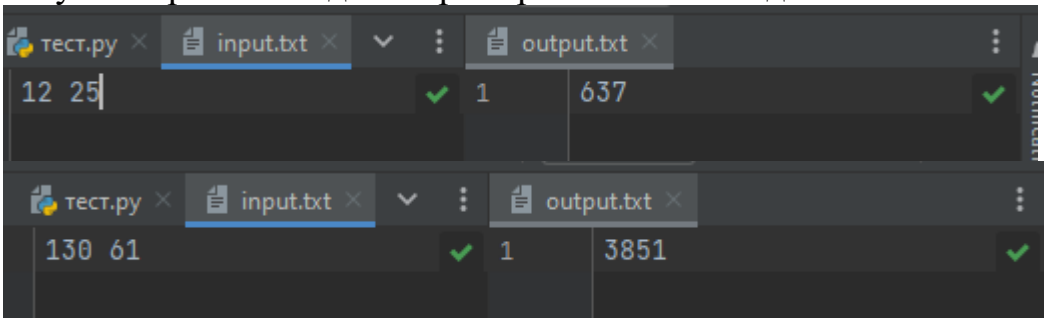
Задача №2. Число Фибоначчи

В данной задаче требуется вычислить значение $a + b^2$ 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 — $a + b^2$.

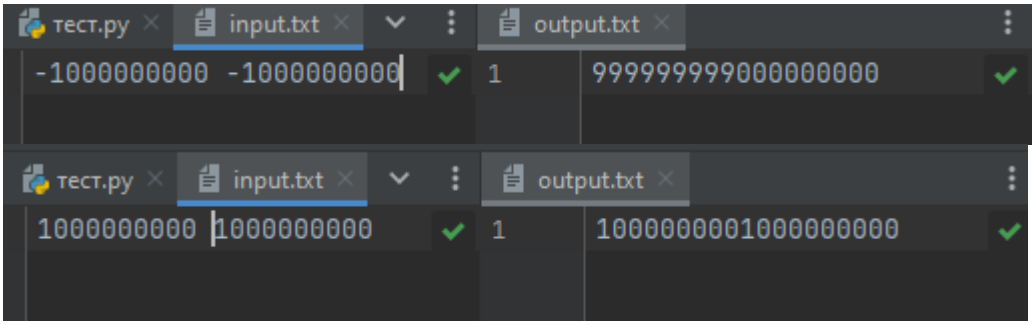
Листинг кода:

```
with open('input.txt', 'r') as file_input:
    a, b = map(int, file_input.read().split())
with open('output.txt', 'w') as file_output:
    file_output.write(str(a + b**2))
```

Результат работы кода на примерах из текста задачи:



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0029375 с	14.05078125 Мб
Пример из задачи	0.0572226 с	13.9921875 Мб
Пример из задачи	0.0028697 с	13.98828125 Мб
Верхняя граница диапазона значений входных данных из текста задачи	0.0031817 с	13.9765625 Мб

Текстовое объяснение решения.
Считываем из файла input.txt два целых числа, после чего выводим в файл output.txt $a + b^2$.

Вывод по задаче: Вспомнил как работать с файлами и находить время выполнения кода, узнал как находить затраты памяти и устанавливать модули в PyCharm.

Задача №2. Число Фибоначчи

Определение последовательности Фибоначчи:

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_i &= F_{i-1} + F_{i-2} \text{ для } i \geq 2. \end{aligned} \tag{1}$$

Таким образом, каждое число Фибоначчи представляет собой сумму двух предыдущих, что дает последовательность

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Ваша цель – разработать эффективный алгоритм для подсчета чисел Фибоначчи. Вам предлагается начальный код на Python, который содержит наивный рекурсивный алгоритм:

```
def calc_fib(n):
        if (n <= 1):
                return n

        return calc_fib(n - 1) + calc_fib(n - 2)

n = int(input())
print(calc_fib(n))
```

- Имя входного файла: input.txt
- Имя выходного файла: output.txt
- Формат входного файла. Целое число n. $0 \leq n \leq 45$.
- Формат выходного файла. Число F_n.
- Пример.

input.txt	10
output.txt	55

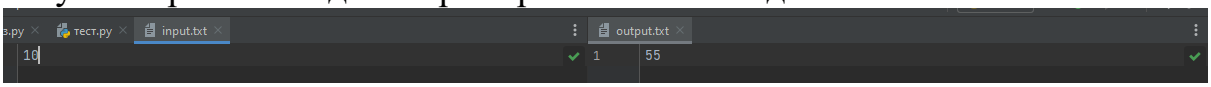
Листинг кода.

```
with open('input.txt', 'r') as file_input:
    n = int(file_input.read())
fib = [0, 1]
for i in range(2, n + 1):
    fib.append(fib[-1] + fib[-2])
with open('output.txt', 'w') as file_output:
    file_output.write(str(fib[n]))
```

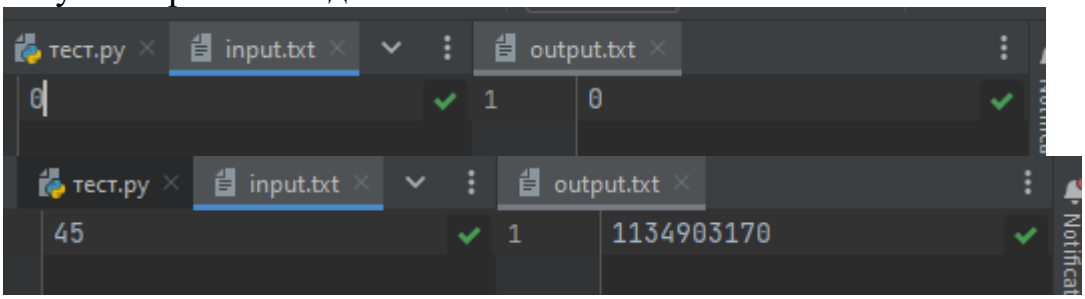
Текстовое объяснение.

