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

Выполнил:

Артемов И.В.

К3141

Проверил:

Афанасьев А.В.

Санкт-Петербург 2024 г.

Содержание отчета

Содержание	отчета				2
Задачи					3
Задача №1	1. Ввод-выво	ОД			3
Задача 15		№ 2.	Чис	ело	Фибоначчи
Задача 19	№ 3.	Ещё	про	числа	Фибоначчи
Задача 23	№ 4.	Тестиро	вание	ваших	алгоритмов
Вывод					
25					

Задачи

Задача №1. Ввод-вывод

Текст задачи №1.

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

Листинг кода.

```
a, b = map(int, input().split())
while True:
    if (-10**9 <= a <= 10**9) and (-10**9 <= b <= 10**9):
        print(a + b)
        break
else:
    print('Не подходит по диапазону')
    a, b = map(int, input().split())
```

Текстовое объяснение решения.

В переменные а и b считываются значения, которые мы вводим и дальше, если значения переменных удовлетворяют условию, а именно $-10^9 \le a$, $b \le 10^9$, то на экран выводится сумма значений переменных а и b.

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

```
130 61
191
Process finished with exit code 0
```

1000000000 1000000000 2000000000

Process finished with exit code 0

- -1000000000 -1000000000
- -2000000000

Process finished with exit code 0

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0002562499976193066 7 секунд	393 байт
Пример из задачи	0.00011920899851247668 секунд	389 байт
Пример из задачи	0.0001620420007384382 2 секунд	391 байт
Верхняя граница диапазона значений входных данных из текста задачи	0.0002208339974458795 секунд	389 байт

Текст задачи №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 $-10^9 \le a$, $b \le 10^9$. Выход: единственное целое число — результат сложения $a+b^2$.

Листинг кода.

```
a, b = map(int, input().split())
while True:
    if (-10**9 <= a <= 10**9) and (-10**9 <= b <= 10**9):
        print(a + b**2)
        break
else:
    print('Не подходит по диапазону')
    a, b = map(int, input().split())
```

Текстовое объяснение решения.

В переменные а и b считываются значения, которые мы вводим и дальше, если значения переменных удовлетворяют условию, а именно $-10^9 \le a$, $b \le 10^9$, то на экран выводится значение выражения $a + b^2$.

```
12 25
637

Process finished with exit code 0
```

```
130 61
3851
Process finished with exit code 0
```

Process finished with exit code 0

-1000000000 -1000000000 99999999000000000

Process finished with exit code 0

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.00022116700347396545 секунд	391 байт
Пример из задачи	0.000363416998879984 секунд	387 байт
Пример из задачи	0.0001446669994038529 7 секунд	391 байт
Верхняя граница диапазона значений входных данных из текста задачи	0.0001945410003827419 секунд	389 байт

Текст задачи №3.

Выполните задачу а + 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 а и b. Для этих чисел выполняются условия $-10^9 \le a$, $b \le 10^9$.
- Формат выходного файла. Выходной файл единственное целое число результат сложения a+b.

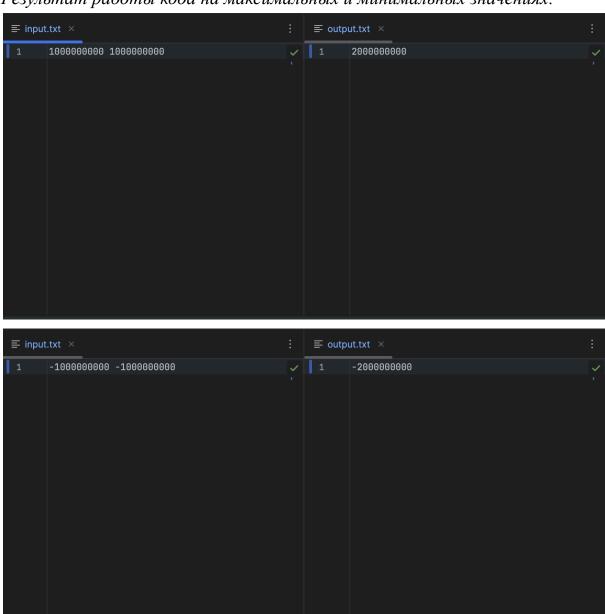
Листинг кода.

```
f = open('input.txt')
a, b = map(int, f.readline().split())
f1 = open('output.txt', 'w')
if (-10**9 <= a <= 10**9) and (-10**9 <= b <= 10**9):
    f1.write(str(a + b))
else:
    print('Не подходит по диапазону, попробуйте ещё раз')
```

