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

Вариант 1

Выполнил:

Лютый Н.А.

К3140

Проверил:

Афанасьев А.В.

Санкт-Петербург

2024 г.

# Содержание отчета

[Содержание отчета 2](#_Toc177344361)

[Задачи по варианту 3](#_Toc177344362)

[Задача №1. Ввод-вывод 3](#_Toc177344363)

[Задача №2. Число Фибоначчи 9](#_Toc177344364)

[Задача №3. Еще про числа Фибоначчи 12](#_Toc177344365)

[Задача №4. Тестирование ваших алгоритмов 15](#_Toc177344366)

[Вывод 17](#_Toc177344367)

# Задачи по варианту

## Задача №1. Ввод-вывод

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 −109 ≤ a, b ≤ 109 . Выход: единственное целое число — результат сложения a + b.
2. Задача a + b2. В данной задаче требуется вычислить значение a + b2. Вход: одна строка, которая содержит два целых числа a и b. Для этих чисел выполняются условия −109 ≤ a, b ≤ 109. Выход: единственное целое число — результат сложения a + b2.
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 −109≤ a, b ≤109.
   * Формат выходного файла. Выходной файл единственное целое число — результат сложения a + b.
4. Выполните задачу a+b2 с использованием файлов аналогично предыдущему пункту.

Задача 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("Числа должны быть от -10\*\*9 до 10\*\*9!")

a,b = map(int, input().split())

if -10\*\*9<=a<=10\*\*9 and -10\*\*9<=b<=10\*\*9:

    print(a+b)

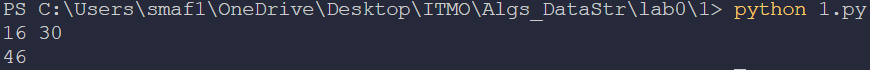
else:

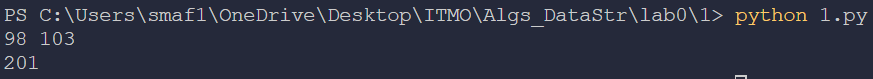
    print("Числа должны быть от -10\*\*9 до 10\*\*9!")

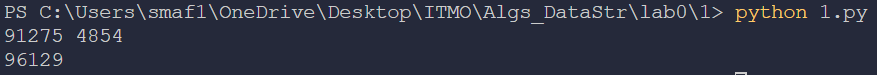
Текстовое объяснение решения:

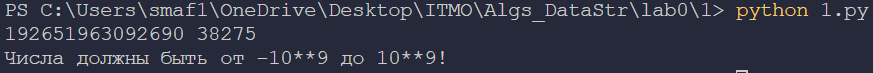
Принимаем строку, которую разделяем по пробелу и каждый элемент приводим к числовому типу. Запоминаем 1-ый в переменную a, а 2-ой в переменную b. Проверяем, удовлетворяют ли числа условию задачи (быть в диапазоне от -10\*\*9 до 10\*\*9). Если удовлетворяют, то выводим их сумму, если нет – выводим ошибку.

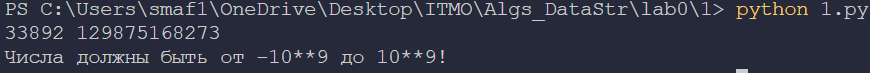
Результат работы кода, учитывая случаи несоответствия изначальных чисел условию:











Задача a + b2

Листинг кода:

a,b = map(int, input().split())

if -10\*\*9<=a<=10\*\*9 and -10\*\*9<=b<=10\*\*9:

    print(a+b\*\*2)

else:

    print("Числа должны быть от -10\*\*9 до 10\*\*9!")

a,b = map(int, input().split())

if -10\*\*9<=a<=10\*\*9 and -10\*\*9<=b<=10\*\*9:

    print(a+b\*\*2)

