САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №0
по курсу «Алгоритмы и структуры данных»

Выполнил:
Скворцов Д.А.
К3140

Проверил:
Афанасьев А. В.

Санкт-Петербург
2024 г.

Содержание отчета

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 $-10^9 \le a, b \le 10^9$. Выход: единственное целое число — результат сложения a + b.

```
a, b = map(int, input().split())
while not (-10**9 <= a <= 10**9 and -10**9 <= b <= 10**9):
    print("Неверные входные данные")
    a, b = map(int, input().split())
print(a + b)
```

Преобразуем строку с двумя числами, идущими через пробел, в список строк, а после в 2 целых числа при помощи *map*. Выводим сумму.

```
ing\ITMO Education\Algorithms and Data Structures\Lab 0\Task 1\1.py'
12 25
37
```

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

```
a, b = map(int, input().split())
while not (-10**9 <= a <= 10**9 and -10**9 <= b <= 10**9):
    print("Неверные входные данные")
    a, b = map(int, input().split())
print(a + b ** 2)
```

Выполняем те же преобразования, но выводим сумму *a* с *b* уже во второй степени.

```
ducation\Algorithms and Data Structures\Lab 0\Task 1\2.py'
12 25
637
```

Задача №3

Выполните задачу a + b с использованием файлов.

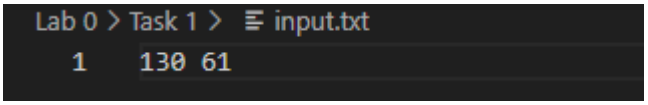
- Имя входного файла: input.txt
- Имя выходного файла: output.txt
- Формат входного файла. Входной файл состоит из одной строки, ко- торая содержит два целых числа a и b. Для этих чисел выполняются условия $-10^9 \le a,b \le 10^9$.
- Формат выходного файла. Выходной файл единственное целое число — результат сложения a + b.

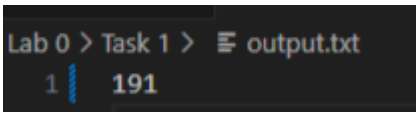
Примеры:

input.txt	12 25	130 61
output.txt	37	191

```
with open('input.txt', 'r') as f:
    a, b = map(int, f.readline().strip().split())
with open('output.txt', 'w') as f:
    print(a + b, file=f)
```

Открываем входной и выходной файлы при помощи менеджера контекста *with*. Считываем строку с числами из файла, преобразуем в переменные, записываем в выходной файл при помощи возможностей *print*.





Задача №4

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

```
with open('input.txt', 'r') as f:
    a, b = map(int, f.readline().strip().split())
with open('output.txt', 'w') as f:
    print(a + b ** 2, file=f)
```

Выполняем те же действия, но вносим в *print* сумму *a* с *b* уже во второй степени.

Lab 0 > Task 1 > input.txt

```
1    130 61
```

Lab 0 > Task 1 > output.txt

```
1    3851
```

Задание №2. Число Фибоначчи

Ваша цель – разработать эффективный алгоритм для подсчета чисел Фибоначчи. Вам предлагается начальный код на Python, который содержит наивный рекурсивный алгоритм.

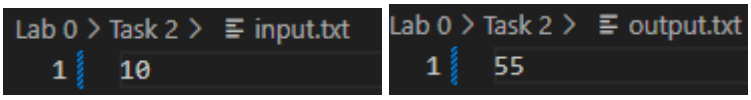
- Имя входного файла: input.txt
- Имя выходного файла: output.txt
- Формат входного файла. Целое число n. $0 \le n \le 45$.
- Формат выходного файла. Число F_n.

Пример:

input.txt	10
output.txt	55

```python
import time

def main():
    def calc_fib(n):
        if n in cache: return cache[n]
        if 0 <= n <= 1: return n
        res = calc_fib(n - 1) + calc_fib(n - 2)
        cache[n] = res
        return res

    cache = {}
    t_start = time.perf_counter()
    with open('input.txt', 'r') as f:
        n = int(f.readline().strip())
    if 0 <= n <= 45:
        with open('output.txt', 'w') as f:
            f.write(str(calc_fib(n)))
    else:
        print("Число не соответствует условию.")
    print("Время работы: %s секунд " % (time.perf_counter() - t_start))
```

Для оптимизации рекурсивного алгоритма используем мемоизации при помощи словаря. Вне функции настраиваем таймер и проверку выполнения условия.




```
Время работы: 0.0007154999766498804 секунд
```


```
Время работы: 0.00104750000173226 секунд
```

	Время выполнения
Нижняя граница диапазона значений входных данных из текста задачи	0.0007554999901913106 с.
Пример из задачи	0.0007154999766498804 с.
Верхняя граница диапазона значений входных данных из текста задачи	0.001033900014590472 с.

