

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

Отчет по лабораторной работе №0
по курсу «Алгоритмы и структуры данных»
Тема: Введение

Выполнила:
Фокина Анастасия
К3160

Проверил:
Аминов Натиг Сабит оглы

Санкт-Петербург
2025 г.

Содержание отчета

Содержание отчета	3
Задачи по варианту	4
Задача №1. Ввод-вывод	4
Задача №2. Число Фибоначчи	8
Задача №3. Еще про числа Фибоначчи	11
Вывод:	15

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 $-109 \leq a, b \leq 109$. 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 $-109 \leq a, b \leq 109$. 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 $-109 \leq a, b \leq 109$.
- Формат выходного файла. Выходной файл единственное целое число — результат сложения $a + b$.

Примеры:

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

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

Листинг кода для пунктов 1 и 3:

```

with open("input.txt") as f:
    a,b = map(int,f.read().split(" "))
    if all(-10**9 <=x<= 10**9 for x in [a,b] ):
        print(a+b)
    with open("output.txt", "a+") as f:
        f.write(str(a+b)+"\n")

```

Листинг кода для пунктов 2 и 4:

```

with open("input.txt") as f:
    a,b = map(int,f.read().split(" "))
    if all(-10**9 <=x<= 10**9 for x in [a,b] ):
        print(a+b**2)
    with open("output.txt", "a+") as f:
        f.write(str(a+b**2)+"\n")

```

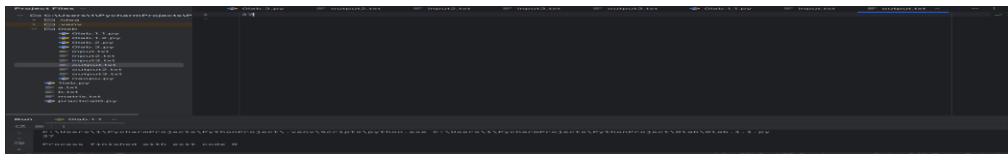
- Сначала открываем файл input.txt для чтения (по умолчанию), в нем должно быть записано 2 числа.
- Далее программа считывает 2 числа из файла, записанные через пробел, преобразуя их в целые числа a и b.
- Записываем условие $-10^{**}9 \leq x \leq 10^{**}9$ для a и b
- Если условие выполняется, то программа вычисляет сумму и выводит результат в консоль
- Затем снова открываем файл, но на этот раз output.txt для записи в режиме "a+" данные записываются в конец файла каждый раз (можно использовать "w" для перезаписи)
- Записываем туда сумму, преобразованную в строку str() и чтобы каждое новое значение записывалось на новой строке используем "\n"

Результат работы кода на примерах из текста задачи:

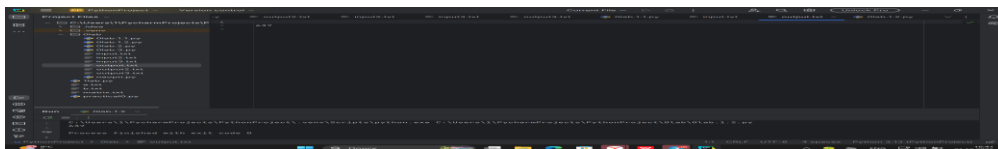
1. При значениях 12 25



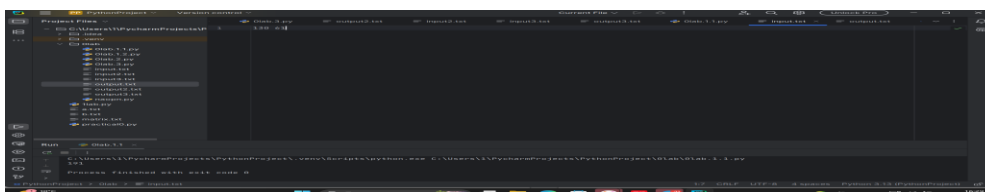
Для $(a+b)$:



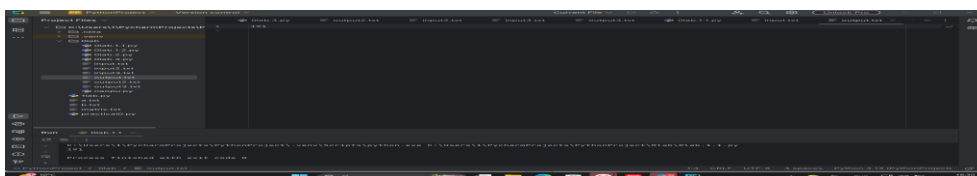
Для $(a+b**2)$:



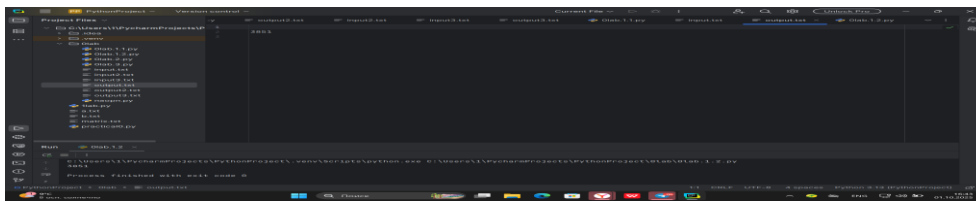
2. При значениях 130 61



Для $(a+b)$:

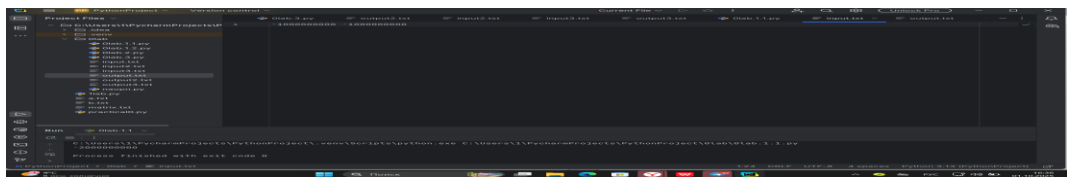


Для $(a+b**2)$:

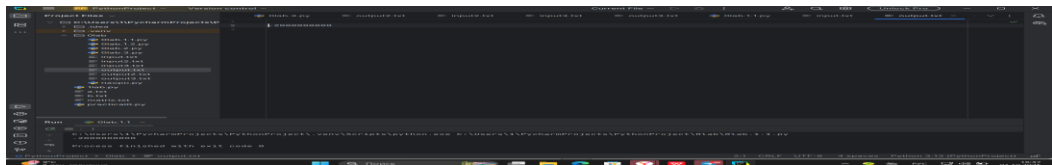


Результат работы кода на максимальных и минимальных значениях:

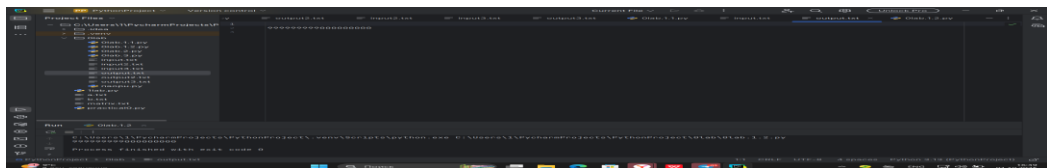
1. На минимальных значениях



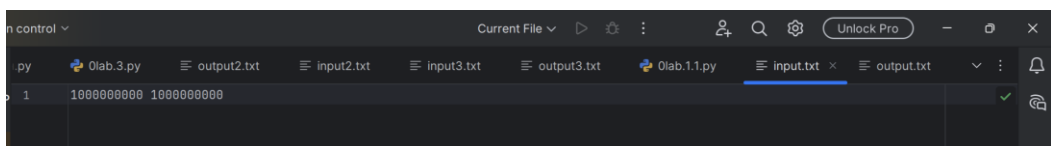
Для $(a+b)$:



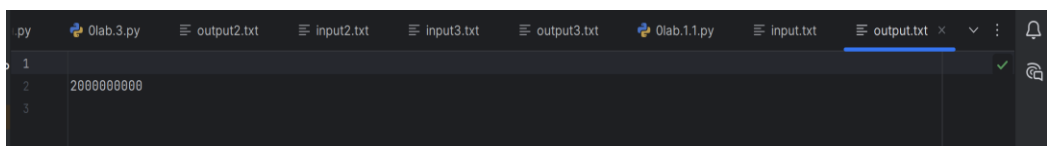
Для $(a+b**2)$:



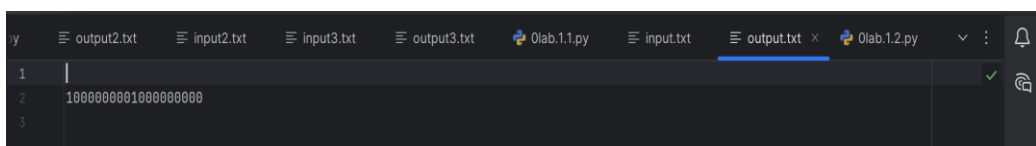
2. На максимальных значениях



Для $(a+b)$:



Для $(a+b**2)$:



Вывод по задаче: программа выполняет сложение двух чисел из входного файла с проверкой граничных условий и сохраняет результат в выходном файле и эффективно с этим справляется.

Задача №2. Число Фибоначчи

Определение последовательности Фибоначчи:

$$F_0 = 0 \quad (1) \quad F_1 = 1 \quad F_i = F_{i-1} + F_{i-2} \text{ для } i \geq 2.$$

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. Вам предлагается начальный код на Python, который содержит наивный рекурсивный алгоритм.

- 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:

```
import time
```



```

start_time = time.perf_counter()
with open("input2.txt") as f:
    n = int(f.read())
def calc_fib(n):
    if n <= 1:
        return n
    a, b = 0, 1
    for i in range(2, n + 1):
        a, b = b, a + b
    return b
if 0 <=n<= 45:
    res = calc_fib(n)
    print(res)
    with open("output2.txt", "a+") as f:
        f.write(str(res) + "\n")
end_time = time.perf_counter()
execution_time = (end_time - start_time) * 1000
print(f"Execution time: {execution_time} ms")

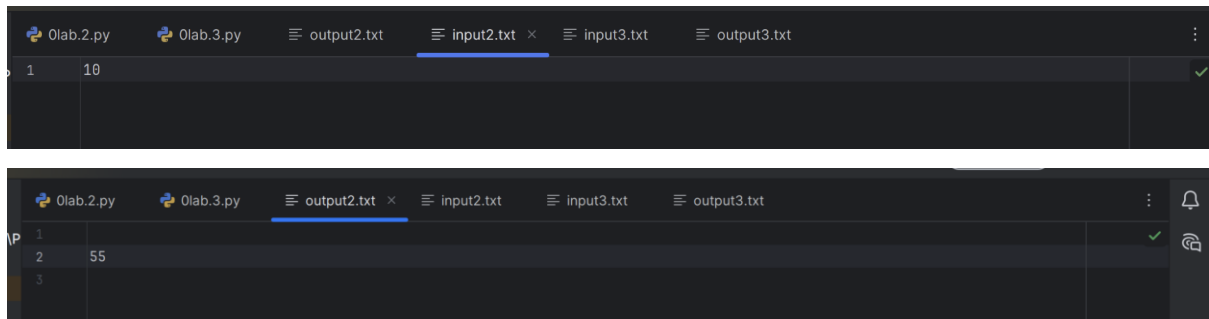
```

- Открываем файл input2.txt и считываем оттуда внесенное значение в переменную n
- Далее пишем функцию для вычисления определенного числа Фибоначчи: если значение $n \leq 1$ - возвращаем само число, если же нет, то используем две переменные a и b.
- Используем интеративный подход, a хранит $F(i-2)$, b хранит $F(i-1)$. На каждой итерации обновляем значение, a хранит предыдущее b, a b сумму двух предыдущих $a+b$
- Проверяем, что $0 \leq n \leq 45$ согласно условию.
- Выводим результат в консоль, присваивая название функции