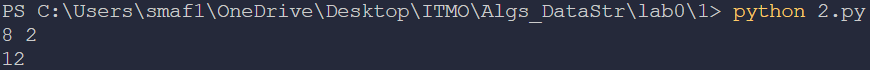
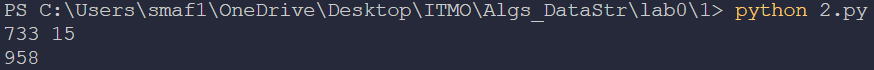
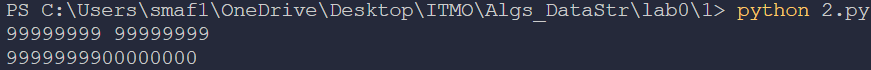
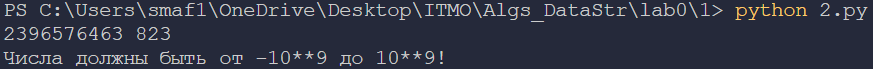
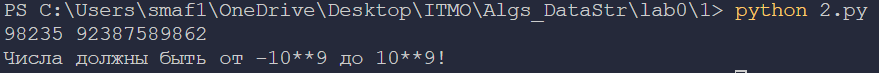
else:

    print("Числа должны быть от -10\*\*9 до 10\*\*9!")

Текстовое объяснение решения:

Принимаем строку, которую разделяем по пробелу и каждый элемент приводим к числовому типу. Запоминаем 1-ый в переменную a, а 2-ой в переменную b. Проверяем, удовлетворяют ли числа условию задачи (быть в диапазоне от -10\*\*9 до 10\*\*9). Если удовлетворяют, то выводим их сумму первого элемента и 2-го, возведенного в квадрат, если нет – выводим ошибку.

Результат работы кода, учитывая случаи несоответствия изначальных чисел условию:

Задача a+b с использованием файлов

Листинг кода:

inp\_file = open('input.txt')

a,b = map(int, inp\_file.readline().split())

inp\_file.close()

if -10\*\*9<=a<=10\*\*9 and -10\*\*9<=b<=10\*\*9:

    out = str(a+b)

    out\_file = open('output.txt', 'w')

    out\_file.write(out)

    out\_file.close()

else:

    print("Числа должны быть от -10\*\*9 до 10\*\*9!")

inp\_file = open('input.txt')

a,b = map(int, inp\_file.readline().split())

inp\_file.close()

if -10\*\*9<=a<=10\*\*9 and -10\*\*9<=b<=10\*\*9:

    out = str(a+b)

    out\_file = open('output.txt', 'w')

    out\_file.write(out)

    out\_file.close()

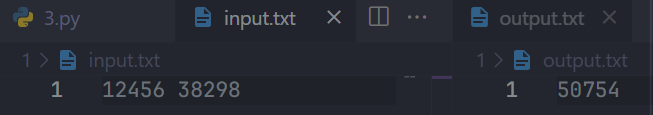
else:

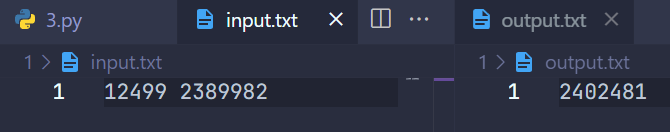
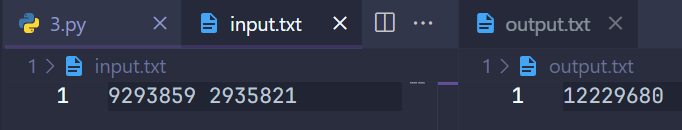
    print("Числа должны быть от -10\*\*9 до 10\*\*9!")

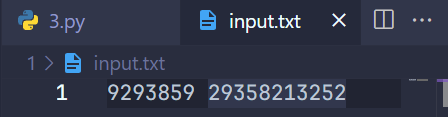
Текстовое объяснение решения:

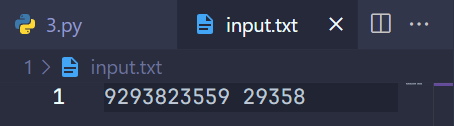
Открываем файл и считываем строку, которую разделяем по пробелу и каждый элемент приводим к числовому типу. Запоминаем 1-ый в переменную a, а 2-ой в переменную b, закрываем файл. Проверяем, удовлетворяют ли числа условию задачи (быть в диапазоне от -10\*\*9 до 10\*\*9). Если удовлетворяют, то считываем их сумму, приводим к строчному типу и запоминаем в переменную out. Создаём/открываем (если уже существует) файл output.txt, записываем в него значение out и закрываем файл. Если изначальные числе не удовлетворяют условию – выводим ошибку.

