

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. Работа с файлами, тестирование

Выполнила:
Беляева В.А.

Проверил:

Санкт-Петербург

2024 г.

Содержание отчета

Содержание отчета	2
Задачи по варианту	3
Задача №1. Ввод-вывод	3
Задача №2. Число Фибоначчи	7
Задача №3. Еще про числа Фибоначчи	9
Задача №4. Тестирование ваших алгоритмов	10
Вывод	12

Задачи по варианту

Задача №1

Пункт. 1

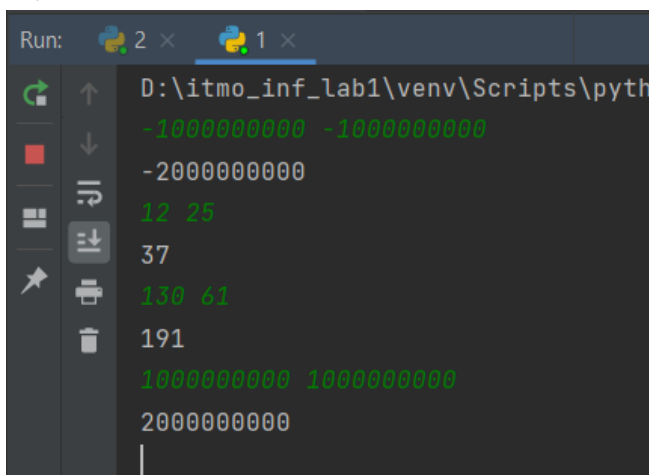
В данной задаче требуется вычислить сумму двух заданных чисел. Вход: одна строка, которая содержит два целых числа a и b . Для этих чисел выполняются условия $-10^9 \leq a, b \leq 10^9$. Выход: единственное целое число — результат сложения $a + b$.

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

```
print(a+b)
```

-После считывания строка расбивается по пробелам; при помощи `map` элементы преобразуются в числа и записываются в a и b .

-сумма выводится через `print`.



```
Run: 2 x 1 x
D:\itmo_inf_lab1\venv\Scripts\python.exe
-1000000000 -1000000000
-2000000000
12 25
37
130 61
191
1000000000 1000000000
2000000000
```

Тест	Время выполнения, сек	Затраты памяти, bytes
-1e9 -1e9	0,0000369	32
12 25	0,0000302	28
130 61	0,0000336	28
1e9 1e9	0,0000338	32

Пункт 2

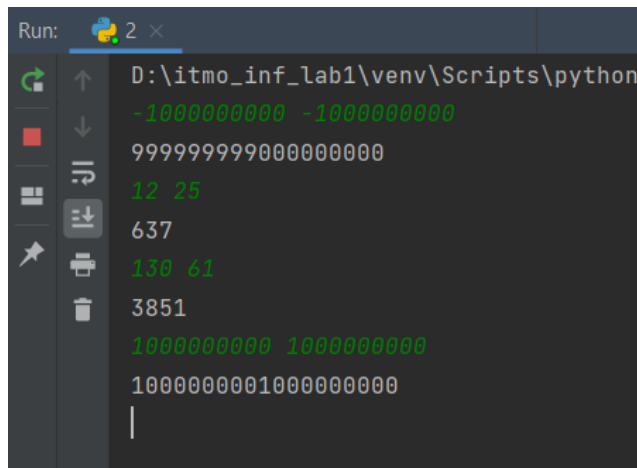
В данной задаче требуется вычислить значение $a + b^2$. Вход: одна строка, которая содержит два целых числа a и b . Для этих чисел

выполняются условия $-10^9 \leq a, b \leq 10^9$. Выход: единственное целое число — результат сложения $a + b^2$.

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

-После считывания строка расбивается по пробелам; при помощи `map` элементы преобразуются в числа и записываются в `a` и `b`.

-сумма `a` и квадрата `b` выводится через `print`.



```
Run: 2 x  
D:\itmo_inf_lab1\venv\Scripts\python  
-1000000000 -1000000000  
3851  
999999999 0000000000  
1000000000  
12 25  
637  
130 61  
13061  
1000000000 1000000000  
10000000001000000000  
|
```

Тест	Время выполнения,сек	Затраты памяти,bytes
-1e9 -1e9	0,0000420	32
12 25	0,0000337	28
130 61	0,0000345	28
1e9 1e9	0,0000362	32

Пункт 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 $a + b$.

```
f = open("input")
```

```

a, b = map(int, f.readline().split())
f.close()
f = open("output", 'w')
f.write(str(a+b))
f.close()

```

- Открывается первая файл ввода
- через readline запрашивается вводная строка, как в п.1 данные преобразуются к двум числам.
- закрытие вводного файла.
- открытие файла на вывод.
- запись в файл для вывода суммы а и b, преобразованной в строку.
- закрытие файла вывода.

input ×		output ×	
1	12 25	1	37
2	130 61	2	191
3	-1000000000 -1000000000	3	-2000000000
4	1000000000 1000000000	4	2000000000

Тест	Время выполнения,сек	Затраты памяти,bytes
-1e9 -1e9	0.000439	32
15 25	0.001038	28
130 61	0.000731	28
1e9 1e9	0.000376	32

Пункт 4

Выполните задачу а+b 2 с использованием файлов аналогично предыдущему пункту.

```

f = open("input")
a, b = map(int, f.readline().split())
f.close()
f = open("output", 'w')
f.write(str(a+b**2))
f.close().

