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:
Заботкина М.А.
К3139

Проверил:
Афанасьев А.В.

Санкт-Петербург
2024 г.

Содержание отчета

Задачи

Задание №1. Ввод-вывод

Задача 1.1

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())
if (-10**9 <= a <= 10**9) and (-10**9 <= b <= 10**9):
    print(a + b)
else:
    print('Числа не подходят под условие')
```

1) На вход программа получает строку с двумя числами, которую с помощью метода split() делит по пробелам на два отдельных элемента, сохраняя в виде списка, map применяет int к каждому элементу, таким образом из стокового вида они переходят в целочисленный и присваиваются переменным a и b.
2) Происходит проверка на соответствие условию: если числа входят в эти промежутки, то они складываются и результат выводится на экран, в противном случае выводится соответствующий текст.

Результат работы кода на примерах из текста задачи:

```
12 25
37

Process finished with exit code 0
```

```
130 61
191

Process finished with exit code 0
```

Результат работы кода на минимальных и максимальных значениях:

```
-1000000000 -1000000000
-2000000000
```

```
1000000000 1000000000
2000000000
```

Вывод по задаче: я вспомнила, как писать код на python и как работает функция map

Задача 1.2

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

```
a, b = map(int, input().split())
if (-10**9 <= a <= 10**9) and (-10**9 <= b <= 10**9):
    print(a + b**2)
else:
    print('Числа не подходят под условие')
```

1) На вход программа получает строку с двумя числами, которую с помощью метода split() делит по пробелам на два отдельных элемента, сохраняя в виде списка, map применяет int к каждому элементу, таким образом из стокового вида они переходят в целочисленный и присваиваются переменным a и b.
2) Происходит проверка на соответствие условию: если числа входят в эти промежутки, то a складывается с b, возведённым в квадрат, и результат выводится на экран, в противном случае выводится соответствующий текст.

Результат работы кода на примерах из текста задачи:

```
12 25
637

Process finished with exit code 0
```

```
130 61
3851

Process finished with exit code 0
```

Результат работы кода на минимальных и максимальных значениях:

```
-1000000000 -1000000000
999999999000000000

Process finished with exit code 0
```

```
1000000000 1000000000
1000000001000000000

Process finished with exit code 0
```

Вывод по задаче: я вспомнила, как в python возводить число в степень и писать элементарный код

Задача 1.3

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 $-10^9 \le a, b \le 10^9$.

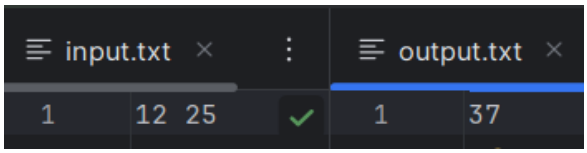
• Формат выходного файла. Выходной файл единственное целое число — результат сложения a + b.

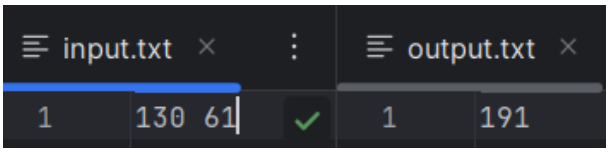
```
f = open('input.txt')
a, b = map(int, f.readline().split())
f.close()

f = open('output.txt', 'w')
if (-10**9 <= a <= 10**9) and (-10**9 <= b <= 10**9):
    f.write(str(a + b))
else:
    print('Числа не подходят под условие')
f.close()
```

1) Открывается файл для чтения, считывается строка с двумя числами, которая с помощью метода split() делится по пробелам на два отдельных элемента и сохраняется в виде списка, map применяет int к каждому элементу, таким образом из стокового вида они переходят в целочисленный и присваиваются переменным a и b. Файл закрывается.

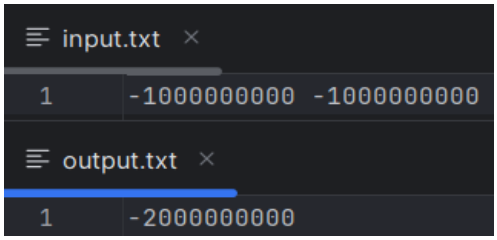
2) Открывается файл для записи, если числа входят в промежуток, то результат сложения записывается в файл, в противном случае на экран выводится соответствующий текст. Файл закрывается.

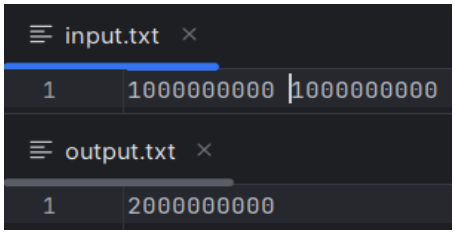
Результат работы кода на примерах из текста задачи:





Результат работы кода на минимальных и максимальных значениях:





Вывод по задаче: я вспомнила, как работать с файлами и узнала дополнительную информацию о работе с ними.

Задача 1.4

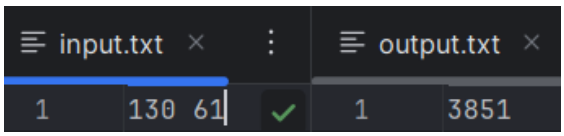
Выполните задачу a+b 2 с использованием файлов аналогично предыдущему пункту.

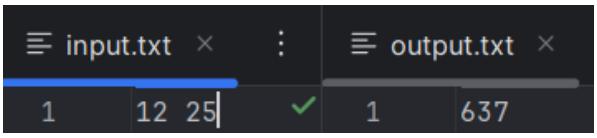
```
f = open('input.txt')
a, b = map(int, f.readline().split())
f.close()