Текстовое объяснение решения.

Мы открываем файл input.txt для того, чтобы считать значения для переменных а и b с помощью open(). Далее мы считываем строку, содержащую два числа, с помощью readline() и присваиваем их переменным а и b соответственно. После этого мы открываем файл output.txt для записи результата сложения переменных а и b и, соответственно, записываем результат сложения в данный файл, не забывая о том, что результат сначала нужно преобразовать из int в str, чтобы не получить ошибку, так как аргумент write() должен быть строкового типа данных.





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0004821669972443487 5 секунд	13885 байт
Пример из задачи	0.0003322080010548234 секунд	13849 байт
Пример из задачи	0.0002966669999295845 6 секунд	13851 байт

Верхняя граница	0.0002876250000554137	13881 байт
диапазона значений	секунд	
входных данных из		
текста задачи		

Текст задачи №4.

Выполните задачу $a + b^2 c$ 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 а и b. Для этих чисел выполняются условия $-10^9 \le a$, $b \le 10^9$.
- Формат выходного файла. Выходной файл единственное целое число результат сложения $a + b^2$.

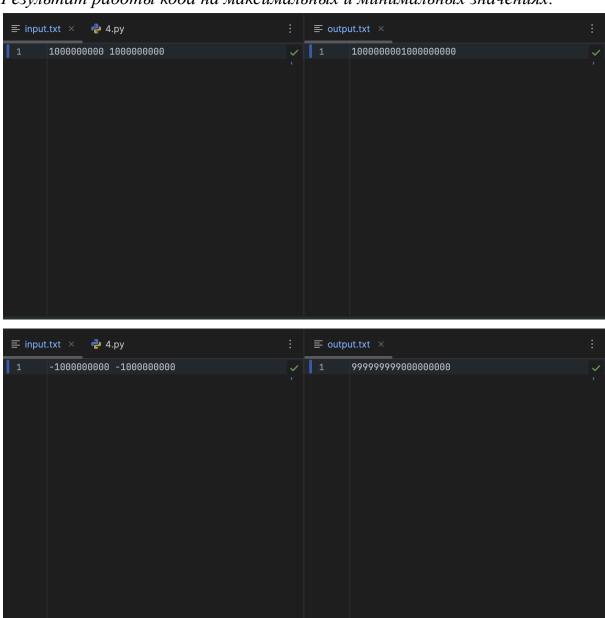
Листинг кода.

```
f = open('input.txt')
a, b = map(int, f.readline().split())
f1 = open('output.txt', 'w')
if (-10**9 <= a <= 10**9) and (-10**9 <= b <= 10**9):
    f1.write(str(a + b**2))
else:
    print('Не подходит по диапазону, попробуйте ещё раз')
```

Текстовое объяснение решения.

Мы открываем файл input.txt для того, чтобы считать значения для переменных а и b с помощью open(). Далее мы считываем строку, содержащую два числа, с помощью readline() и присваиваем их переменным а и b соответственно. После этого мы открываем файл оцриt.txt для записи значения выражения а + b^2 и, соответственно, записываем значения выражения в данный файл, не забывая о том, что значение выражения сначала нужно преобразовать из int в str, чтобы не получить ошибку, так как аргумент write() должен быть строкового типа данных.





