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

Вариант 11

Выполнил:
Тимаков Е.П. (фамилия имя)
К3141 (номер группы)

Проверила:
Артамонова В.Е.

Санкт-Петербург 2024 г.

Содержание отчета

Задачи по варианту

Задача №1. Ввод-вывод

Вам необходимо выполнить 4 следующих задачи:

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 $-109 \le a, b \le 109$. Выход: единственное целое число — результат сложения a + b. 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 $-109 \le a, b \le 109$. Выход: единственное целое число — результат сложения a + b 2 .

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 $-109 \le a, b \le 109$.

• Формат выходного файла. Выходной файл единственное целое число — результат сложения a + b.

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

Листинг кода задачи 3:

```
In = open("input.txt","r")
a, b = (int(x) for x in In.read().split())
In.close()
Out = open("output.txt","w")
Out.write(str(a+b))
```

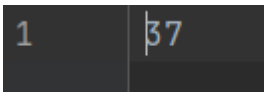
Текстовое объяснение решения:

В данном коде, мы просто считываем значения из input.txt, затем сразу выводим результат в output.txt

Результат работы кода на примерах из текста задачи:

input.txt:

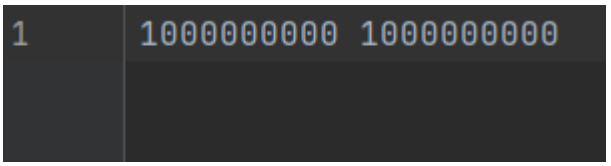
output.txt:



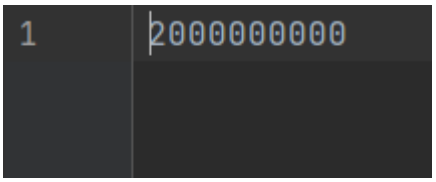
input.txt:

output.txt:

Результат работы кода на максимальных:
input.txt:

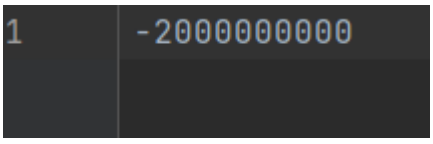


output.txt:



Результат работы кода на минимальных значениях:
input.txt:

output.txt:

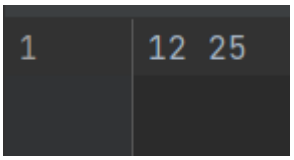


Листинг кода задачи 4:

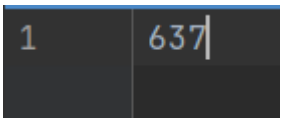
```
In = open("input.txt","r")
a, b = (int(x) for x in In.read().split())
In.close()
Out = open("output.txt","w")
Out.write(str(a+b**2))
```

Текстовое объяснение решения:

В данном коде, все аналогично предыдущему, за исключением операции **, которая возводит число в нужную степень.
Результат работы кода на примерах из текста задачи:
input.txt:

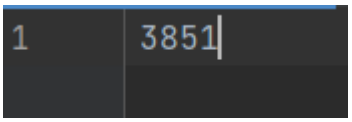


output.txt:

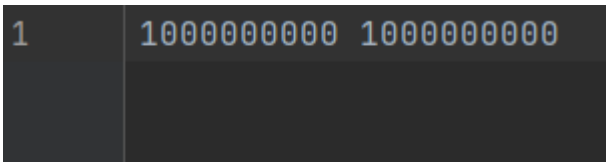


input.txt:

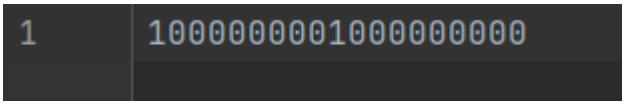
output.txt:



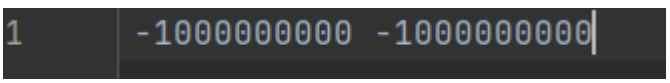
Результат работы кода на максимальных:
input.txt:



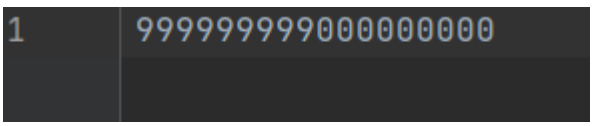
output.txt:



Результат работы кода на минимальных значениях:
input.txt:



output.txt:



Вывод по задаче:

В данной задаче мы изучили работу с файлами.

Задача №2. Число Фибоначчи

Определение последовательности Фибоначчи:
F0 = 0 (1)
F1 = 1
Fi = Fi−1 + Fi−2 для i ≥ 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.