Результат работы кода, учитывая случаи несоответствия изначальных чисел условию:



Задача a + b2 с использованием файлов

Листинг кода:

a,b = map(int, input().split())

inp\_file = open('input.txt')

a,b = map(int, inp\_file.readline().split())

inp\_file.close()

if -10\*\*9<=a<=10\*\*9 and -10\*\*9<=b<=10\*\*9:

    out = str(a+b\*\*2)

    out\_file = open('output.txt', 'w')

    out\_file.write(out)

    out\_file.close()

else:

    print("Числа должны быть от -10\*\*9 до 10\*\*9!")

inp\_file = open('input.txt')

a,b = map(int, inp\_file.readline().split())

inp\_file.close()

if -10\*\*9<=a<=10\*\*9 and -10\*\*9<=b<=10\*\*9:

    out = str(a+b\*\*2)

    out\_file = open('output.txt', 'w')

    out\_file.write(out)

    out\_file.close()

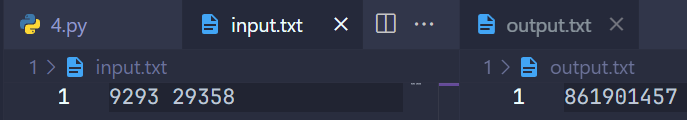
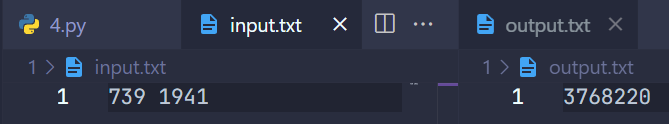
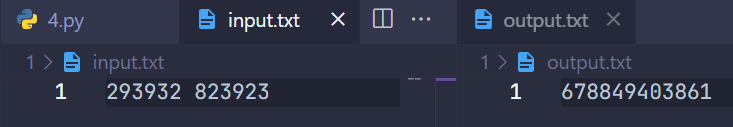
else:

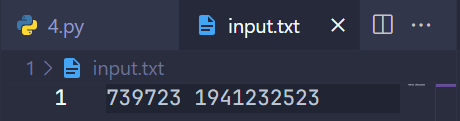
    print("Числа должны быть от -10\*\*9 до 10\*\*9!")

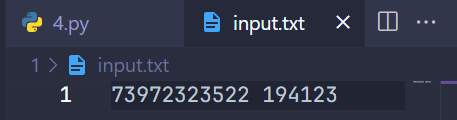
Текстовое объяснение решения:

Открываем файл и считываем строку, которую разделяем по пробелу и каждый элемент приводим к числовому типу. Запоминаем 1-ый в переменную a, а 2-ой в переменную b, закрываем файл. Проверяем, удовлетворяют ли числа условию задачи (быть в диапазоне от -10\*\*9 до 10\*\*9). Если удовлетворяют, то считываем сумму 1-го и квадрата 2-го числа, приводим к строчному типу и запоминаем в переменную out. Создаём/открываем (если уже существует) файл output.txt, записываем в него значение out и закрываем файл. Если изначальные числе не удовлетворяют условию – выводим ошибку.

Результат работы кода, учитывая случаи несоответствия изначальных чисел условию:

