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

Отчет по лабораторной работе №1

по курсу «Алгоритмы и структуры данных»

Тема: Тема работы

Вариант 1

Выполнила:
Петрова М. В
К3139

Санкт-Петербург
2024 г.

Содержание отчета

Задачи по варианту

Задача №1. Ввод-вывод

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 $a + b$.
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 $a + b^2$.
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 —результа сложения $a + b$.

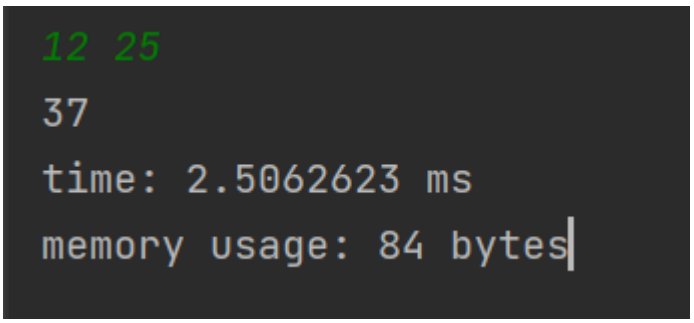
4. Выполните задачу $a+b^2$ с использованием файлов аналогично предыдущему пункту.

1) a + b

```
import time
import sys

start = time.perf_counter()
def summ(a, b):
    if (-1_000_000_000 <= a and a <= 1_000_000_000)
and (-1_000_000_000 <= b and b <= 1_000_000_000):
        return a + b
    return "Введите числа еще раз"

a, b = map(int, input().split())

result = summ(a, b)
stop = time.perf_counter()
print(result)
print("time: %s ms" % (stop - start))
memory_usage = sys.getsizeof(a) + sys.getsizeof(b) +
sys.getsizeof(result)
print(f"memory usage: {memory_usage} bytes")
```

Текстовое объяснение решения.
1. Импорт необходимых модулей time и sys time:
2. Функция summ возвращает a + b.
3. Засечение общего времени выполнения программы python t_start = time.perf_counter() time.perf_counter(): фиксируем время старта выполнения всей программы.
4. Ввод данных a, b = map(int, input().split()) input():
6. Вызов функции для вычисления суммы result = summ(a, b)
7. Засечение времени окончания вычисления stop = time.perf_counter()
8. Выводим результат сложения чисел.
9. Вычисление времени выполнения в микросекундах duration_microseconds = (stop - start) * 1_000_000
10. Вывод общего времени выполнения программы в микросекундах
11. Подсчёт использования памяти в байтах memory_usage = sys.getsizeof(a) + sys.getsizeof(b) + sys.getsizeof(result) sys.getsizeof():
12. Вывод информации об использовании памяти

Тест	Время выполнения микросек	Затраты памяти
12 25	2.5062623	84
130 61	2.8272263	84
-10^9 -10^9	13.0578245	88
10^9 10^9	16.815409	88



```
130 61
191
time: 2.8272263 ms
memory usage: 84 bytes
```

```
-1000000000 -1000000000
-2000000000
time: 13.0578245 ms
memory usage: 88 bytes
```

```
1000000000 1000000000
2000000000
time: 16.815409 ms
memory usage: 88 bytes
```

Вывод по задаче:

Программа принимает два целых числа, вычисляет их сумму и выводит результат. Также она замеряет и выводит время выполнения операции сложения в микросекундах, общее время работы программы в миллисекундах и объём памяти, занимаемый переменными.

