

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: Вводная работа.

Вариант 1

Выполнил:  
Ананьев Н.В.  
группа К3140

Проверила:  
Артамонова В.Е.

Санкт-Петербург

2022 г.

## Содержание отчета

Содержание отчета_	2
Задачи по варианту_	3
Задание № 1 [ввод-вывод]	3
Задача №1 ( $a + b$ )	3
Задача №2 ( $a + b^2$ )	5
Задание № 2 [Число Фибоначчи]	7
Задание №3 [Еще про числа Фибоначчи]	10
Вывод	13

## Задание №1[ввод-вывод]

Задача №1 [a + b]

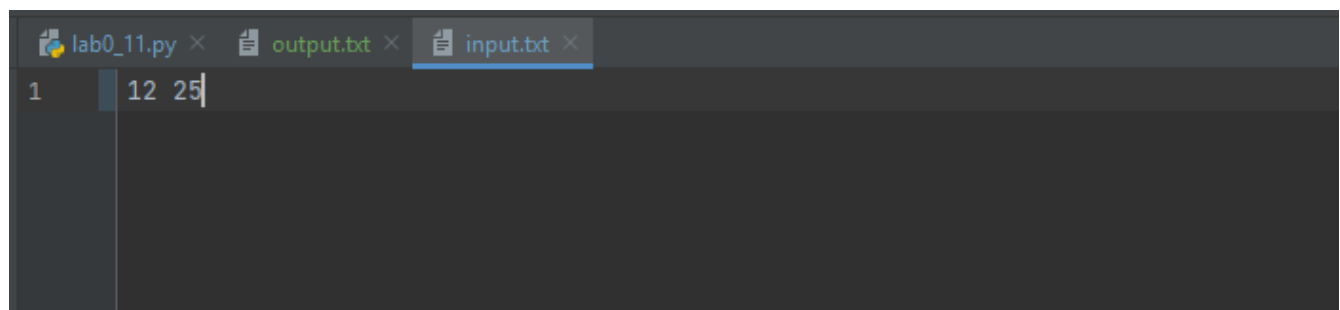
Листинг кода:

```
import time
import psutil

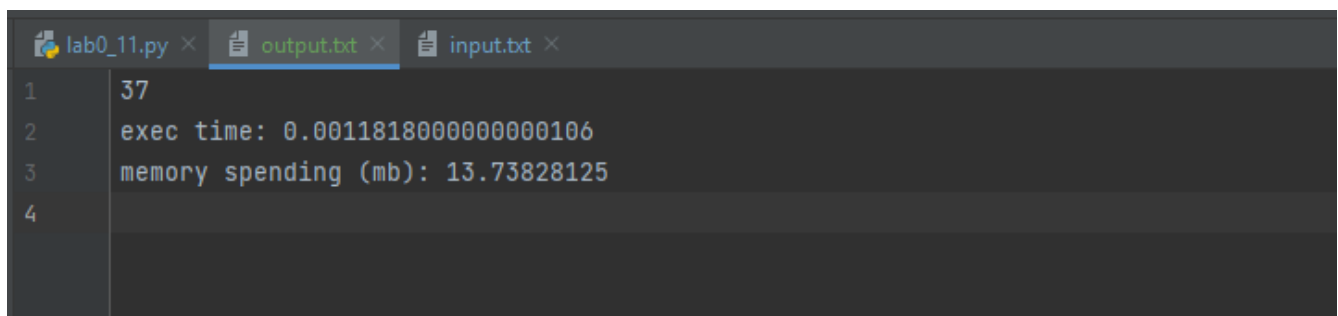
start = time.perf_counter()
with open("input.txt", "r") as file:
    data = file.read()
    a, b = map(int, data.split())
    res = a + b
    with open("output.txt", "w") as out:
        print(res, file=out)
        print("exec time:", time.perf_counter() - start, file=out)
        print("memory spending (mb):", psutil.Process().memory_info().rss / (1024
* 1024), file=out)
```

Программа принимает на вход данные с файла input.txt в виде строки, которая содержит 2 числа , разделенные пробелом. Эти числа преобразуются в тип int, после чего переменной res присваивается сумма чисел. Результат выводится в файл output.txt.

Результат работы кода на примерах из текста задачи:



The screenshot shows a code editor with three tabs: lab0\_11.py, output.txt, and input.txt. The input.txt tab is active, showing the content '12 25' on line 1.



The screenshot shows a code editor with three tabs: lab0\_11.py, output.txt, and input.txt. The output.txt tab is active, showing the output of the program on lines 1, 2, and 3: '37', 'exec time: 0.0011818000000000106', and 'memory spending (mb): 13.73828125'.