переменной `res`

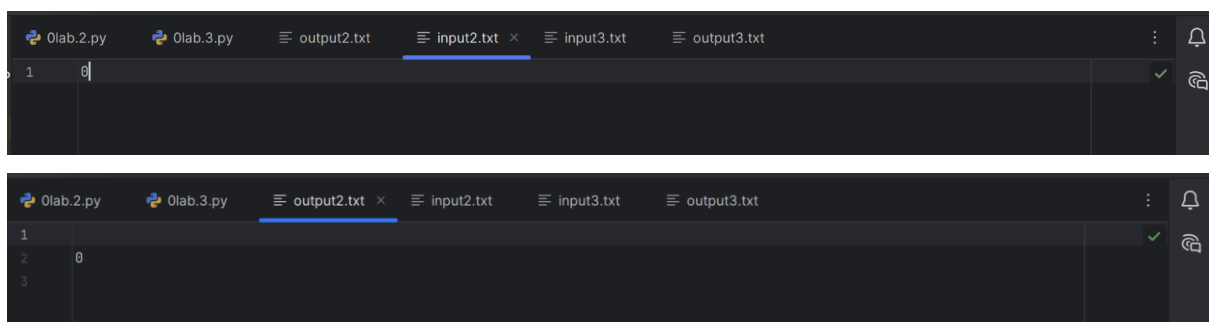
- Затем снова открываем файл, но на этот раз `output2.txt` для записи в режиме “a+” данные записываются в конец файла каждый раз (можно использовать “w” для перезаписи)
- Записываем туда значение `res`, преобразованное в строку `str()` и чтобы каждое новое значение записывалось на новой строке используем “\n”
- Подключаем встроенный модуль `time` для измерения времени, оборачиваем код в `time.perf_counter()`
- Вычисляем время выполнения в миллисекундах домножая на 1000 и выводим результат

Результат работы кода на примерах из текста задачи:

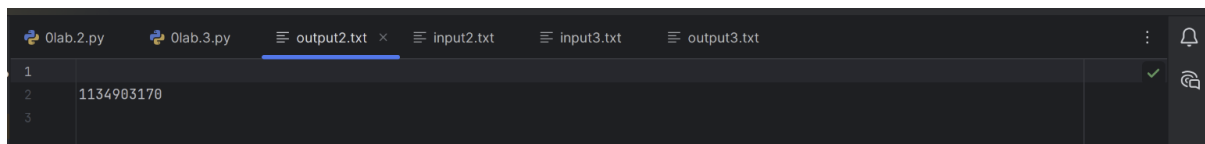
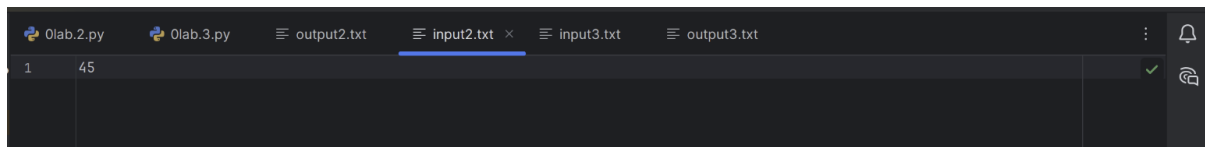


Результат работы кода на максимальных и минимальных значениях:

1. На минимальных



2. На максимальных



	Время выполнения
Нижняя граница диапазона значений входных данных из текста задачи	Execution time: 2.044099965132773 ms
Пример из задачи	Execution time: 2.3045000270940363 ms
Верхняя граница диапазона значений входных данных из текста задачи	Execution time: 5.522899969946593 ms

Вывод по задаче: Данный итеративный алгоритм работает куда эффективнее рекурсивного, так как отсутствуют повторные вычисления, что сокращает затраты времени и памяти.

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: $F \bmod 10$

- 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

$F_{331} =$

6689966153880050315310000812417454153067665172467745519645952921
86469.

