

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:
Мкртчян К.Г.
К3141

Проверил:
Афанасьев А.В.

Санкт-Петербург
2024 г.

Содержание отчета

Содержание отчета	2
Задачи по варианту.....	3
Задание №1. Ввод-вывод.....	3
Задача №1. Задача $a + b$	3
Задача №2. Задача $a + b^2$	4
Задача №3.....	5
Задача №4.....	6
Задание №2. Число Фибоначчи.	7
Задание №3. Ещё про числа Фибоначчи.	8
Вывод по лабораторной работе:.....	10

Задачи по варианту

Задание №1. Ввод-вывод.

Задача №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 $a + b$.

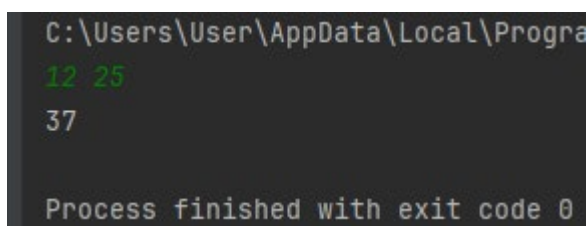
```
def sum_ab(s, n):
    if n == 2:
        print("Числа не удовлетворяют ограничению!")
    else:
        a, b = map(int, list(s.split()))
        if -10 ** 9 <= a <= 10 ** 9 and -10 ** 9 <= b <= 10 ** 9:
            return a + b
        else:
            print("Введите числа ещё раз: ")
            sum_ab(input(), n+1)

print(sum_ab(input(), 0))
```

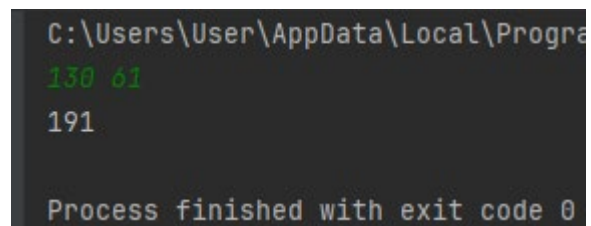
Эта программа выполняет следующие действия:

- 1) Функция разделяет полученную строку на две строки через пробел. Создает список из этих строк с помощью встроенной функции `list()` и преобразует эти строки в числа с помощью встроенной функции `map()`, записывая каждый элемент списка в отдельную переменную.
- 2) Проверяет полученные числа с помощью условия.
- 3) Если числа удовлетворяют этому ограничению, то программа возвращает сумму этих чисел.
- 4) С помощью функции `input()` мы считываем строку, введенную пользователем, затем передаем её в функцию `sum_ab()` и с помощью функции `print()` выводим результат работы функции в консоль.
- 5) Если числа ограничению не удовлетворяют, то мы ещё два раза просим пользователя ввести корректные значения. Если на третий раз значения некорректные, то выводим сообщение: "Числа не удовлетворяют ограничению!"

Проверим работу программы на заданных тестах:



```
C:\Users\User\AppData\Local\Progra
12 25
37
Process finished with exit code 0
```



```
C:\Users\User\AppData\Local\Progra
130 61
191
Process finished with exit code 0
```

Эта задача не требует проверки времени выполнения и затрат памяти.

Вывод по задаче:

Мы вспомнили, как писать простые программы на Python в виде функций, как считывать и проверять полученные данные в условии, как возвращать данные через функцию, а затем выводить их в консоль.

Задача №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 $a + b^2$.

```
def sum_ab2(s, n):
    if n == 2:
        print("Числа не удовлетворяют ограничению!")
    else:
        a, b = map(int, list(s.split()))
        if -10 ** 9 <= a <= 10 ** 9 and -10 ** 9 <= b <= 10 ** 9:
            return a + b ** 2
        else:
            print("Введите числа ещё раз: ")
            sum_ab2(input(), n + 1)

print(sum_ab2(input(), 0))
```

Эта программа выполняет следующие действия:

- 1) Функция, аналогично предыдущей задаче, считывает и записывает два числа в переменные **a**, **b**.
- 2) Проверяет полученные числа с помощью условия.
- 3) Если числа удовлетворяют этому ограничению, то программа возводит **b** в квадрат, прибавляет **a** и возвращает эту сумму.
- 4) С помощью функции `input()` мы считываем строку, введенную пользователем, затем передаем её в функцию `sum_ab2()` и с помощью функции `print()` выводим результат работы функции в консоль.
- 5) Если числа ограничению не удовлетворяют, то мы ещё два раза просим пользователя ввести корректные значения. Если на третий раз значения некорректные, то выводим сообщение: "Числа не удовлетворяют ограничению!"