2)a + b^2

```
import time
import sys

t_start = time.perf_counter()

def summ(a, b):
    if (-1_000_000_000 <= a and a <= 1_000_000_000)
and (-1_000_000_000 <= b and b <= 1_000_000_000):
        return a + b**2
    return "Введите числа еще раз"
```

```
a, b = map(int, input().split())
result = summ(a, b)
stop = time.perf_counter()
print(result)
print("time: %s ms" % (stop - t_start))
memory_usage = sys.getsizeof(a) + sys.getsizeof(b) +
sys.getsizeof(result)
print(f"memory usage: {memory_usage} bytes")
```

Текстовое объяснение решения.
1. Импорт необходимых модулей time и sys time:
2. Функция summ возвращает a + b**2.
3. Засечение общего времени выполнения программы python t_start = time.perf_counter() time.perf_counter(): фиксируем время старта выполнения всей программы.
4. Ввод данных a, b = map(int, input().split()) input():
6. Вызов функции для вычисления суммы result = summ(a, b)
7. Засечение времени окончания вычисления stop = time.perf_counter()
8. Выводим результат сложения чисел.
9. Вычисление времени выполнения в микросекундах duration_microseconds = (stop - start) * 1_000_000
10. Вывод общего времени выполнения программы в микросекундах
11. Подсчёт использования памяти в байтах memory_usage = sys.getsizeof(a) + sys.getsizeof(b) + sys.getsizeof(result) sys.getsizeof():
12. Вывод информации об использовании памяти

Тест	Время выполнения микросек	Затраты памяти
130 61	3.3268469	84
-10^9 -10^9	10.5436458	88
10^9 10^9	9.4475704	88

```
130 61
3851
time: 3.3268469 ms
memory usage: 84 bytes
```

```
-1000000000 -1000000000
99999999000000000
time: 10.5436458 ms
memory usage: 88 bytes
```

```
1000000000 1000000000
100000001000000000
time: 9.4475704 ms
memory usage: 88 bytes
```

Вывод по задаче:
Программа принимает два целых числа, вычисляет сумму a и b^2 и выводит результат. Также она замеряет и выводит время выполнения операции сложения в микросекундах, общее время работы программы в миллисекундах и объём памяти, занимаемый переменными.

3) a + b

```python
import time
import sys

start = time.perf_counter()

with open('input.txt') as f:
```

```
    a, b = map(int, f.readline().split())
    if (-1_000_000_000 <= a and a <= 1_000_000_000)
and (-1_000_000_000 <= b and b <= 1_000_000_000):
        summ = a + b
    else:
        print("Введите числа еще раз")

with open('output.txt', 'w') as f:
    f.write(str(summ))
stop = time.perf_counter()
duration_microseconds = (stop - start) * 1_000_000
print("time: %s ms" % (time.perf_counter() - start))
memory_usage = sys.getsizeof(a) + sys.getsizeof(b) +
sys.getsizeof(summ)
print(f"memory usage: {memory_usage} bytes")
```

Текстовое объяснение решения.

1. Импорт необходимых модулей time и sys time:
2. Функция summ возвращает a + b**2.
3. Засечение общего времени выполнения программы python t_start = time.perf_counter() time.perf_counter(): фиксируем время старта выполнения всей программы.
4. Открытие файлов и считывание данных
5. Вычисление суммы
6. Запись суммы в файл output
7. Засечение времени окончания вычисления stop = time.perf_counter()
8. Выводим результат сложения чисел.
9. Вычисление времени выполнения в микросекундах duration_microseconds = (stop - start) * 1_000_000
10. Вывод общего времени выполнения программы в микросекундах
11. Подсчёт использования памяти в байтах memory_usage = sys.getsizeof(a) + sys.getsizeof(b) + sys.getsizeof(result) sys.getsizeof():
12. Вывод информации об использовании памяти

Тест	Время выполнения микросек	Затраты памяти
130 61	0.003857700000000005	84

-10^9 -10^9	0.003657600000000004	88
10^9 10^9	0.003002499999999998	88

```
time: 0.003857700000000056 ms
memory usage: 84 bytes
```

```
time: 0.0036576000000000004 ms
memory usage: 88 bytes
```

```
time: 0.0030024999999999982 ms
memory usage: 88 bytes
```

Вывод по задаче:
Программа считывает данные из файла и вычисляет сумму a и b и выводит результат. Также она замеряет и выводит время выполнения операции сложения в микросекундах, общее время работы программы в миллисекундах и объём памяти, занимаемый переменными. замеряет и выводит время выполнения операции сложения в микросекундах, общее время работы программы в миллисекундах и объём памяти, занимаемый переменными.

