

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

Выполнила:

Гайдук А. С.

К3241

Проверила:

Ромакина О. М.

Санкт-Петербург

2024 г.

Содержание отчета

Содержание отчета.....	2
Задачи по варианту	3
Задача №1. Ввод-вывод.....	3
Задача №2. Число Фибоначчи	6
Задача №3. Еще про числа Фибоначчи.....	7
Задача №4. Тестирование ваших алгоритмов.	9
Вывод	11

Задачи по варианту

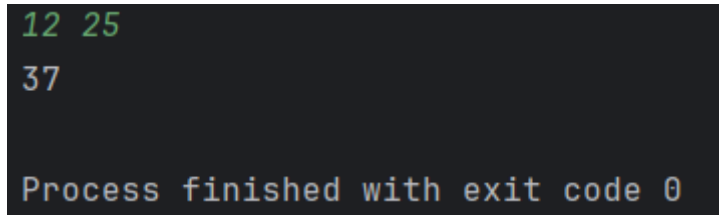
Задача №1. Ввод-вывод

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 $a + b$.

Листинг кода:

```
a, b = map(int, input().split())  
print(a+b)
```

Я воспользовалась методом `map(int,...)` для того, чтобы сразу преобразовать строковые данные в целочисленный тип. Так же я использовала метод `input().split()`, чтобы введенные пользователем значения разделялись (по пробелу) на два числа.



```
12 25  
37  
  
Process finished with exit code 0
```

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 $a + b^2$.

Листинг кода:

```
a, b = map(int, input().split())  
print(a+b**2)
```

В данной задаче я использовала те же методы, что и в №1.1, за исключением того, что возвела в квадрат b .

```
12 25
637

Process finished with exit code 0
```

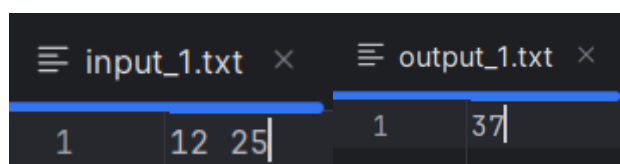
3. Выполните задачу $a + b$ с использованием файлов.

- 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 $a + b$.

Листинг кода:

```
with open('input_1.txt', 'r') as input_f:
    a, b = map(int, input_f.readline().split())
with open('output_1.txt', 'w') as output_f:
    output_f.write(str(a+b))
```

Для решения данной задачи я открыла для чтения файл input_1.txt, созданный мной заранее. Строка с числами, разделенными пробелами, преобразуется в целочисленный тип с помощью метода `map(int, ...)` и сохраняются в переменные a , b . Далее я открыла для записи файл output_1.txt, куда записывается сумма чисел a и b .



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из	0.1405119999995804 сек	0.0390625 Мб

текста задачи		
Пример из задачи	0.20437189999993871	0.06143999999999995 Мб
Пример из задачи	0.204451999999994657 сек	0.024576 Мб
Верхняя граница диапазона значений входных данных из текста задачи	0.20380559999975958 сек	0.057344 Мб

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

Листинг кода:

```
with open('input_2.txt', 'r') as input_f:
    a, b = map(int, input_f.readline().split())
with open('output_2.txt', 'w') as output_f:
    output_f.write(str(a+b**2))
```

Аналогично предыдущей задаче, за исключением возведения в квадрат числа b.

input_2.txt	output_2.txt
1 12 25	1 637

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.20388800000000063 сек	0.024576 Мб

Пример из задачи	0.20396640000035404 сек	0.024576 Мб
Пример из задачи	0.20417350000002518 сек	0.024576 Мб
Верхняя граница диапазона значений входных данных из текста задачи	0.20424549999916053 сек	0.024576 Мб

Вывод по задаче: в ходе решения этих задач я вспомнила методы по работе с файлами.

Задача №2. Число Фибоначчи

Ваша цель – разработать эффективный алгоритм для подсчета чисел Фибоначчи.

- 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 .