Проверим работу программы на заданных тестах:

<pre>C:\Users\User\AppData\Local\Programs\Python\Python38-64\python.exe C:\Users\User\AppData\Local\Programs\Python\Python38-64\python.exe 12 25 637 Process finished with exit code 0</pre>	<pre>C:\Users\User\AppData\Local\Programs\Python\Python38-64\python.exe C:\Users\User\AppData\Local\Programs\Python\Python38-64\python.exe 130 61 3851 Process finished with exit code 0</pre>
--	--

Эта задача не требует проверки времени выполнения и затрат памяти.

Вывод по задаче:

Помимо действий в прошлой задаче здесь мы вспомнили, как в Python записывается возведение числа в степень и применили эту функцию к одной из переменных.

Задача №3.

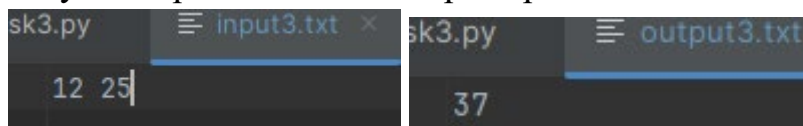
Выполнить задачу №1, используя ввод-вывод через файлы.

```
def sum_ab_file(path1, path2):  
    file = open(path1)  
    s = file.readline()  
    a, b = map(int, list(s.split()))  
    if -10 ** 9 <= a <= 10 ** 9 and -10 ** 9 <= b <= 10 ** 9:  
        open(path2, "w").write(str(a + b))  
  
sum_ab_file("input3.txt", "output3.txt")
```

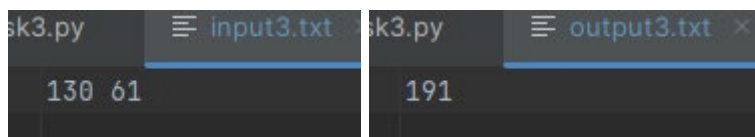
Программа выполняет следующие действия:

- 1) Мы вызываем функцию и передаем ей два параметра: путь к файлу input3.txt, из которого будет производиться считывание данных, и путь к файлу output3.txt, куда будет записан результат.
- 2) В функции мы открываем первый файл и считываем из него строку.
- 3) Записываем два числа из строки в переменные аналогично двум предыдущим заданиям.
- 4) Проверяем переменные.
- 5) Если они удовлетворяют заданным ограничениям, то функция открывает второй файл и записывает в него значение суммы двух чисел в строковом формате.

Результат работы кода на примерах из текста задачи:



input3.txt: 12 25
output3.txt: 37



input3.txt: 130 61
output3.txt: 191

Эта задача не требует проверки времени выполнения и затрат памяти.

Вывод по задаче:

Помимо действий в первой задаче, мы вспомнили, как на языке программирования Python нужно считывать данные из файла и записывать результат работы функции в другой файл.

Задача №4.