4) a + b^2

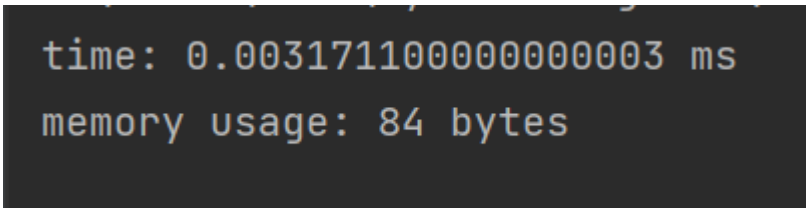
```python
import time
import sys


start = time.perf_counter()

with open('input.txt') as f:
```

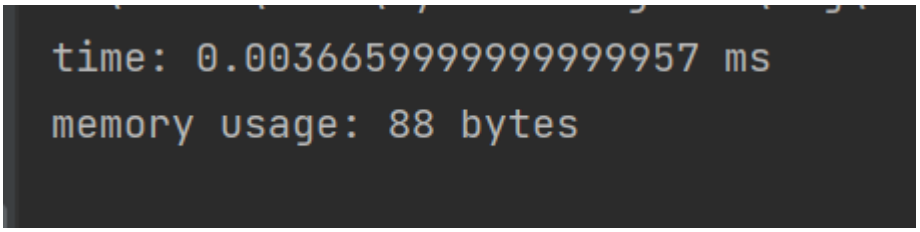
```
    a, b = map(int, f.readline().split())
    if (-1_000_000_000 <= a and a <= 1_000_000_000)
and (-1_000_000_000 <= b and b <= 1_000_000_000):
        summ = a + b**2
    else:
        print("Введите числа еще раз")

with open('output.txt', 'w') as f:
    f.write(str(summ))
stop = time.perf_counter()
print("time: %s ms" % (stop - start))
memory_usage = sys.getsizeof(a) + sys.getsizeof(b) +
sys.getsizeof(summ)
print(f"memory usage: {memory_usage} bytes")
```

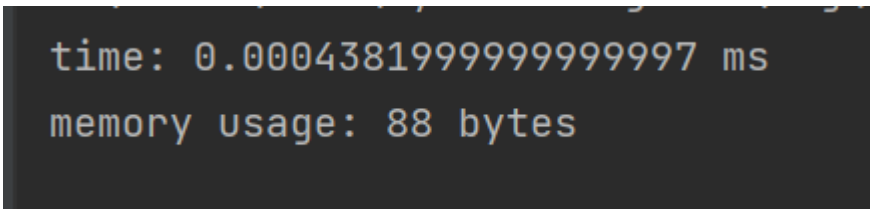
Текстовое объяснение решения.
1. Импорт необходимых модулей time и sys time:
2. Функция summ возращает a + b**2.
3. Засечение общего времени выполнения программы python t_start = time.perf_counter() time.perf_counter(): фиксируем время старта выполнения всей программы.
4.Открытие файлов и считывание данных
5.Вычисление суммы a и b**2
6.Запись суммы в файл output
7. Засечение времени окончания вычисления stop = time.perf_counter()
8. Выводим результат сложения чисел.
9. Вычисление времени выполнения в микросекундах duration_microseconds = (stop - start) * 1_000_000
10. Вывод общего времени выполнения программы в микросекундах
11. Подсчёт использования памяти в байтах memory_usage = sys.getsizeof(a) + sys.getsizeof(b) + sys.getsizeof(result) sys.getsizeof():
12. Вывод информации об использовании памяти

Тест	Время выполнения микросек	Затраты памяти
130 61	0.0031711000000000003	84

-10^9 -10^9	0.003665999999999995	88
10^9 10^9	0.0004381999999999999	88







Вывод по задаче:
Программа считывает данные из файла и вычисляет сумму a и b**2 и записывает результат в файл. Также она замеряет и выводит время выполнения операции сложения в микросекундах, общее время работы программы в миллисекундах и объём памяти, занимаемый переменными.