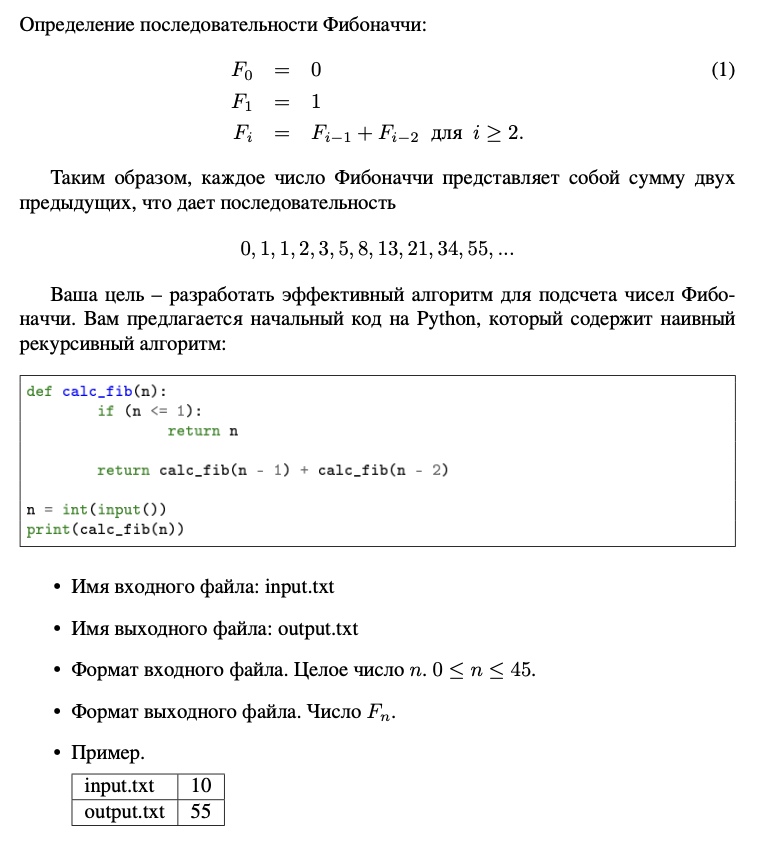
 

Вывод по задаче:

1. Сложность алгоритма – линейная. Чем больше числа, поступающие на вход, тем дольше выполняется программа.
2. При работе с файлами Python’у требуется совершить больше действий, чем при чтении данных из консоли.

## [Задача №2. Число Фибоначчи](#_Задача_№2._Число)



Листинг кода:

import time

t\_start = time.perf\_counter()

def calc\_fib(n):

    lst = [0,1]

    for i in range(1,n):

        lst.append(lst[i-1]+lst[i])

    return lst[n]

inp\_file = open('input.txt')

inp = int(inp\_file.readline())

inp\_file.close()

if 0<=inp<=45:

    out = str(calc\_fib(inp))

    out\_file = open('output.txt', 'w')

    out\_file.write(out)

    out\_file.close()

else:

    print("Число должно быть от 0 до 45!")

print("Время работы: %s секунд " % (time.perf\_counter()-t\_start))

import time

t\_start = time.perf\_counter()

def calc\_fib(n):

    lst = [0,1]

    for i in range(1,n):

        lst.append(lst[i-1]+lst[i])

    return lst[n]

inp\_file = open('input.txt')

inp = int(inp\_file.readline())

inp\_file.close()

if 0<=inp<=45:

    out = str(calc\_fib(inp))

    out\_file = open('output.txt', 'w')

    out\_file.write(out)

    out\_file.close()

else:

    print("Число должно быть от 0 до 45!")

print("Время работы: %s секунд " % (time.perf\_counter()-t\_start))

Текстовое объяснение решения:

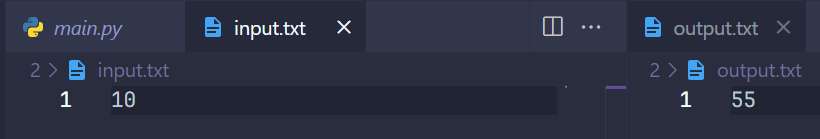
Для замера времени работы программы импортируем библиотеку time. Создаем переменную t\_start, в которую записываем время, пройденное от начала запуска программы. В конце берем время работы программы с помощью time.perf\_counter() и вычитаем из него время начала. Выводим время работы в секундах.

Открываем файл, считываем строку и приводим ее к числовому типу, файл закрываем. Если изначальные числе не удовлетворяют условию задачи – выводим ошибку. Иначе записываем в переменную out результат работы функции calc\_fib, приведенный к строковому типу. Создаём/открываем (если уже существует) файл output.txt, записываем в него значение out и закрываем файл.

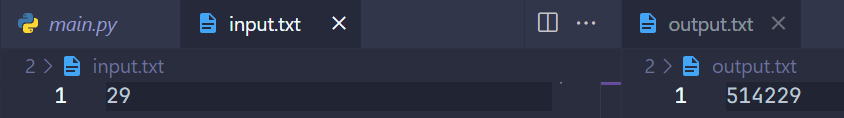
Функция calc\_fib:

Принимает число в переменную n. Создаёт список lst из 2-х элементов – первые 2 элемента последовательности Фибоначчи. Запускает цикл от 1 до n и добавляет в список следующий элемент последовательности путем суммирования предыдущих двух. Возвращает из функции n-ый элемент списка, содержащего последовательность.