Результат работы кода на максимальных и минимальных значениях:

```
lab0_11.py × output.txt × input.txt ×  
1 -1000000000 -1000000000  
  ⚡
```

```
lab0_11.py × output.txt × input.txt ×  
1 -2000000000  
2 exec time: 0.0007377000000000078  
3 memory spending (mb): 13.75390625  
4
```

```
lab0_11.py × output.txt × input.txt ×  
1 1000000000 1000000000  
  ⚡
```

```
lab0_11.py × output.txt × input.txt ×  
1 2000000000  
2 exec time: 0.0004423000000000066  
3 memory spending (mb): 13.7890625  
4
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0008565с	13.7265625 mb
Пример из задачи (1)	0.0005882с	13.734375 mb
Пример из задачи (2)	0.0007373с	13.765625 mb
Верхняя граница диапазона значений входных данных из текста задачи	0.0008315с	13.753906 mb

Задача №2  $[a + b^2]$

Листинг кода:

```
import time
import psutil

start = time.perf_counter()
with open("input.txt", "r") as file:
    data = file.read()
    a, b = map(int, data.split())
    res = a + b * b
    with open("output.txt", "w") as out:
        print(res, file=out)
        print("exec time:", time.perf_counter() - start, file=out)
        print("memory spending (mb):", psutil.Process().memory_info().rss / (1024 *
1024), file=out)
```

Программа аналогична программе для первой задачи, за исключением того, что складывается первое число и квадрат второго.

Результат работы кода на максимальных и минимальных значениях:

```
lab0_11.py × output.txt × lab0_12.py × input.txt ×
1 -10000000000 -10000000000
  ⚡
```

```
lab0_11.py × output.txt × lab0_12.py × input.txt ×
1 999999999000000000
2 exec time: 0.0004156000000000021
3 memory spending (mb): 13.796875
4
```

```
lab0_11.py × output.txt × lab0_12.py × input.txt ×
1 10000000000 10000000000
  ⚡
```

```
lab0_11.py × output.txt × lab0_12.py × input.txt ×
1 10000000001000000000
2 exec time: 0.0005949000000000093
3 memory spending (mb): 13.73828125
4
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000607с	13.725625 mb
Пример из задачи (1)	0.0004951с	13.742187 mb
Пример из задачи (2)	0.0005626с	13.769531 mb
Верхняя граница диапазона значений входных данных из текста задачи	0.0005117с	13.718753 mb

## Задание №2 [Число Фибоначчи]

Листинг кода:

```
import time
import psutil

start = time.perf_counter()
Fib = [0] * 50
Fib[0], Fib[1] = 0, 1
with open("input.txt", "r") as file:
    n = int(file.read())
    if n >= 2:
        for i in range(2, n + 1):
            Fib[i] = Fib[i - 1] + Fib[i - 2]
        with open("output.txt", "w") as out:
            print(Fib[n], file=out)
            print("exec time:", time.perf_counter() - start, file=out)
            print("memory spending (mb):", psutil.Process().memory_info().rss / (1024 * 1024), file=out)
```

Для вычисления чисел Фибоначчи создается глобальный список, количество элементов в котором заранее  $>$  входного  $n$ . Первые два элемента последовательности занесены в соответствующие им ячейки списка. Следующие элементы вычисляются рекуррентно из суммы двух предыдущих и сразу записываются в соответствующую ячейку.

Результат работы кода на примерах из текста задачи:

```
lab0_11.py × output.txt × lab0_2.py × lab0_12.py × input.txt ×
1 10
```

```
lab0_11.py × output.txt × lab0_2.py × lab0_12.py × input.txt ×
1 55
2 exec time: 0.0009648000000000156
3 memory spending (mb): 13.74609375
4
```

Результат работы кода на максимальных и минимальных значениях:

```
lab0_11.py × output.txt × lab0_2.py × lab0_12.py × input.txt ×
1 1
  ⚡
```

```
lab0_11.py × output.txt × lab0_2.py × lab0_12.py × input.txt ×
1 1
2 exec time: 0.00046260000000000745
3 memory spending (mb): 13.78515625
4
```



```
lab0_11.py × output.txt × lab0_2.py × lab0_12.py × input.txt ×
1 45
```

```
lab0_11.py × output.txt × lab0_2.py × lab0_12.py × input.txt ×
1 1134903170
2 exec time: 0.00058459999999999907
3 memory spending (mb): 13.75
4
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0004626с	13.785156 mb
Пример из задачи (1)	0.0009648с	13.746093 mb
Верхняя граница диапазона значений входных данных из текста задачи	0.0005846с	13.75 mb

Вывод по задаче: для решения задачи удалось реализовать алгоритм с линейной сложностью. Затраты по памяти незначительны, т. к. по условию  $n \leq 45$ .

### Задание №3 [Еще про числа Фибоначчи]

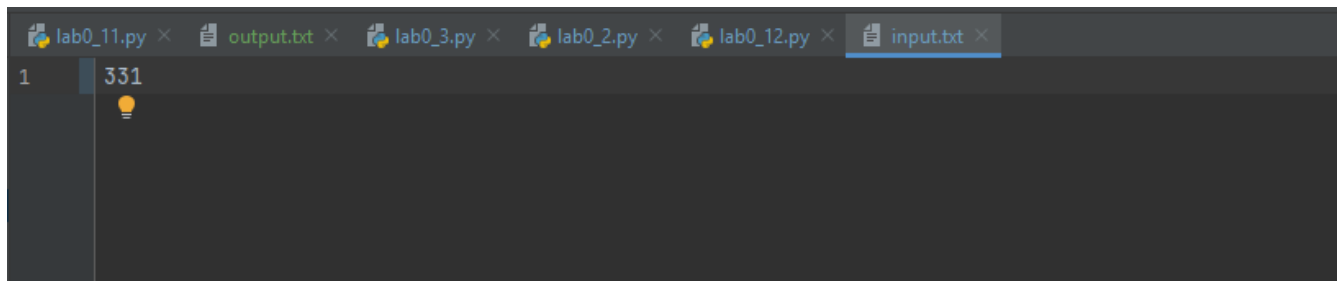
Листинг кода:

```
import time
import psutil

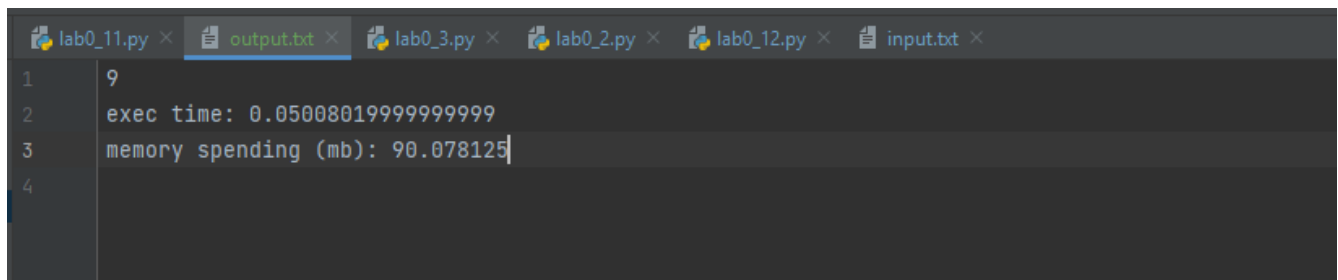
start = time.perf_counter()
Fib = [0] * (10**7 + 1)
Fib[0], Fib[1] = 0, 1
with open("input.txt", "r") as file:
    n = int(file.read())
    if n >= 2:
        for i in range(2, n + 1):
            Fib[i] = (Fib[i - 1] + Fib[i - 2]) % 10
        with open("output.txt", "w") as out:
            print(Fib[n], file=out)
            print("exec time:", time.perf_counter() - start, file=out)
            print("memory spending (mb):", psutil.Process().memory_info().rss / (1024 *
1024), file=out)
```

Решение почти не отличается от предыдущей задачи, за исключением того, что записывается не само число, а его остаток от деления на 10.

Результат работы кода на примерах из текста задачи:



The screenshot shows a code editor with several tabs: lab0\_11.py, output.txt, lab0\_3.py, lab0\_2.py, lab0\_12.py, and input.txt. The 'input.txt' tab is active, displaying the number 331 on line 1.



The screenshot shows the same code editor with the 'output.txt' tab active. It displays the results of the program execution on four lines: 9, exec time: 0.05008019999999999, memory spending (mb): 90.078125, and a blank line 4.

```
lab0_11.py × output.txt × lab0_3.py × lab0_2.py × lab0_12.py × input.txt ×
1 327305
```

```
lab0_11.py × output.txt × lab0_3.py × lab0_2.py × lab0_12.py × input.txt ×
1 5
2 exec time: 0.1671803
3 memory spending (mb): 90.07421875
4
```

Результат работы кода на максимальных и минимальных значениях:

```
lab0_11.py × output.txt × lab0_3.py × lab0_2.py × lab0_12.py × input.txt ×
1 1
```

```
lab0_11.py × output.txt × lab0_3.py × lab0_2.py × lab0_12.py × input.txt ×
1 1
2 exec time: 0.05146149999999999
3 memory spending (mb): 90.06640625
4
```

```
lab0_11.py × output.txt × lab0_3.py × lab0_2.py × lab0_12.py × input.txt ×
1 10000000
```

```
lab0_11.py × output.txt × lab0_3.py × lab0_2.py × lab0_12.py × input.txt ×
1 5
2 exec time: 3.1654095
3 memory spending (mb): 90.06640625
4
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.051461с	90.066406 mb
Пример из задачи (1)	0.05с	90.07 mb
Пример из задачи (2)	0.1671с	90.07 mb
Верхняя граница диапазона значений входных данных из текста задачи	3.1654с	90.066 mb

Вывод по задаче: сложность алгоритма остается линейной, как и затраты по памяти. Второе является слабым местом в решении, возможна оптимизация.

Вывод: в ходе выполнения лабораторной работы я узнал как нужно оформлять работу, потренировался в использовании системы контроля версий git и в написании простейших алгоритмов на python.