f = open('output.txt', 'w')
if (-10**9 <= a <= 10**9) and (-10**9 <= b <= 10**9):
    f.write(str(a + b**2))
else:
    print('Числа не подходят под условие')
f.close()
```

1) Открывается файл ínput.txt для чтения, считывается строка с двумя числами, которая с помощью метода split() делится по пробелам на два отдельных элемента и сохраняется в виде списка, map применяет int к каждому элементу, таким образом из стокового вида они переходят в целочисленный и присваиваются переменным a и b. Файл закрывается.

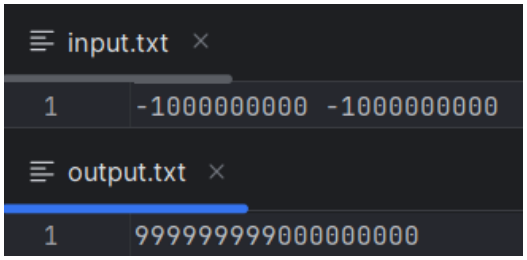
2) Открывается файл output.txt для записи, если числа входят в промежуток, то результат сложения a с числом b, возведённым в квадрат, записывается в файл, в противном случае на экран выводится соответствующий текст. Файл закрывается.

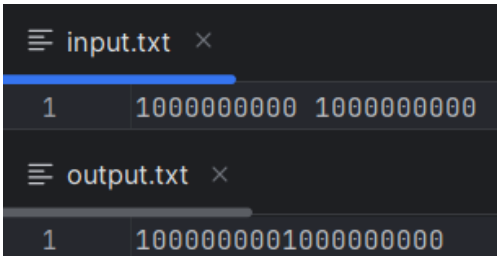
Результат работы кода на примерах из текста задачи:





Результат работы кода на минимальных и максимальных значениях:





Вывод по задаче: закрепление инфрмации о работе с файлами, полученной в задаче 1.3

Задание №2. Число Фибоначчи

Определение последовательности Фибоначчи:

$$F_0 = 0$$

$$F_1 = 1$$

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

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

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

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.

```
f = open('input.txt')
n = int(f.readline())
f.close()
f = open('output.txt', 'w')
f0, f1 = 0, 1
if 0 <= n <= 1:
    f.write(str(n))
elif 2 <= n <= 45:
    for i in range(2, n + 1):
        f0, f1 = f1, f0 + f1
    f.write(str(f1))
else:
    print('Число не подходит под ограничения')

f.close()
```

1) Открывается файл input.txt для чтения. Считывается первая строка из файла, преобразуется её в целое число и присваивается переменной n. Файл input.txt закрывается.

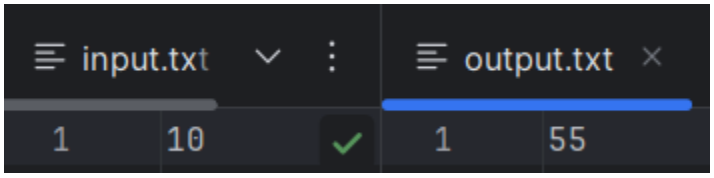
2) Открывается файл output.txt для записи. Переменным f0 и f1 присваиваются значения 0 и 1 соответственно. Эти переменные будут использоваться для вычисления чисел Фибоначчи.

3) Происходит проверка числа n: если значение n находится в диапазоне от 0 до 1 включительно, то значение записывается в файл output.txt в виде строки без изменений; если значение n находится в диапазоне от 2 до 45 включительно, то запускается цикл от 2 до n включительно, в противном случае выводится соответствующий текст.

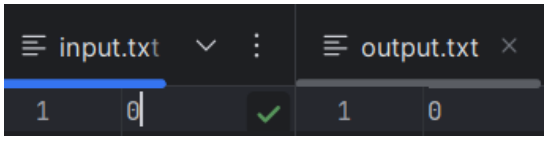
4) В цикле обновляются значения f0 и f1 для вычисления следующего числа Фибоначчи. f1 становится новым значением числа Фибоначчи, а f0 — предыдущим значением.

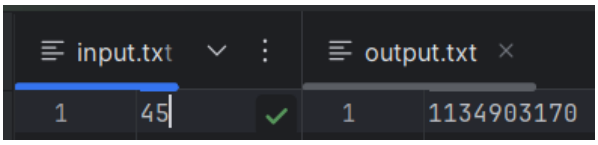
5) Записывается значение f1 (последнее вычисленное число Фибоначчи) в файл output.txt в виде строки. Файл output.txt закрывается.

Результат работы кода на примерах из текста задачи:



Результат работы кода на минимальных и максимальных значениях:





Вывод по задаче: закрепление информации из задания №1, написала рабочий код для нахождения n-го числа Фибоначчи

Задание №3. Ещё 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 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 Fn.

• 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.

