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 21

Выполнила:
Савченко А.С.

Санкт-Петербург
2024 г.

Содержание отчета

Задачи по варианту

Задача №1. Ввод-вывод

Задание 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 $-10^9 \leq a, b \leq 10^9$. Выход: единственное целое число — результат сложения $a + b$.
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 $-10^9 \leq a, b \leq 10^9$. Выход: единственное целое число — результат сложения $a + b^2$.
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 $-10^9 \leq a, b \leq 10^9$.
   - Формат выходного файла. Выходной файл единственное целое число — результат сложения $a + b$.

Примеры.

input.txt	12 25	130 61
output.txt	37	191

Задание 1. a + b.

Текст задачи:

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.

Листинг кода:

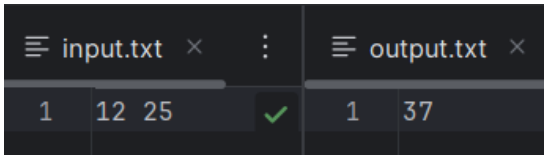
```
with open('input.txt', 'r', encoding='utf-8') as
input_file, open('output.txt', 'w',
```

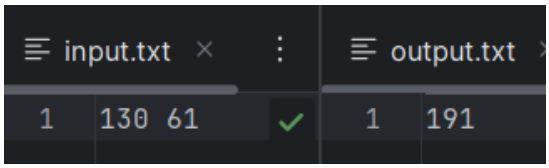
```
encoding='utf-8') as output_file:
    a, b = map(int, input_file.read().split())
    output_file.write(str(a + b))
```

Текстовое объяснение решения:

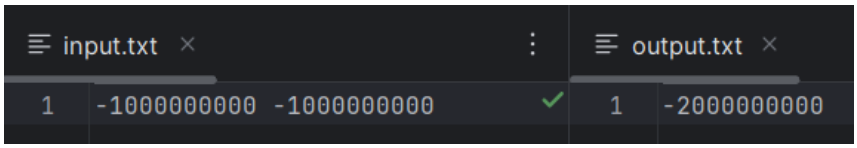
Считываем два числа из файла 'input.txt' и записываем результат их сложения в файл 'output.txt'. Для открытия и закрытия файла используем менеджер контекста with.

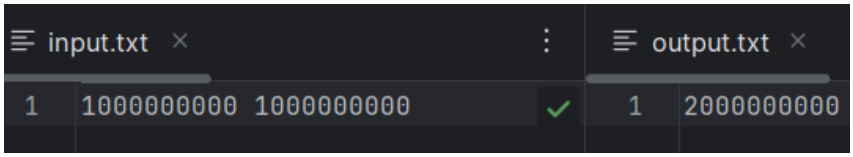
Результат работы кода на примерах из текста задачи:





Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





Задание 2. a + b^2.

Текст задачи:

В данной задаче требуется вычислить значение a + b2. Вход: одна строка, которая содержит два целых числа a и b. Для этих чисел вы-

полняются условия −109 ≤ a, b ≤ 109. Выход: единственное целое число — результат сложения a + b2..

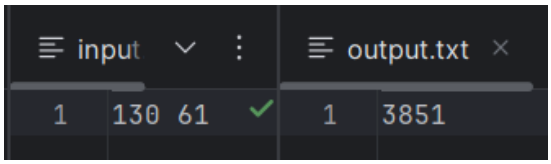
Листинг кода:

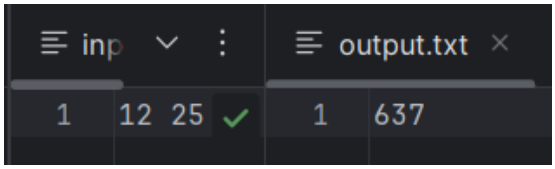
```
with open('input.txt', 'r', encoding='utf-8') as
input_file, open('output.txt', 'w',

encoding='utf-8') as output_file:
    a, b = map(int, input_file.read().split())
    output_file.write(str(a + b ** 2))
```

Текстовое объяснение решения:

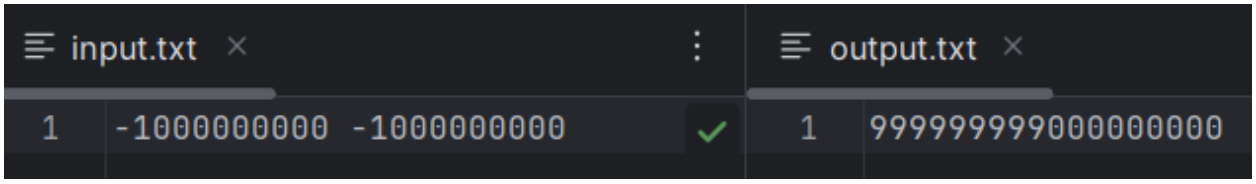
Аналогично предыдущей задаче считываем два числа из файла 'input.txt' и записываем результат сложения a + b^2 в файл 'output.txt'.

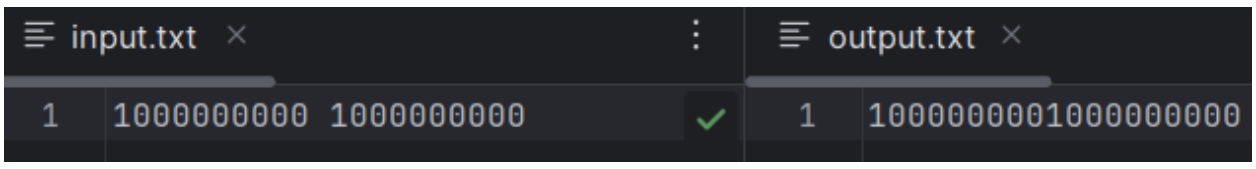
Результат работы кода на примерах из текста задачи:





Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





Вывод по задаче:

Задача направлена на работу с файлами, знание режимов работы с файлами, считывание и запись данных из и в файлы.

Задача №2. Число Фибоначчи

Текст задачи:

Задание 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:

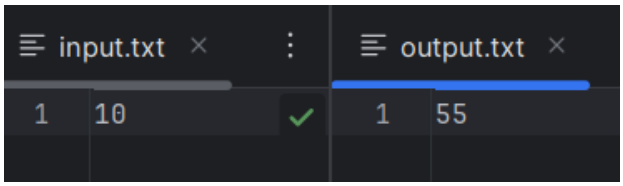
```
with open('input.txt', 'r', encoding='utf-8') as input_file, open('output.txt', 'w', encoding='utf-8') as output_file:
```

```
    n = int(input_file.read())
    fibonacci = [0, 1]
    for i in range(2, n + 1):
                        fibonacci.append((fibonacci[-1]    +
fibonacci[-2]))

    output_file.write(str(fibonacci[n]))
```

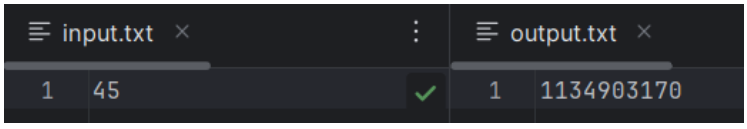
Текстовое объяснение решения:

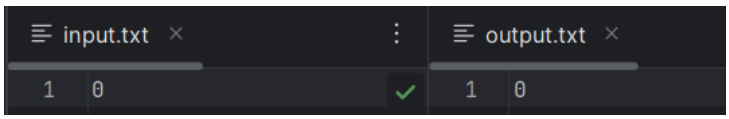
Читаем числа из входного файла. Создаем список fibonacci, содержащий первые два числа последовательности. В цикле вычисляем и добавляем числа Фибоначчи в список до n-го элемента, складывая два предыдущих числа. Записываем n-е число в выходной файл

Результат работы кода на примерах из текста задачи:



Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





	Время выполнения, сек	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи	0.000438600080	14.4609375
Пример из задачи	0.00039980	14.453125
Верхняя граница диапазона значений	0.00039870012551	14.19140625

входных данных из текста задачи		

Вывод по задаче:

Я вспомнила разные алгоритмы для нахождения числа фибоначчи и выбрала более эффективный.