№2 Числа Фибоначчи

Определение последовательности Фибоначчи:

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_i &= F_{i-1} + F_{i-2} \text{ для } i \geq 2. \end{aligned} \tag{1}$$

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.
- Пример.

input.txt	10
output.txt	55

```
import time
import sys

start = time.perf_counter()



with open('input.txt') as f:
    n = int(f.readline())
    el1 = 0
    el2 = 1
    if n < 0 or n > 45:
        print("Неправильный формат данных. Ввведите
число еще раз")
        el1 = el2 = 0
```

```
    else:
        for i in range(2, n + 1):
            c = el2
            el2 = el1 + el2
            el1 = c

with open('output.txt', 'w') as f:
    f.write(str(el2))

stop = time.perf_counter()
print("time: %s ms" % (stop - start))
memory_usage = sys.getsizeof(n) + sys.getsizeof(el1)
+ sys.getsizeof(el2)
print(f"memory usage: {memory_usage} bytes")
```

Текстовое объяснение решения.

1. Импорт необходимых модулей time и sys time:

2. Засечение общего времени выполнения программы python t_start = time.perf_counter() time.perf_counter(): фиксируем время старта выполнения всей программы.

2. Открытие файла и чтение данных

- Если `n` меньше 0 или больше 10 миллионов, выводится сообщение о неверном формате данных, так как это недопустимые значения для задачи.

3. Вычисление числа Фибоначчи:

- Переменные `el1` и `el2` инициализируются как первые два числа последовательности Фибоначчи: 0 и 1 соответственно.
- Затем выполняется цикл, который вычисляет последовательность до индекса `n`. На каждой итерации:
  - Значение переменной `el2` обновляется как сумма двух предыдущих значений.
  - Предыдущее значение сохраняется в `el1` для следующего шага.

4. Запись результата:

- Результат (последняя цифра числа Фибоначчи для заданного `n`) записывается в файл `output.txt`.

5. Завершение измерения времени:

- Используется функция `time.perf_counter()`, чтобы замерить время окончания выполнения и рассчитать продолжительность программы в микросекундах.

6. Функция `sys.getsizeof()` используется для подсчета памяти, занятой переменными: `n`, `el1` и `el2`. Результат выводится в байтах.

Тест	Время выполнения микросек	Затраты памяти
10	0.0009959000000000001	84
0	0.000828499999999995	76
45	0.001370099999999999	88

```
time: 0.0009959000000000001 ms
memory usage: 84 bytes
```

```
time: 0.0008284999999999959 ms
memory usage: 76 bytes
```

```
time: 0.0013700999999999999 ms
memory usage: 88 bytes
```

Вывод

Эта программа предназначена для вычисления числа Фибоначчи для заданного числа `n` и вывода результата в файл `output.txt`. Программа также измеряет время выполнения и оценивает объем используемой памяти.

№3 Еще 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. $0 \leq n \leq 10^7$.
- 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.

```
import time
import sys

t_start = time.perf_counter()

with open('input.txt') as f:
    n = int(f.readline())
    el1 = 0
```

```
    el2 = 1
    if n < 0 or n > 10**7:
        print("Неправильный формат данных. Введите
число еще раз")
    else:

        for i in range(2, n + 1):
            c = el2
            el2 = (el1 + el2) % 10
            el1 = c
with open('output.txt', 'w') as f:
    f.write(str(el2))
stop = time.perf_counter()
print("time: %s ms" % (stop - t_start))
memory_usage = sys.getsizeof(n) + sys.getsizeof(el1)
+ sys.getsizeof(el2)
print(f"memory usage: {memory_usage} bytes")
```

Текстовое объяснение решения.