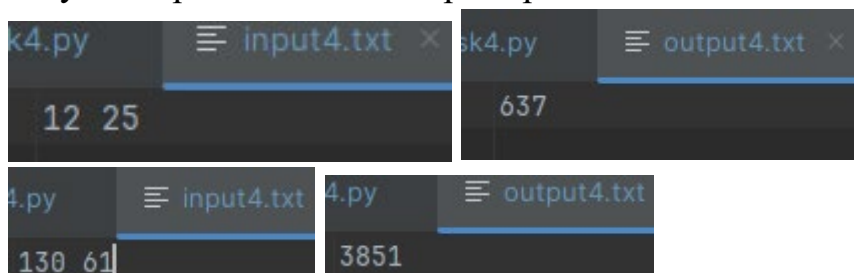
Выполнить задачу №2, используя ввод-вывод через файлы.

```
def sum_ab2_file(path1, path2):  
    file = open(path1)  
    s = file.readline()  
    a, b = map(int, list(s.split()))  
    if -10 ** 9 <= a <= 10 ** 9 and -10 ** 9 <= b <= 10 ** 9:  
        open(path2, "w").write(str(a + b ** 2))  
  
sum_ab2_file("input4.txt", "output4.txt")
```

Программа выполняет следующие действия:

По аналогии с предыдущей программой, данная считывает строку из файла input4.txt, преобразует её и записывает в output4.txt результат сложения числа **a** и квадрата числа **b**.

Результат работы кода на примерах:



Эта задача не требует проверки времени выполнения и затрат памяти.

Вывод по задаче: мы закрепили материал из задачи №2 и задачи №3.

Задание №2. Число Фибоначчи.

Разработать эффективный алгоритм для подсчёта n-ого числа фибоначчи.

```
import psutil
import time

t_start = time.perf_counter()

def fib_number(path1, path2):
    file = open(path1)
    n = int(file.readline())
    if 0 <= n <= 45:
        a, b = 0, 1
        for i in range(n):
            a, b = b, a + b
        open(path2, "w").write(str(a))

fib_number("input.txt", "output.txt")
print("Время работы: %s секунд" % (time.perf_counter() - t_start))
print(f"Память: {psutil.Process().memory_info().rss / 1024 ** 2:.2f} МБ")
```

Функция выполняет следующие действия:

- 1) Открывает файл input.txt и считывает первую строку из него.
- 2) Преобразует эту строку в число, записывая его в переменную **n**.
- 3) Если **n** удовлетворяет заданному ограничению, то мы инициализируем переменные **a** и **b**. Присваиваем им значения 0 и 1 соответственно.
- 4) В цикле n-раз изменяем переменные: переменной **a** присваиваем значение переменной **b**, а **b** – **a** (предыдущее знач.) + **b**. В Python это можно записать как **a, b = b, a + b**. В других ЯП придётся использовать дополнительную переменную для сохранения значения **a**.
- 5) После окончания цикла необходимое значение – значение переменной **a** записываем в файл output.txt.

С помощью модуля time и функции perf_counter () мы записываем начальное время в переменную **t_start**. После завершения работы функции мы ещё раз находим время и вычитаем из этого значения **t_start**. Эта разность и будет временем выполнения программы.

С помощью модуля psutil измерим необходимый программе объём памяти.

Входные данные	Выходные данные	Время работы в сек.	Затраты памяти в Мб.
0	0	0.0018	15.35
10	55	0.0016	15.37
45	1134903170	0.00106	15.10

Вывод по задаче:

Сложность этого алгоритма $O(n)$. Итеративный метод — это самый простой способ вычисления чисел Фибоначчи. Он основан на поочередном вычислении каждого числа в последовательности на основе двух предыдущих чисел. Реализация с помощью рекурсии повторяет математическое определение числа Фибоначчи, но такой метод неэффективен при больших значениях n , потому что из-за множества вызовов функции происходит переполнение стека.

Самый эффективный, но сложный в реализации метод вычисления числа Фибоначчи – вычисление через умножение матриц. Его сложность $O(\log(n))$. При $n = 45$ он выполняется за 0.00089 секунд и требует 15.23 МБ памяти.

```
import psutil
import time

t_start = time.perf_counter()

K = [[0, 1], [1, 1]]

def fast_multiply(x, n):
    if n == 0:
        return [[1, 0], [0, 1]]
    elif n % 2 == 0:
        y = fast_multiply(x, n / 2)
        return matrix_multiply_2x2(y, y)
    else:
        y = fast_multiply(x, (n - 1) / 2)
        y2 = matrix_multiply_2x2(y, y)
        return matrix_multiply_2x2(y2, x)

def matrix_multiply_2x2(A, B):
    C = [[0, 0], [0, 0]]
    C[0][0] = (A[0][0] * B[0][0] + A[0][1] * B[1][0])
    C[1][0] = (A[1][0] * B[0][0] + A[1][1] * B[1][0])
    C[0][1] = (A[0][0] * B[0][1] + A[0][1] * B[1][1])
    C[1][1] = (A[1][0] * B[0][1] + A[1][1] * B[1][1])
    return C

open("output.txt", "w").write(str(fast_multiply(K,
int(open("input.txt").readline())[0][1])))
print("Время работы: %s секунд" % (time.perf_counter() - t_start))
print(f"Память: {psutil.Process().memory_info().rss / 1024 ** 2:.2f} МБ")
```