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0002965839994431007 7 секунд	13885 байт
Пример из задачи	0.0002761659998213872 3 секунд	13849 байт
Пример из задачи	0.0002792500017676502 5 секунд	13851 байт

Верхняя граница диапазона значений	0.0003230410002288408 6 секунд	13881 байт
входных данных из текста задачи		

Задача №2. Числа Фибоначчи

Текст задачи.

Определение последовательности Фибоначчи:

$$F_0 = 0$$
 (1)
$$F_1 = 1$$

$$F_i = F_{i-1} + F_{i-2}$$
 для $i \ge 2$.

Таким образом, каждое число Фибоначчи представляет собой сумму двух предыдущих, что дает последовательность

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots$$

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))</pre>
```

- Имя входного файла: input.txt
- Имя выходного файла: output.txt
- Формат входного файла. Целое число $n.\ 0 \le n \le 45$.
- Формат выходного файла. Число F_n .
- Пример.

input.txt	10
output.txt	55

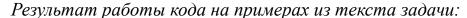
Листинг кода.

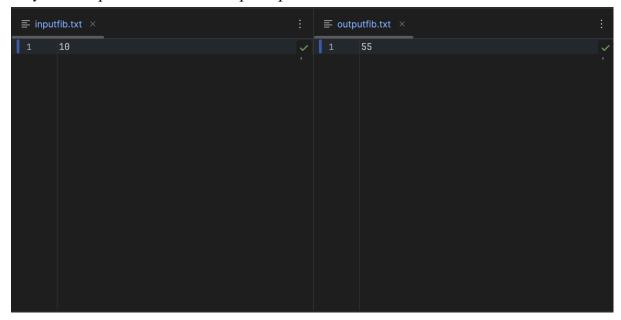
```
f = open('inputfib.txt')
n = int(f.readline())
f1 = open('outputfib.txt', 'w')
if 0 <= n <= 45:
    a, b = 0, 1
    for i in range(2, n + 1):
        a, b = b, a + b
    f1.write(str(b))
else:
    print('Не подходит диапазону, попробуйте ещё раз')</pre>
```

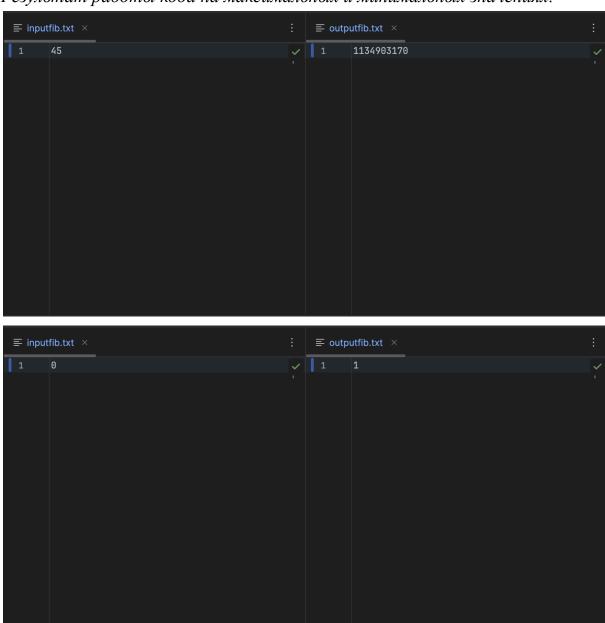
Текстовое объяснение решения.

Сначала мы открываем файл inputfib.txt, который содержит число, которое указывает, какой по счёту элемент последовательности Фибоначчи нужно

вычислить. Строку с этим числом мы считываем из файла и, преобразовав в int, присваиваем переменной п. Далее мы открываем файл outputfib.txt для записи п-го числа Фибоначчи. После этого идёт проверка значение п, на то, входит ли оно в диапазон, заданный условием задачи. Если да, то инициализируются первые два числа последовательности Фибоначчи: 0 и 1, которые присваиваются переменным а и b соответственно. Далее начинается цикл, в котором переменные а и b обновляются: b становится новым числом Фибоначчи, а переменная а принимает предыдущее значение b. Так вычисляется n-ое число последовательности Фибоначчи. После завершения цикла значение b, которое в себе содержит n-ое число Фибоначчи, записывается в файл outputfib.txt.







	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0002760839997790754 секунд	13799 байт
Пример из задачи	0.00032166700111702085 секунд	13843 байт

Верхняя граница	0.0003125000002910383	13843 байт
диапазона значений	секунд	
входных данных из		
текста задачи		

Задача №3. Ещё про числа Фибоначчи