- 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.

```
import time

start_time = time.perf_counter()
with open("input3.txt") as f:
    n = int(f.read())
def calc_fib(n):
    n=n%60
    if n <= 1:
        return n
    a, b = 0, 1
    for i in range (2, n+1):
        a, b = b, (a + b)%10
```

```

        return b
if 0 <= n <= 10**7:
    res = calc_fib(n)
    print(res)
    with open("output3.txt", "a+") as f:
        f.write(str(res)+ "\n")
end_time = time.perf_counter()
execution_time = (end_time - start_time) * 1000
print(f"Execution time: {execution_time} ms")

```

- Открываем файл input3.txt и считываем оттуда внесенное значение в переменную n

- Используем период Пизано для модуля 10: $n=n\%60$, эта оптимизация позволит выполнять максимум 60 итераций

Далее пишем функцию для вычисления определенного числа Фибоначчи: если значение $n \leq 1$ - возвращаем само число, если же нет, то используем две переменные a и b.

- Используем интеративный подход, a хранит $F(i-2)$, b хранит $F(i-1)$. На каждой итерации обновляем значение, a хранит предыдущее b, a b остаток от суммы двух предыдущих при делении на 10 $(a+b)\%10$, то есть последнюю цифру числа

- Проверяем, что $0 \leq n \leq 10**7$ согласно условию.

- Выводим результат в консоль, присваивая название функции переменной res

- Затем снова открываем файл, но на этот раз output3.txt для записи в режиме "a+" данные записываются в конец файла каждый раз (можно использовать "w" для перезаписи)

- Записываем туда значение res, преобразованное в строку str() и чтобы каждое новое значение записывалось на новой строке используем "\n"

- Подключаем встроенный модуль time для измерения времени, оборачиваем код в time.perf_counter()

- Вычисляем время выполнения в миллисекундах домножая на 1000 и выводим результат

Результат работы кода на примерах из текста задачи:

1. Пример 1

The first screenshot shows the 'input3.txt' file with the value '331' on line 1. The second screenshot shows the 'output3.txt' file with the value '9' on line 1. Both files are part of a project named '0lab.3.py'.

2. Пример 2

The first screenshot shows the 'input3.txt' file with the value '327305' on line 1. The second screenshot shows the 'output3.txt' file with the value '5' on line 1. Both files are part of a project named '0lab.3.py'.

Результат работы кода на максимальных и минимальных значениях:

1. На минимальных

The first screenshot shows the 'input3.txt' file with the value '0' on line 1. The second screenshot shows the 'output3.txt' file with the value '0' on line 1. Both files are part of a project named '0lab.3.py'.

2. На максимальных

The first screenshot shows the 'input3.txt' file with the value '10000000' on line 1. The second screenshot shows the 'output3.txt' file with the value '5' on line 1. Both files are part of a project named '0lab.3.py'.

	Время выполнения
Нижняя граница диапазона значений входных данных из текста задачи	Execution time: 2.065100008621812 ms
Пример из задачи 1	Execution time: 2.5394000113010406 ms
Пример из задачи 2	Execution time: 2.7141000027768314 ms
Верхняя граница диапазона значений входных данных из текста задачи	Execution time: 3.381399961654097 ms

Вывод по задаче: для того, чтобы понять, какая в необходимом по очереди числе Фибоначчи последняя цифра, мы можем использовать период Пизано для модуля 10, таким образом программа будет выполняться во множество раз быстрее, т.к. всегда происходит не более 60 итераций.

Вывод:

В ходе выполнения работы были повторены различные конструкции Python, такие как циклы, механизм работы с файлами (чтение и запись), ввод данных в одну строку, написание функций, а также тестирование времени работы алгоритма. Была изучена и на практике проверена разница между рекурсивным и итеративным алгоритмом. Стала наглядно понятна разница между константной сложностью $O(1)$ и линейной сложностью

$O(n)$ алгоритмов. Разработанные алгоритмы работают в соответствии с условиями задачи, с разными входными данными.