Задание №3. Ещё про числа Фибоначчи.

Определить последнюю цифру большого числа Фибоначчи.

```
import time
import psutil

t_start = time.perf_counter()

def last_digit(path1, path2):
```



```

file = open(path1)
n = int(file.readline())
if 0 <= n <= 10 ** 7:
    a, b = 0, 1
    for i in range(n):
        a, b = b % 10, (a + b) % 10
    open(path2, "w").write(str(a))

last_digit("input.txt", "output.txt")
print("Время работы: %s секунд" % (time.perf_counter() - t_start))
print(f"Память: {psutil.Process().memory_info().rss / 1024 ** 2:.2f} МБ")

```

Функция выполняет следующие действия:

- 1) Она аналогично прошлой функции считывает данные из файла, проверяет, удовлетворяют ли они условию.
- 2) Если удовлетворяют, то мы инициализируем переменные **a = 0** и **b = 1**.
- 3) Затем n-раз изменяем их в цикле: в **b** записываем остаток от деления на 10 (разряд единиц) суммы **a** и **b**, а в **a** – прошлое значение **b**.
- 4) Записываем результат в файл output.txt.

Как и в предыдущей задаче мы вычисляем время работы программы и необходимый ей объем памяти.

Входные данные	Выходные данные	Время работы в сек.	Затраты памяти в Мб.
0	0	0.0016	15.26
331	9	0.0016	15.30
327305	5	0.021	15.36
10**7	5	0.509	15.28

Вывод по задаче: этот алгоритм работает с той же сложностью, что и предыдущий, но для подсчёта последней цифры нам достаточно запоминать остатки от деления на 10 (разряд единиц), а не сами числа.

Предложу решение этой задачи с помощью возведения матрицы в степень n:

```

import psutil
import time

t_start = time.perf_counter()

K = [[0, 1], [1, 1]]

def fast_multiply(x, n, m):
    if n == 0:
        return [[1, 0], [0, 1]]
    elif n % 2 == 0:
        y = fast_multiply(x, n / 2, m)
        return matrix_multiply_2x2(y, y, m)
    else:
        y = fast_multiply(x, (n - 1) / 2, m)
        y2 = matrix_multiply_2x2(y, y, m)

```

```

        return matrix_multiply_2x2(y2, x, m)

def matrix_multiply_2x2(A, B, m):
    C = [[0, 0], [0, 0]]
    C[0][0] = (A[0][0] * B[0][0] + A[0][1] * B[1][0]) % m
    C[1][0] = (A[1][0] * B[0][0] + A[1][1] * B[1][0]) % m
    C[0][1] = (A[0][0] * B[0][1] + A[0][1] * B[1][1]) % m
    C[1][1] = (A[1][0] * B[0][1] + A[1][1] * B[1][1]) % m
    return C

open("output.txt", "w").write(str(fast_multiply(K,
int(open("input.txt").readline()), m=10)[0][1]))
print("Время работы: %s секунд" % (time.perf_counter() - t_start))
print(f"Память: {psutil.Process().memory_info().rss / 1024 ** 2:.2f} МБ")

```

Объем памяти останется неизменным, а время будет 0.0011 сек.

При $n = 10^7$.

Вывод по лабораторной работе:

Мы вспомнили, как писать простые программы, как работать с файлами. Научились писать эффективные алгоритмы и вычислять время их работы и используемую память.