```
f = open('input.txt')
n = int(f.readline())
f.close()

f = open('output.txt', 'w')
f0, f1 = 0, 1
if 0 <= n <= 1:
    f.write(str(n))
elif 2 <= n <= 10**7:
    for i in range(2, n + 1):
        f0, f1 = f1, (f0 + f1) % 10
    f.write(str(f1))
else:
    print('Число не подходит под ограничения')

f.close()
```

1) Открывается файл input.txt для чтения. Считывается первая строка из файла, преобразуется её в целое число и присваивается переменной n. Файл input.txt закрывается.

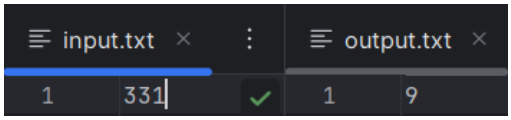
2) Открывается файл output.txt для записи. Переменным f0 и f1 присваиваются значения 0 и 1 соответственно. Эти переменные будут использоваться для вычисления чисел Фибоначчи.

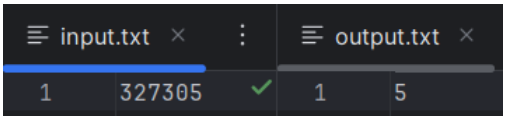
3) Происходит проверка числа n: если значение n находится в диапазоне от 0 до 1 включительно, то значение записывается в файл output.txt в виде строки без изменений; если значение n находится в диапазоне от 2 до 45 включительно, то запускается цикл от 2 до n включительно, в противном случае выводится соответствующий текст.

4) В цикле обновляются значения f0 и f1 для вычисления следующего числа Фибоначчи. f1 становится новым значением числа Фибоначчи, а f0 — предыдущим значением, так как нам нужно найти последнюю цифру числа, то находим остаток от деления f1 на 10.

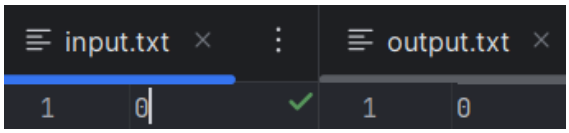
5) Записывается значение f1 (последняя цифра n-го числа Фибоначчи) в файл output.txt в виде строки. Файл output.txt закрывается.

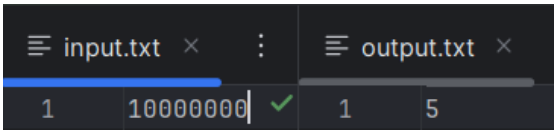
Результат работы кода на примерах из текста задачи:





Результат работы кода на минимальных и максимальных значениях:





Вывод по задаче: поиск последней цифры n-го числа Фибоначчи происходит дольше, чем поиск самого числа, поэтому выгоднее вместо f1 записывать не сумму двух предыдущих чисел, а остаток от деления этой суммы на 10.

Задание №4. Тестирование алгоритмов

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.

Тестирование алгоритма из задания №2:

Время работы:

```
import time
t_start = time.perf_counter()
f = open('input.txt')
n = int(f.readline())
f.close()

f = open('output.txt', 'w')
f0, f1 = 0, 1
if 0 <= n <= 1:
    f.write(str(n))
elif 2 <= n <= 45:
    for i in range(2, n + 1):
        f0, f1 = f1, f0 + f1
    f.write(str(f1))
else:
    print('Число не подходит под ограничения')

f.close()
print(f'Время работы: {time.perf_counter()-t_start} секунд')
```

1) Импортируется модуль time, чтобы использовать его функции для измерения времени.

2) time.perf_counter() записывает текущее время в переменную t_start. Функция perf_counter() возвращает высокоточное время (в секундах) с момента запуска программы, которое будет использовано для замера времени работы программы.

3) Вставляется код, написанный для задания №2
4) Вычисляется разница между текущим временем (time.perf_counter()) и временем начала программы (t_start). Таким образом, выводится время выполнения программы в секундах.

Объём используемой процессом памяти:

```
import psutil
process = psutil.Process()

f = open('input.txt')
n = int(f.readline())
f.close()
f = open('output.txt', 'w')
f0, f1 = 0, 1
if 0 <= n <= 1:
    f.write(str(n))
elif 2 <= n <= 45:
    for i in range(2, n + 1):
        f0, f1 = f1, f0 + f1
    f.write(str(f1))
else:
    print('Число не подходит под ограничения')

f.close()
print(process.memory_info().rss / 2**20, 'Мб')
```

1) Импортируется библиотека psutil, которая предоставляет функции для работы с информацией о процессах и системе, включая мониторинг использования памяти.
2) Создаётся объект process, представляющий текущий процесс программы.
3) Вставляется код, написанный для задания №2
4) Используется метод memory_info() объекта process для получения информации о текущем использовании памяти (в байтах). Значение делится на 2^{20} для перевода байтов в мегабайты. Затем это значение выводится на экран.

Результаты тестирования для алгоритма из задания №2:

	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0005395999760366976	22.1953125
Пример из задачи	0.0006326999864540994	22.11328125
Верхняя граница диапазона значений	0.0008575000101700425	22.2421875

входных данных из текста задачи		

Тестирование алгоритма из задания №3

Время работы:

```
import time
t_start = time.perf_counter()
f = open('input.txt')
n = int(f.readline())
f.close()

f = open('output.txt', 'w')
f0, f1 = 0, 1
if 0 <= n <= 1:
    f.write(str(n))
elif 2 <= n <= 10**7:
    for i in range(2, n + 1):
        f0, f1 = f1, (f0 + f1) % 10
    f.write(str(f1))
else:
    print('Число не подходит под ограничения')

f.close()
print(f'Время работы: {time.perf_counter()-t_start} секунд')
```

1) Импортируется модуль time, чтобы использовать его функции для измерения времени.
2) time.perf_counter() записывает текущее время в переменную t_start. Функция perf_counter() возвращает высокоточное время (в секундах) с момента запуска программы, которое будет использовано для замера времени работы программы.
3) Вставляется код, написанный для задания №3
4) Вычисляется разница между текущим временем (time.perf_counter()) и временем начала программы (t_start). Таким образом, выводится время выполнения программы в секундах.

Объём используемой процессом памяти:

```
import psutil
process = psutil.Process()
f = open('input.txt')
n = int(f.readline())
f.close()

f = open('output.txt', 'w')
f0, f1 = 0, 1
if 0 <= n <= 1:
    f.write(str(n))
elif 2 <= n <= 10**7:
    for i in range(2, n + 1):
        f0, f1 = f1, (f0 + f1) % 10
    f.write(str(f1))
else:
```

```
    print('Число не подходит под ограничения')

f.close()
print(process.memory_info().rss / 2**20, 'Мб')
```

1) Импортируется библиотека psutil, которая предоставляет функции для работы с информацией о процессах и системе, включая мониторинг использования памяти.
2) Создаётся объект process, представляющий текущий процесс программы.
3) Вставляется код, написанный для задания №3
4) Используется метод memory_info() объекта process для получения информации о текущем использовании памяти (в байтах). Значение делится на 2^{20} для перевода байтов в мегабайты. Затем это значение выводится на экран.

Результаты тестирования для алгоритма из задания №2:

	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0004572999896481633	22.4921875
Пример из задачи	0.00057790003484115	22.3046875
Пример из задачи	0.02606289996765554	22.1171875
Верхняя граница диапазона значений входных данных из текста задачи	0.7547038999618962	22.27734375

Вывод по задаче: вычисление больших значений требует намного больше времени, но при этом процесс при каждом вычислении занимает одинаковое количество памяти

Вывод

В ходе работы был рассмотрен ряд задач, связанных с вводом-выводом данных, и обработкой данных из файлов. Каждая задача помогла вспомнить и закрепить базовые навыки программирования на python. Особое внимание было уделено задачам по вычислению чисел Фибоначчи, что позволило вспомнить алгоритмы для нахождения этих чисел Фибоначчи, а также закрепить понимание последовательностей. Тестирование времени и

памяти работы алгоритмов показало, что выполнение некоторых задач требует большего времени по мере увеличения входных данных, но количество занимаемой процессом памяти остаётся неизменным.