Задача №3. Еще про числа Фибоначчи

Текст задачи:

Задание 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:

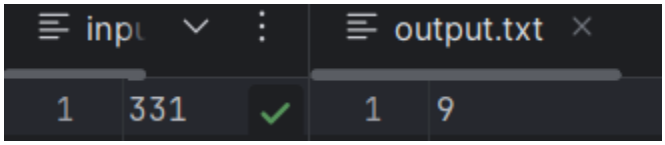
```
with open('input.txt', 'r', encoding='utf-8') as
input_file, open('output.txt', 'w', encoding='utf-8')
as output_file:
    n = int(input_file.read())
    fibonacci = [0, 1]
    for i in range(2, n + 1):
                    fibonacci.append((fibonacci[-1] +
fibonacci[-2]) % 10)
```

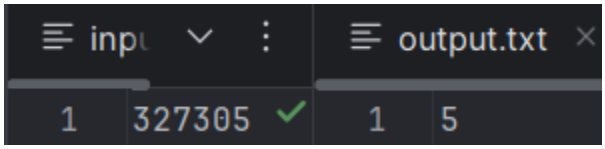
```
output_file.write(str(fibonacci[n]))
```

Текстовое объяснение решения:

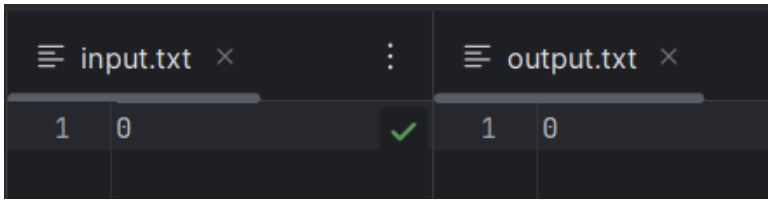
Используем алгоритм предыдущей задачи, но найдем не значение F(n), а последнюю цифру числа Фибоначчи используя %10.

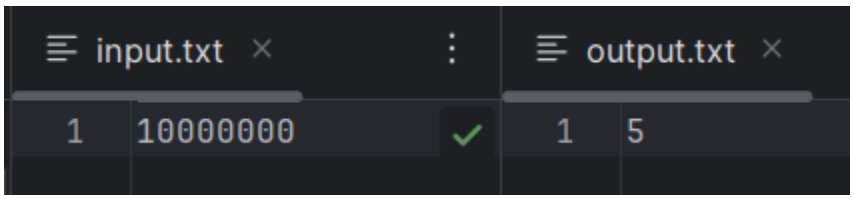
Результат работы кода на примерах из текста задачи:





Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





	Время выполнения, сек	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0004126999	14.3359375
Пример из задачи	0.000906700	14.183593

Пример из задачи	0.0342552999	16.91015
Верхняя граница диапазона значений входных данных из текста задачи	1.09075299999	90.7265625

Вывод по задаче:

Задача №4. Тестирование ваших алгоритмов

Текст задачи:

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 psutil import Process
from os import getpid

start_time = perf_counter_ns()


def testing():
    with open('input.txt', 'r', encoding='utf-8') as
input_file, open('output.txt', 'w',

encoding='utf-8') as output_file:
        n = int(input_file.read())
        fibonacci = [0, 1]
        for i in range(2, n + 1):
                    fibonacci.append((fibonacci[-1] +
fibonacci[-2]) % 10)
```

```
        output_file.write(str(fibonacci[n]))


testing()
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:

Для подсчеты затрат времени и памяти импортируем необходимые библиотеки, а точнее лишь некоторые функции из их . Для измерения времени перед основным блоком кода используем start_time = perf_counter_ns(), чтобы после отнять его от общего времени работы. Для выводы переводим байты в мегабайты.

Вывод по задаче:

Я узнала о разных способах подсчета времени и памяти в Python. Протестировала таким образом задачи 2, 3.

Вывод (по всей лабораторной)

Лабораторная позволяет вспомнить работу с файлами в python, а также показывает как измерять затраты времени и памяти.