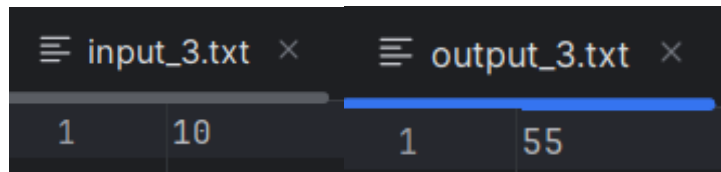
Листинг кода:

```
with open('input_3.txt', 'r') as input_f:
    n = int(input_f.readline())
    a, b = 0, 1
    i = 0
    while i < n:
        a, b = b, a + b
        i += 1

with open('output_3.txt', 'w') as output_f:
    output_f.write(str(a))
```

Для решения данной задачи я открыла файл input_3.txt для чтения, считала число n – позицию числа Фибоначчи. Далее я инициализировала

переменные a, b значениями для первых двух чисел Фибоначчи – 0 и 1. В ходе цикла переменная a получает предыдущее значение переменной b, a b = сумму предыдущих значений a и b. Далее я открыла файл output_3.txt для записи и записала туда результат.



Вывод по задаче: в ходе решения этой задачи я вспомнила, как работать с циклами.

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.035223999993831967 сек	0.0296875 MiB
Пример из задачи	0.10272710000026564 сек	0.021365 Мб
Верхняя граница диапазона значений входных данных из текста задачи	0.10423399999854155 сек	0.024576 Мб

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 .
- Ограничение по времени: 5сек.
- Ограничение по памяти: 512 мб.

Листинг кода:

```
import time
from memory_profiler import memory_usage
memory_usage_before = memory_usage()[0]
t_start = time.perf_counter()
with open('input_4.txt', 'r') as input_f:
    n = int(input_f.readline())
a, b = 0, 1
if n == 0:
    last_digit = a
else:
    for i in range(2, n + 1):
        a, b = b % 10, (a + b) % 10
    last_digit = b
with open('output_4.txt', 'w') as output_f:
    output_f.write(str(last_digit))
```

Здесь мне пришлось сильно оптимизировать код из задачи №2 для удовлетворения всего диапазона возможных значений n . Я свела все вычисления только к последней цифре числа Фибоначчи на каждом этапе, чтобы избежать лишних вычислений с большими числами.

input_4.txt	output_4.txt
1 327305	1 5

input_4.txt	output_4.txt
1 331	1 9

Вывод по задаче: в ходе решения этой задачи мне пришлось изменить написанный в задаче №2 код, используя арифметическую операцию остатка деления (%).

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.10370139999940875 сек	0.0390625 Мб
Пример из задачи	0.10388120000061463 сек	0.028672 Мб
Пример из задачи	0.1485224999996717 сек	0.0546875 Мб
Верхняя граница диапазона значений входных данных из текста задачи	1.3561597000007168 сек	0.04095999999999996 Мб

Задача №4. Тестирование ваших алгоритмов.

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

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

Для решения этой задачи я импортировала модуль `time` для измерения времени выполнения программы, а также модуль `memory_profiler` – для измерения объема используемой памяти. Рассмотрим на примере второй задачи:

```
import time
from memory_profiler import memory_usage
memory_usage_before = memory_usage()[0]
t_start = time.perf_counter()
with open('input_3.txt', 'r') as input_f:
```

```

n = int(input_f.readline())
a, b = 0, 1
i = 0
while i < n:
    a, b = b, a + b
    i += 1

with open('output_3.txt', 'w') as output_f:
    output_f.write(str(a))
print(f"Время выполнения {time.perf_counter() -
t_start} секунд")
print(f"Объем использованной памяти
{memory_usage()[0] - memory_usage_before} MiB")

```

Для каждой программы я определила начало замера времени и памяти (`t_start = time.perf_counter()` и `memory_usage_before = memory_usage()[0]`). После блока основного кода я замерила время и память, а затем вывела эти значения:

```

Время выполнения 0.10287459999926796 секунд
Объем использованной памяти 0.024576 M6

```

Аналогичные действия я провела для третьей задачи. Вывод программы:

```

Время выполнения 0.10315170000103535 секунд
Объем использованной памяти 0.028672 M6

```

Вывод по задаче: я ознакомилась с принципом работы с модулями `time` и `memory_profiler`, и протестировала время выполнения моих алгоритмов и объем используемой памяти при их выполнении.

Вывод

В ходе данной лабораторной работы я выполнила задачи по работе с числами и последовательностью Фибоначчи, закрепила свои навыки работы с файлами для ввода и вывода данных. Я научилась работать с модулями и библиотеками для вычисления времени и памяти, а также изучила принципы построения эффективных алгоритмов.