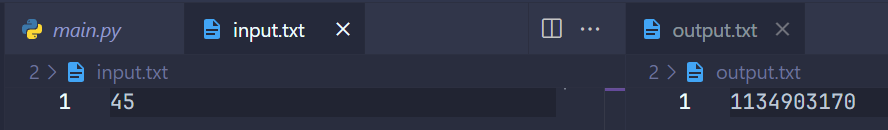
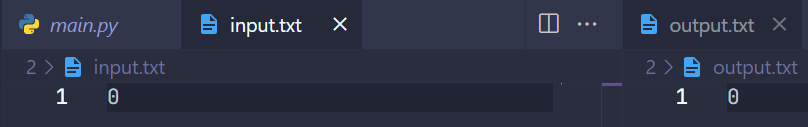
Результат работы кода:









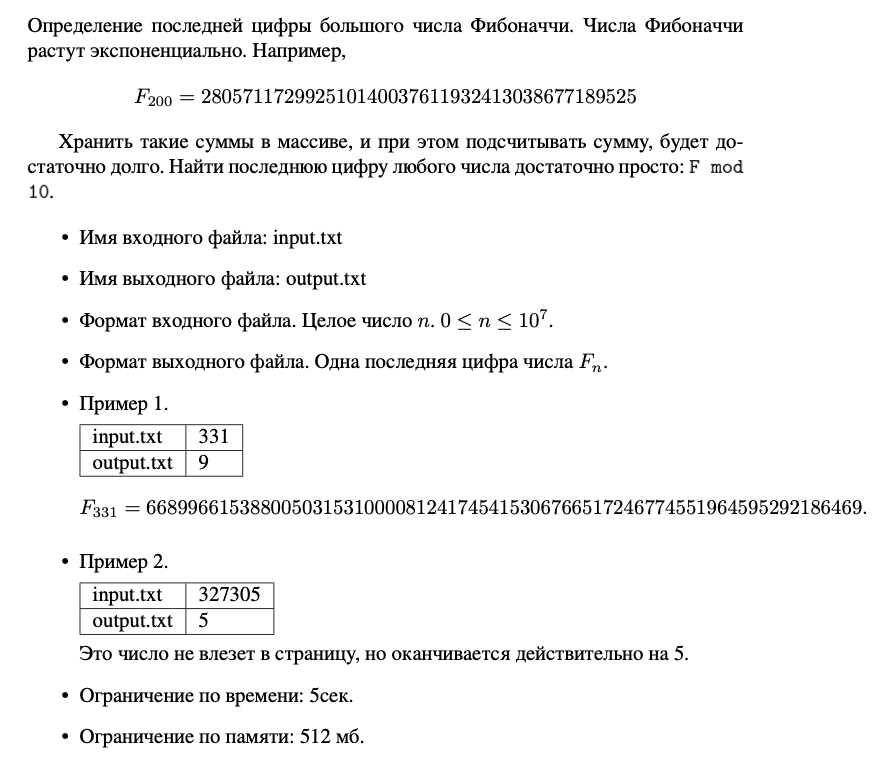
   

|  |  |
| --- | --- |
|  | Время выполнения, c |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.0007419000030495226 |
| Пример из задачи | 0.0007886000021244399 |
| Пример из задачи | 0.0008748999971430749 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0011378000053809956 |

Вывод по задаче:

1. Скорость выполнения алгоритма растет линейно: чем больше мы подаём на вход число n, тем дольше выполняется программа. Это обусловлено сложностью программы O(n).