- Имя входного файла: input.txt
- Имя выходного файла: output.txt
- Формат входного файла. Целое число n. 0 ≤ n ≤ 45.
- Формат выходного файла. Число Fn

Листинг кода :

```
import time
from memory_profiler import memory_usage

S_time= time.perf_counter

def fibonacci(i, n1, n2):
    if i<1:
        return n2
    return fibonacci(i-1, n2, n1 + n2)
In = open("input.txt", "r")
i = int(In.read())
In.close()
Out = open("output.txt", "w")
if i == 1:
    Out.write("0")
elif i == 2:
    Out.write("1")
else:
    Out.write(str(fibonacci(i, 0, 1)))
print(time.perf_counter() - S_time)
```

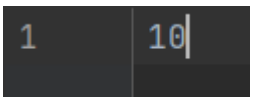
```
print(memory_usage())
```

Текстовое объяснение решения:

В данном коде мы считаем числа фибоначчи. Для вычисления чисел фибоначчи мы используем рекурсию, где передаем аргументы i - счетчик, n1 - предыдущее число, n2 - текущее.

Результат работы кода на примерах из текста задачи:

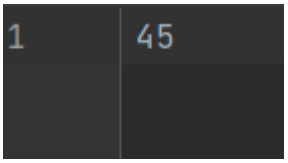
input.txt:



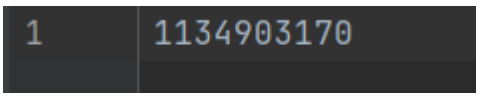
output.txt:

Результат работы кода на максимальных:

input.txt:

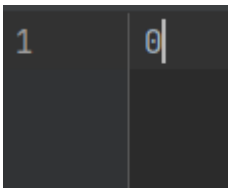


output.txt:



Результат работы кода на минимальных значениях:

input.txt:



output.txt:

	Время выполнения	Затраты памяти

Нижняя граница диапазона значений входных данных из текста задачи	0.002	25
Пример из задачи	0.002	25
Верхняя граница диапазона значений входных данных из текста задачи	0.002	25

Вывод по задаче:

В данной задаче мы применили рекурсию для решения задачи.

Задача №3. Еще про числа Фибоначчи

Определение последней цифры большого числа Фибоначчи. Числа Фибоначчи растут экспоненциально. Например, F200 = 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. $0 \leq n \leq 107$.
- Формат выходного файла. Одна последняя цифра числа Fn.

Листинг кода :

```
import time
from memory_profiler import memory_usage

t_start = time.perf_counter();

def last_digit_fibonacci(n):
    if n <= 1:
        return n
    pisano_period = 60
    n %= pisano_period

    if n <= 1:
```

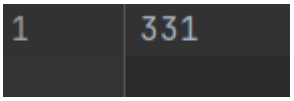
```
        return n
    previous, current = 0, 1
    for _ in range(n - 1):
        previous, current = current, (previous + current) % 10

    return current


with open('input.txt', 'r') as file:
    n = int(file.read().strip())
with open('output.txt', 'w') as file:
    file.write(str(last_digit_fibonacci(n)))
print(time.perf_counter() - t_start)
print(memory_usage())
```

Текстовое объяснение решения:

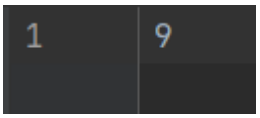
В данном коде мы считаем последнюю цифру чисел фибоначчи. У чисел фибоначчи наблюдается периодичность. Последние числа фибоначчи повторяются каждый 60 раз. Сначала мы считаем остаток, от деления, номер нужного числа от 60. Затем мы считаем число фибоначчи равное этому остатку, после этого мы ищем остаток от данного числа деления на 10.

Результат работы кода на примерах из текста задачи:

input.txt:

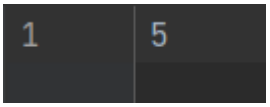


output.txt:



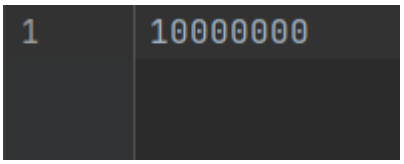
input.txt:

output.txt:

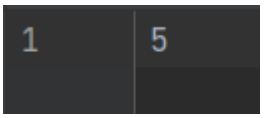


Результат работы кода на максимальных:

input.txt:



output.txt:



Результат работы кода на минимальных значениях:
input.txt:

output.txt:

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.002	25
Пример из задачи	0.002	25
Пример из задачи	0.002	25
Верхняя граница диапазона значений входных данных из текста задачи	0.003	25

Вывод по задаче:

В данной задаче мы с помощью математики смогли оптимизировать алгоритм для решения задачи .

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.

Объяснение:

В данной задаче мы используем два модуля time и memory_profiler.

Для вычисления времени работы, мы используем модуль time. Сначала засекая время перед началой работы алгоритма, и выводя разницу между текущим временем и временем, которое было перед началом работы.

Для вычисления памяти мы используем модуль memory_profiler. Чтобы узнать сколько потратила памяти программа, нужно использовать функцию memory_usage.

Вывод по задаче:

В данной задаче мы изучили инструменты, для вычисления времени работы программы и затраченной памяти.

Вывод:

В данной лабораторной работе мы изучили основы языка Python для дальнейшей работы по курсу.