Текст задачи.

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 mod 10.

- Имя входного файла: input.txt
- Имя выходного файла: output.txt
- Формат входного файла. Целое число $n.\ 0 \le n \le 10^7.$
- Формат выходного файла. Одна последняя цифра числа F_n .
- Пример 1.

input.txt	331
output.txt	9

 $F_{331} = 668996615388005031531000081241745415306766517246774551964595292186469.$

• 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.

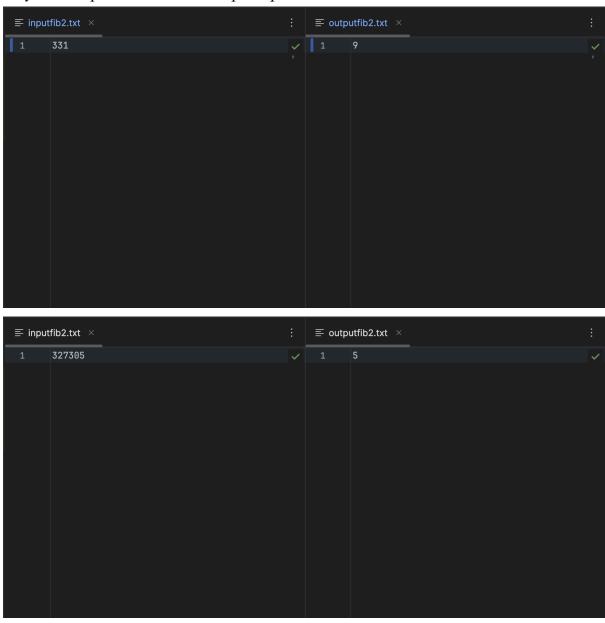
Листинг кода.

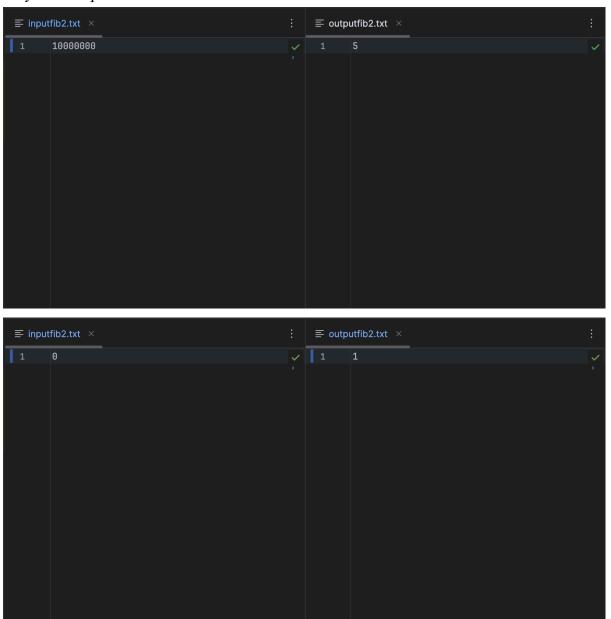
```
f = open('inputfib2.txt')
n = int(f.readline())
f1 = open('outputfib2.txt', 'w')
if 0 <= n <= 10**7:
    a, b = 0, 1
    for i in range(2, n + 1):
        a, b = b % 10, (a + b) % 10
    f1.write(str(b))
else:
    print('Не подходит диапазону, попробуйте ещё раз')</pre>
```

Текстовое объяснение решения.

Сначала мы открываем файл inputfib2.txt, который содержит число, которое указывает, какой по счёту элемент последовательности Фибоначчи нужно вычислить. Строку с этим числом мы считываем из файла и, преобразовав в int, присваиваем переменной п. Далее мы открываем файл outputfib2.txt для записи n-го числа Фибоначчи. После этого идёт проверка значение n, на то, входит ли оно в диапазон, заданный условием задачи. Если да, то инициализируются первые два числа последовательности

Фибоначчи: 0 и 1, которые присваиваются переменным а и b соответственно. Далее начинается цикл, в котором переменные а и b обновляются: b становится последней цифрой нового числа Фибоначчи, а переменная а принимает предыдущее значение b. После завершения цикла значение b, которое в себе содержит последнюю цифру n-ого числа Фибоначчи, записывается в файл outputfib2.txt.





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0003124590002698824 секунд	13799 байт
Пример из задачи	0.0004152080000494607 секунд	13845 байт

Пример из задачи	0.2578129999965313 секунд	13851 байт
Верхняя граница диапазона значений входных данных из текста задачи	0.4740221637557379 секунд	13851 байт

Вывод по задаче: при увеличении значения переменной п время для того, чтобы найти последнюю цифру n-го числа последовательности Фибоначчи, увеличивается. Затраты памяти остаются примерно те же.

Задача №4. Тестирование ваших алгоритмов

Текст задачи.

Задача: вам необходимо протестировать время выполнения вашего алгоритма в Задании 2 и Задании 3. Дополнительно: вы можете протестировать объем используемой памяти при выполнении вашего алгоритма.

Листинг кода.

Код теста задания №2