## [Задача №3. Еще про числа Фибоначчи](#_Задача_№3._Еще)



Листинг кода:

import time

t\_start = time.perf\_counter()

def calc\_fib(n):

    a,b = 0,1

    s=0

    for i in range(n-1):

        s=(a+b)%10

        a,b = b,s

    return s

inp\_file = open('input.txt')

inp = int(inp\_file.readline())

inp\_file.close()

if 0<=inp<=10\*\*7:

    out = str(calc\_fib(inp))

    out\_file = open('output.txt', 'w')

    out\_file.write(out)

    out\_file.close()

else:

    print("Число должно быть от 0 до 10\*\*7!")

print("Время работы: %s секунд " % (time.perf\_counter()-t\_start))

import time

t\_start = time.perf\_counter()

def calc\_fib(n):

    a,b = 0,1

    s=0

    for i in range(n-1):

        s=(a+b)%10

        a,b = b,s

    return s

inp\_file = open('input.txt')

inp = int(inp\_file.readline())

inp\_file.close()

if 0<=inp<=10\*\*7:

    out = str(calc\_fib(inp))

    out\_file = open('output.txt', 'w')

    out\_file.write(out)

    out\_file.close()

else:

    print("Число должно быть от 0 до 10\*\*7!")

print("Время работы: %s секунд " % (time.perf\_counter()-t\_start))

Текстовое объяснение решения:

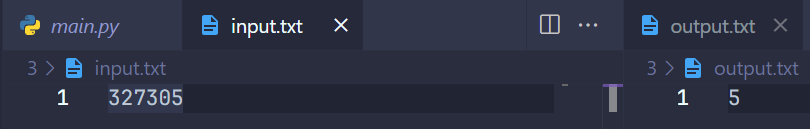
Для замера времени работы программы импортируем библиотеку time. Создаем переменную t\_start, в которую записываем время, пройденное от начала запуска программы. В конце берем время работы программы с помощью time.perf\_counter() и вычитаем из него время начала. Выводим время работы в секундах.

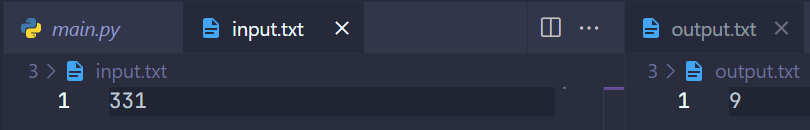
Открываем файл, считываем строку и приводим ее к числовому типу, файл закрываем. Если изначальные числе не удовлетворяют условию задачи – выводим ошибку. Иначе записываем в переменную out результат работы функции calc\_fib, приведенный к строковому типу. Создаём/открываем (если уже существует) файл output.txt, записываем в него значение out и закрываем файл.

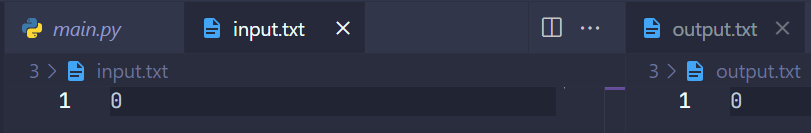
Функция calc\_fib:

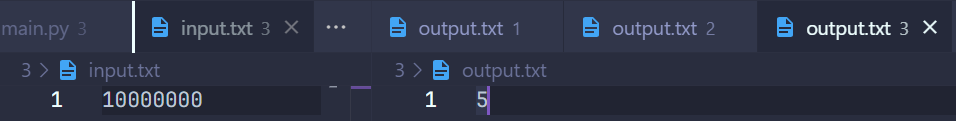
Принимает число в переменную n. Создает 2 переменные a и b, содержащие 1 и 2 элемент последовательности Фибоначчи соответственно. Создает переменную s, в которой будет содержаться результат функции. Запускает цикл от 0 до n-1, записывает остаток от деления на 10 суммы элементов последовательности (последняя цифра числа). Далее записывает в переменную a значение переменной b, а в b – значение s. Делает так, ибо в результате нам нужна будет только последняя цифра числа, а остальные для нас не важны, следовательно мы можем работать только с последними цифрами (остатками от деления на 10). После выполнения цикла возвращает результат функции, содержащийся в s.

Результат работы кода:

|  |  |
| --- | --- |
|  | Время выполнения, c |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.0007362999967881478 |
| Пример из задачи | 0.02355320000060601 |
| Пример из задачи | 0.001356699998723343 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.70531750000373 |

Вывод по задаче:

1. Скорость выполнения алгоритма растет линейно: чем больше мы подаём на вход число n, тем дольше выполняется программа. Это обусловлено сложностью программы O(n).

## [Задача №4. Тестирование ваших алгоритмов](#_Задача_№4._Тестирование)

Текст задачи:

Вам необходимо протестировать время выполнения вашего алгоритма в Задании 2 и Задании 3.