Вывод.

При увеличении значений незначительно возрастает время выполнения программы и затрачиваемая память. Алгоритм эффективен для заданного диапазона значений входных данных.

Задание №3. Еще про числа Фибоначчи

Определение последней цифры большого числа Фибоначчи.

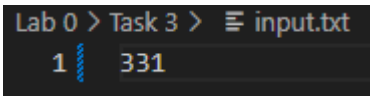
- Имя входного файла: input.txt
- Имя выходного файла: output.txt
- Формат входного файла. Целое число n. $0 \le n \le 10^7$.
- Формат выходного файла. Одна последняя цифра числа F_n.

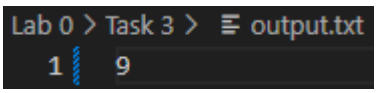
Пример:

input.txt	331	327305
output.txt	9	5

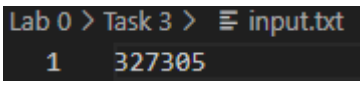
```python
import time

t_start = time.perf_counter()

def calc_fib(n):
    if 0 <= n <= 1: return n
    a, b = 0, 1
    for _ in range(n-1):
        a, b = b, (a + b) % 10
    return b

def main():
    t_start = time.perf_counter()
    with open('input.txt', 'r') as f:
        n = int(f.readline().strip())
    if 0 <= n <= 10**7:
        with open('output.txt', 'w') as f:
            f.write(str(calc_fib(n)))
    else:
        print("Число не соответствует условию.")
    print("Время работы: %s секунд " % (time.perf_counter() - t_start))
```

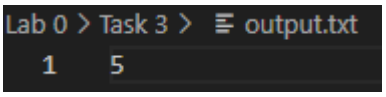
Для уменьшения затрат по памяти используем алгоритм, основанный на цикле. В функции *calc_fib* работаем только с остатками чисел *a* и *b*, поскольку остальное не влияет на ответ. Далее структура функции *main* аналогична заданию №2.






```
Время работы: 0.0008648999501019716 секунд
```






```
Время работы: 0.03678530000615865 секунд
```

	Время выполнения
Нижняя граница диапазона значений входных данных из текста задачи	0.0009900000295601785 с.
Пример из задачи	0.03678530000615865 с.

Верхняя граница диапазона значений входных данных из текста задачи	0.6842000999604352 с.

Вывод.

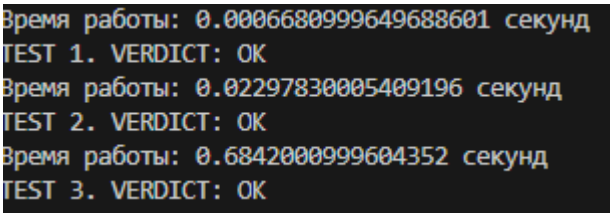
Алгоритм работает за линейное время и не затрачивает память на рекурсивный вызов. Эффективен для заданного диапазона значений.

Задание№4. Тестирование ваших алгоритмов.

Задача: вам необходимо протестировать время выполнения вашего алгоритма в Задании 2 и Задании 3.

```
with open('input.txt', 'w') as f:
    f.write('331')
main()
with open('output.txt', 'r') as f:
    print("TEST 1. VERDICT:", "OK" if f.readline().strip() == '9' else
"WRONG")

with open('input.txt', 'w') as f:
    f.write('327305')
main()
with open('output.txt', 'r') as f:
    print("TEST 2. VERDICT:", "OK" if f.readline().strip() == '5' else
"WRONG")

with open('input.txt', 'w') as f:
    f.write('10000000')
main()
with open('output.txt', 'r') as f:
    print("TEST 3. VERDICT:", "OK" if f.readline().strip() == '5' else
"WRONG")
```

Тестирующая система записывает числа во входной файл, запускает основной код (листинг с задания №3) и сверяет ответ с представленным в примере.


Время работы: 0.0006680999649688601 секунд
TEST 1. VERDICT: OK
Время работы: 0.02297830005409196 секунд
TEST 2. VERDICT: OK
Время работы: 0.6842000999604352 секунд
TEST 3. VERDICT: OK

Вывод по лабораторной работе

Для операций сложения и возведения в квадрат чисел, не превышающих 10^9 по модулю, алгоритм работает за константное время.
Для эффективной работы рекурсивного алгоритма Фибоначчи диапазон входных чисел небольшой, так как программа не экономна по памяти.
Алгоритм нахождения последней цифры числа Фибоначчи работает линейно и более эффективный, чем предыдущий, так как использует минимальные затраты памяти, что исключает медленную арифметику больших чисел.