```

- Открывается первая файл ввода
- через readline запрашивается вводная строка, как в п.1 данные преобразуются к двум числам.
- закрытие вводного файла.
- открытие файла на вывод.
- запись в файл для вывода суммы a и квадрата b, преобразованной в строчку.
- закрытие файла вывода.

input ×		output ×	
1	12 25	1	637
2	130 61	2	3851
3	-1000000000 -1000000000	3	999999999000000000
4	1000000000 1000000000	4	1000000001000000000

Тест	Время выполнения,сек	Затраты памяти,bytes
-1e9 -1e9	0.001272	32
12 25	0.000378	28
130 61	0.000731	28
1e9 1e9	0.0003762	32

Вывод по задаче: При увеличении числа (по модулю) для его хранения выделяется бóльшее кол-во байтов памяти

Задача №2 Число Фибоначчи

Определение последовательности Фибоначчи: $F_0 = 0$ (1) $F_1 = 1$ $F_i = F_{i-1} + F_{i-2}$ 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:

```
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 .

```
fl = open("input")  
n = int(fl.readline())  
fl.close()
```

```
fl = open("output", 'w')  
fib = [0 for i in range(n+1)]  
fib[1] = 1  
for i in range(2, n+2):  
    fib[i] = fib[i-1]+fib[i-2]
```

```
fl.write(str(fib[n]))  
fl.close()
```

- Открывается первая файл ввода
- Полученная строка переводится в число и записывается в n
- файл закрывается
- Открывается файл для вывода
- Создается лист `fib` для записи чисел фибоначчи

- первому эл-ту присваивается 1
- Запускается цикл на $n+2$ (для учета крайних границ)
- нужный эл-т записывается в файл
- После файл закрывается.

input		output	
1	0	1	0
2	10	2	55
3	45	3	1134903170

Тест	Время выполнения, сек	Затраты памяти, bytes
0	0.001878	56
10	0.001580	106
45	0.000932	274

Вывод по задаче: Использование рекурсии не рационально по времени

Задача №3 Еще про числа Фибоначчи

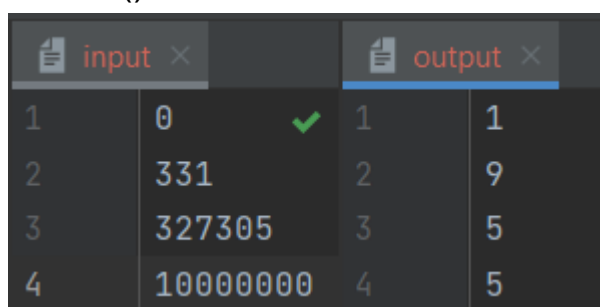
Определение последней цифры большого числа Фибоначчи. Числа Фибоначчи растут экспоненциально. Например, $F_{200} = 280571172992510140037611932413038677189525$. Хранить такие суммы в массиве, и при этом подсчитывать сумму, будет достаточно долго. Найти последнюю цифру любого числа достаточно просто: $F \bmod 10$.

- 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.

```
fl = open("input")
n = int(fl.readline())
fl.close()

fl = open("output", 'w')
a, b = 0, 1
for i in range(2, n+1):
    a, b = b, (a+b)%100
```

```
fl.write(str(b%10))
fl.close()
```



input	output
1 0	1 1
2 331	2 9
3 327305	3 5
4 10000000	4 5

Тест	Время выполнения, сек	Затраты памяти, bytes
0	0.0000198	26
331	0.0000607	28
327305	0.0562672	28

10000000	1.5855918	28
----------	-----------	----

Вывод по задаче: Использование рекуррентного соотношения для пары чисел, значительно оптимальнее по времени, чем динамика на списке

Задача №4. Тестирование ваших алгоритмов.

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

К заданию 2:

```
import time
```

```
t_start = time.perf_counter()
```

```
fl = open("input")
```

```
n = int(fl.readline())
```

```
fl.close()
```

```
fl = open("output", 'w')
```

```
fib = [0 for i in range(n+1)]
```

```
fib[1] = 1
```

```
for i in range(2, n+1):
```

```
    fib[i] = fib[i-1]+fib[i-2]
```

```
fl.write(str(fib[n]))
```

```
fl.close()
```

```
print(f"ремя работы кода:{time.perf_counter() - t_start}")
```

Код аналогичен коду к заданию 2, но

- Строка 1 – подключение модуля
- Строка 2 – засекается время начала кода
- Строка 14 – вывод разницы текущего и засечного времени
- Строка 15 – закрытие файла для ввода

Тест	Время выполнения, сек	Затраты памяти,bytes
0	0.0000505	72
10	0.0000169	122
45	0,0000261	290

```

import time
t_start = time.perf_counter()
fl = open("input")
n = int(fl.readline())
fl.close()

fl = open("output", 'w')
a, b = 0, 1
for i in range(2, n+1):
    a, b = b, (a+b)%100

fl.write(str(b%10))
fl.close()
print(f"ремя работы кода:{time.perf_counter() - t_start}")

```

- Строка 1 – подключение модуля
- Строка 2 – засекается время начала кода
- Строка 12 – вывод разницы текущего и засечного времени
- Строка 13 – закрытие файла для ввода

Тест	Время выполнения, сек	Затраты памяти,bytes
0	0.0000491	42
331	0.0000662	44
327305	0.0582838	44
10000000	1.658069	44

Вывод: на то, чтобы засечь время требуется незначительное кол-во времени

Вывод по лабораторной:

Python делает приблизительно $1.5e6$ операций в секунду. Хранить большой объем данные в списке экономнее по памяти чем по отдельности.