```
import time
import tracemalloc
t_start = time.perf_counter()
tracemalloc.start()
f = open('inputfibtest.txt')
n = int(f.readline())
f1 = open('outputfibtest.txt', 'w')
if 0 <= n <= 45:
    a, b = 0, 1
    for i in range(2, n + 1):
        a, b = b, a + b
    f1.write(str(b))
else:
    print('He подходит по диапазону, попробуйте ещё раз')
print("Время работы: %s секунд" % (time.perf_counter() - t_start))
print("Затрачено памяти:", tracemalloc.get_traced_memory()[1], "байт")
tracemalloc.stop()</pre>
```

Код теста задания №3

```
import tracemalloc
import time
t_start = time.perf_counter()
tracemalloc.start()
f = open('inputfibtest.txt')
n = int(f.readline())
f1 = open('outputfibtest.txt', 'w')
if 0 <= n <= 10**7:
    a, b = 0, 1
    for i in range(2, n + 1):
        a, b = b % 10, (a + b) % 10
    f1.write(str(b))
else:
    print('He подходит по диапазону, попробуйте ещё раз')
print("Время работы: %s секунд" % (time.perf_counter() - t_start))
print("Затрачено памяти:", tracemalloc.get_traced_memory()[1], "байт")
tracemalloc.stop()</pre>
```

Текстовое объяснение решения.

Тестирование происходит с помощью библиотек time (для измерения времени) и tracemalloc (для измерения затрат памяти)

Результат работы кода на примерах из текста задачи:

Время работы: 0.001199792000988964 секунд

Затрачено памяти: 13845 байт

Process finished with exit code 0

Время работы: 0.24444783400394954 секунд

Затрачено памяти: 13851 байт

Process finished with exit code 0

Вывод по задаче: при увеличении значения переменной п время для того, чтобы найти последнюю цифру n-го числа последовательности Фибоначчи или n-ое число последовательности Фибоначчи, увеличивается. Затраты памяти остаются примерно те же.

Вывод:

Лабораторная работа №0 позволяет вспомнить алгоритм нахождения n-го числа последовательности Фибоначчи, а также позволяет вспомнить, как работать с текстовыми файлами и как замерять время выполнения программы и затраты памяти.