1. Импорт необходимых модулей time и sys time:

2. Засечение общего времени выполнения программы python t_start = time.perf_counter() time.perf_counter(): фиксируем время старта выполнения всей программы.

2. Открытие файла и чтение данных

- Если `n` меньше 0 или больше 10 миллионов, выводится сообщение о неверном формате данных, так как это недопустимые значения для задачи.

3. Вычисление последней цифры числа Фибоначчи:

- Инициализируются две переменные `el1 = 0` и `el2 = 1`, представляющие первые два числа Фибоначчи.
- Для каждого числа от 2 до `n`, программа пересчитывает очередное число Фибоначчи по модулю 10, чтобы сохранить только последнюю цифру. В цикле:
  - `el2` обновляется как сумма предыдущих двух чисел Фибоначчи, взятая по модулю 10.
  - `el1` обновляется для следующей итерации.

4. Запись результата:

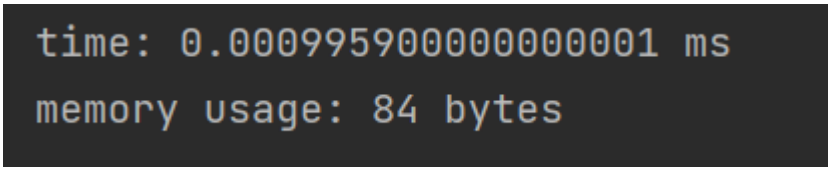
- Результат (последняя цифра числа Фибоначчи для заданного `n`) записывается в файл `output.txt`.

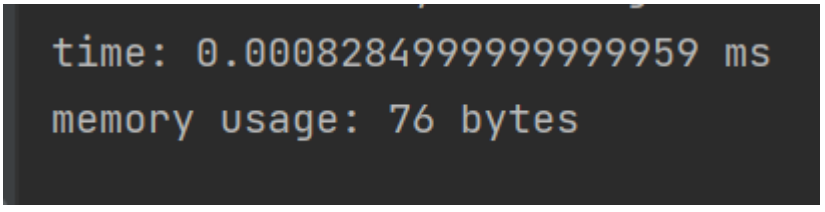
5. Завершение измерения времени:

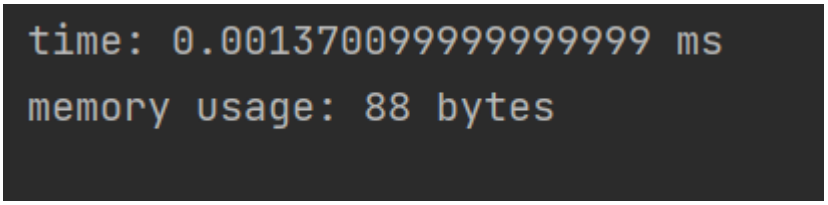
- Используется функция `time.perf_counter()`, чтобы замерить время окончания выполнения и рассчитать продолжительность программы в микросекундах.

6. Функция `sys.getsizeof()` используется для подсчета памяти, занятой переменными: `n`, `el1` и `el2`. Результат выводится в байтах.

Тест	Время выполнения микросек	Затраты памяти
10	0.0009959000000000001	84
0	0.0008284999999999995	76
45	0.0013709999999999999	88







Вывод

Эта программа предназначена для решения задачи по вычислению последней цифры числа Фибоначчи для заданного индекса n. Программа также измеряет время выполнения и подсчитывает используемую память.

Вывод по всей лабораторной.
Лабораторная 0 помогает вспомнить базовые конструкции python и алгоритмы решения задач на нахождение числа Фибоначчи.