Листинг кода:

import time

t\_start = time.perf\_counter()

//Код программы

print("Время работы: %s секунд " % (time.perf\_counter()-t\_start))

import time

t\_start = time.perf\_counter()

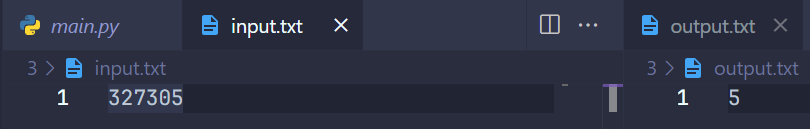
//Код программы

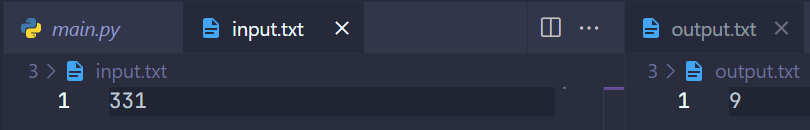
print("Время работы: %s секунд " % (time.perf\_counter()-t\_start))

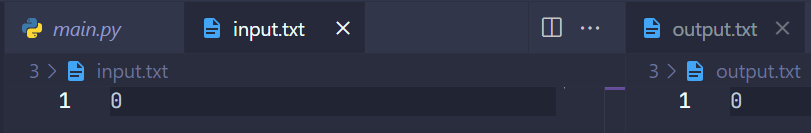
Текстовое объяснение решения:

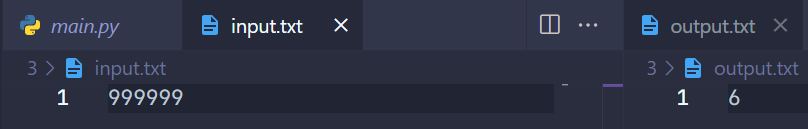
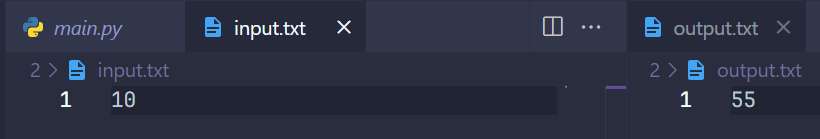
Для замера времени работы программы импортируем библиотеку time. Создаем переменную t\_start, в которую записываем время, пройденное от начала запуска программы. В конце берем время работы программы с помощью time.perf\_counter() и вычитаем из него время начала. Выводим время работы в секундах.

Результат работы кода:

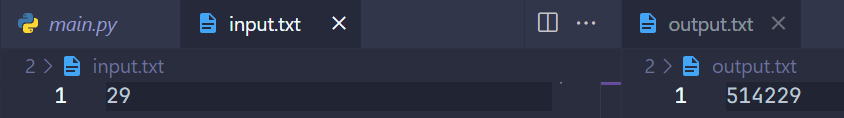
 

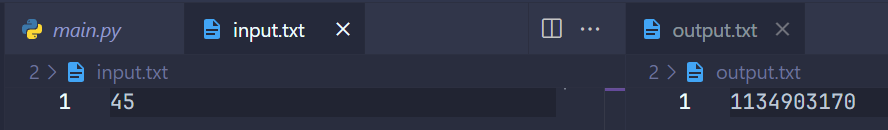
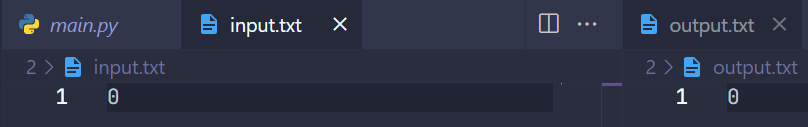
 







Вывод по задаче:

1. Замерять время выполнения алгоритма довольно полезно. Оно позволяет понять, оптимизируешь ты код или наоборот излишне его нагружаешь.

# Вывод

В ходе выполнения работы были получены знания по выполнению задач на ввод/вывод, работу с файлами и последовательностью Фибоначчи. Была изучена библиотека time для измерения времени работы алгоритма и применена на практике для оптимизации программ. Также была показана на практике линейная зависимость времени работы алгоритма сложностью O(n) от входных данных: чем больше числа на входе, тем больше итераций выполняет программа, следовательно, тем дольше она получает результат.