Считываем из файла input.txt номер числа Фибоначчи. Создадим массив, добавляем новые элементы, которые являются суммой двух последних элементов. Выводим в файл output.txt число Фибоначчи.

Результат работы кода на примерах из текста задачи:



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0027931	14.0234375
Пример из задачи	0.0027247	13.98828125
Верхняя граница диапазона значений входных данных из текста задачи	0.0021373	14.0078125

Вывод по задаче: вспомнил как работать с массивами и циклами.

Задача №3. Еще про числа Фибоначчи

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

$$F_{200} = 280571172992510140037611932413038677189525$$

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. $0 \leq n \leq 10^7$.
- Формат выходного файла. Одна последняя цифра числа F_n.
- Пример 1.

input.txt	331
output.txt	9

$F_{331} = 668996615388005031531000081241745415306766517246774551964595292186469.$

- 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.

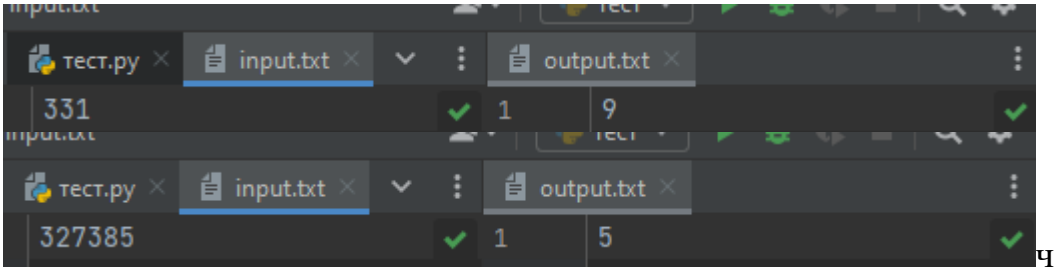
Листинг кода:

```
ith open('input.txt', 'r') as file_input:
    n = int(file_input.read())
fib = [0, 1]
for i in range(2, n + 1):
    fib.append((fib[-1] + fib[-2])%10)
with open('output.txt', 'w') as file_output:
    file_output.write(str(fib[n]))
```

Текстовое объяснение.

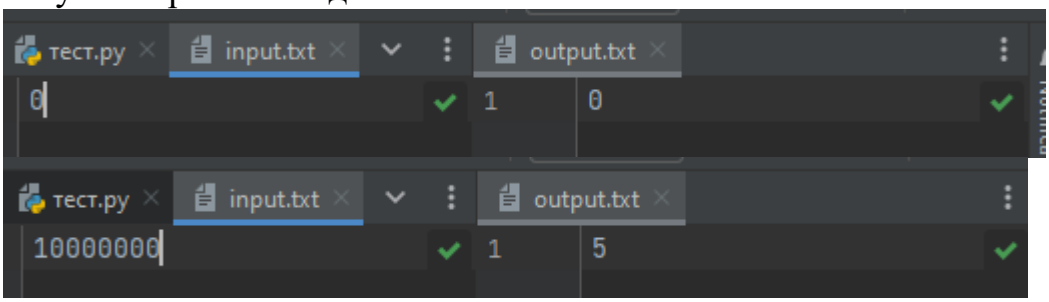
Считываем из файла input.txt номер числа Фибоначчи. Создадим массив, добавляем новые элементы, которые являются остатком от деления на 10 суммы двух последних элементов. Выводим в файл output.txt последнее число Фибоначчи.

Результат работы кода на примерах из текста задачи:



ч

Результат работы кода на максимальных и минимальных значениях:



	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0026703	14.023437
Пример из задачи	0.0025431	13.98828125
Пример из задачи	0.1714433	18.30078125
Верхняя граница диапазона значений входных данных из текста задачи	6.18369	92.609375

Вывод по задаче: Я применил знание эффективных решений.

Задача №4. Тестирование ваших алгоритмов.

Задание 4. Тестирование ваших алгоритмов.

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

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

Листинг кода.

```
from time import perf_counter_ns
from os import getpid
from psutil import Process



start_time = perf_counter_ns()

with open('input.txt', 'r') as file_input:
    n = int(file_input.read())
fib = [0, 1]
for i in range(2, n + 1):
    fib.append((fib[-1] + fib[-2])%10)
with open('output.txt', 'w') as file_output:
    file_output.write(str(fib[n]))

print('Время выполнения:', (perf_counter_ns() -
start_time) / 10 ** 9, 'c')
print('Затраты памяти:',
Process(getpid()).memory_info().rss / 1024 ** 2,
'Мб')
```

Текстовое объяснение решения:

Импортируем библиотеки os, psutil, time. замеряем время разностью времен начала и конца выполнения алгоритма. Функциями getpid и memory_info, которые мы импортировали рассчитываем память. Выводим необходимую информацию.

Вывод

Я повторил способы измерения времени выполнения и затрат памяти.