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 1

Выполнил:
Мурашов Никита Александрович
Группа К3141

Проверил:
_____

Санкт-Петербург
2024 г.

Содержание отчета

Оглавление

Задачи по варианту

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 ≤ a, b ≤ 10^9. Выход: единственное целое число — результат сложения a + b.

```
while True:
    try:
        print(sum(map(int, input().split())))
        break
    except:
        print('Неверный ввод')
```

Описание кода:

1. Цикл ввода данных:
   Программа использует бесконечный цикл while True, чтобы повторно запрашивать ввод, пока пользователь не введет корректные данные.
2. Обработка ввода и вычисление суммы:
   a. программа считывает строку ввода и разбивает ее на отдельные элементы по пробелам.
   b. преобразует каждый элемент списка в целое число.
   c. вычисляет сумму введенных чисел.
   d. выводит результат на экран.
3. Обработка исключений:
   Если при преобразовании или суммировании возникает ошибка (например, введены нечисловые данные или недостаточное количество чисел), блок except перехватывает исключение и выводит сообщение 'Неверный ввод'.

4. Прерывание цикла:
   Если операция прошла успешно, используется break для выхода из цикла.

Задача не требует проверки времени выполнения и затрат памяти.

Вывод по задаче:
В процессе решения задачи я повторил основы обработки исключений в Python и научился организовывать надежный ввод данных, обеспечивая корректность работы программы при различных вводах пользователя.

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 выполняются условия −10^9 ≤ a, b ≤ 10^9. Выход: единственное целое число — результат сложения a + b^2.

```
while True:
    try:
        a, b = map(int, input().split())
        print(a + b ** 2)
        break
    except:
        print('Неверный ввод')
```

Описание кода:

1. Ввод значений: аналогично задаче №1.
2. Проверка диапазона: аналогично задаче №1.
3. Сложение первого числа с квадратом второго:

Задача не требует проверки времени выполнения и затрат памяти.

Вывод по задаче:
Задача позволила закрепить усвоенный материал.

Задача №3.

Выполнить задачу №1, используя ввод-вывод через файлы.

```
with open('input.txt') as f:
    a, b = map(int, f.readline().split())
with open('output.txt', mode='w') as f:
    f.write(str(a + b))
```

Описание кода:

Программа аналогична 1 задаче, за исключением:

1. Чтения изначальных данных из txt файла:
2. Запись результата в файл:

Задача не требует проверки времени выполнения и затрат памяти.

Вывод по задаче:
Выполнение этой задачи позволило мне освоить основы файлового ввода-вывода в Python, включая чтение данных из файла и запись результатов в другой файл.

Задача №4.

Выполнить задачу №2, используя ввод-вывод через файлы.

```
with open('input.txt') as f:
    a, b = map(int, f.readline().split())
with open('output.txt', mode='w') as f:
    f.write(str(a + (b ** 2)))
```

Описание кода:

Программа аналогична 2 задаче, за исключением:

1. Чтения изначальных данных из txt файла:
2. Запись результата в файл:

Задача не требует проверки времени выполнения и затрат памяти.

Вывод по задаче:
Задача позволила закрепить усвоенный материал.

Задание №2. Числа Фибоначчи

Разработать эффективный алгоритм для подсчета чисел Фибоначчи.

```
import time

start = time.perf_counter()
with open('input.txt') as f:
    n = int(f.readline())
if n == 0:
    answer = 0
elif n == 1:
    answer = 1
else:
    a, b = 0, 1
    for _ in range(n - 1):
        a, b = b, a + b
    answer = b
with open('output.txt', 'w') as f:
    f.write(str(answer))
print(time.perf_counter() - start)
```

Описание кода:

1. Импорт модуля для измерения времени: Используется import time для отслеживания времени выполнения программы.
2. Начало отсчета времени:

   Фиксируется время начала выполнения.
3. Чтение входных данных:

   Открывается файл и считывается значение.
4. Вычисление n-го числа Фибоначчи:
   a. Проверяются особые случаи, когда n равно 0 или 1.
   b. Для остальных значений используется цикл for для итеративного вычисления последовательности:
      i. Переменные a и b инициализируются как 0 и 1.
      ii. В цикле обновляются значения a и b по формуле a, b = b, a + b
   c. После цикла answer содержит n-е число Фибоначчи.
5. Результат записывается в файл.
6. Печатается время выполнения программы.

Вывод по задаче:

Я освоил эффективный итеративный метод вычисления чисел Фибоначчи, что позволяет избежать избыточных вычислений и существенно снижает время работы программы.

	Время работы (секундах)
Нижняя граница диапазона значений входных данных из текста задачи (n = 0)	0.0013
Пример 1 из задачи (n = 10)	0.0018
Верхняя граница диапазона значений входных данных из текста задачи (n = 45)	0.0019

Задание №3. Числа Фибоначчи: последняя цифра.

Разработать эффективный алгоритм для подсчета последней цифры чисел Фибоначчи.

```
import time


start = time.perf_counter()
with open('input.txt') as f:
    n = int(f.readline())
if n == 0: answer = 0
elif n == 1: answer = 1
else:
    a, b = 0, 1
    for _ in range(n - 1):
        a, b = b, (a + b) % 10
    answer = b
with open('output.txt', 'w') as f:
    f.write(str(answer))
print(time.perf_counter() - start)
```

Описание кода:

Программа аналогична прошлой, за исключением того, что в итоговый файл записывается не всё число, а только его последняя цифра.

	Время работы (микросекунды)
Нижняя граница диапазона значений входных данных из текста задачи (n = 0)	0.0021
Пример 1 из задачи (n = 331)	0.00214
Пример 2 из задачи (n = 327305)	0.06
Верхняя граница диапазона значений входных данных из текста задачи (n = 10^7)	1.75

Выводы по задаче:

Благодаря вычислениям по модулю 10, код избегает работы с большими числами, которые быстро растут в последовательности Фибоначчи. Даже если n очень велико, каждое число не занимает больше 1 цифры, что делает память и скорость выполнения оптимальными.

Задание №4. Проверка времени работы заданий №2, 3.

В таблицах к заданию №2 и №3 представлены результаты времени работы программ.

Вывод по заданию:
Время работы не сильно изменялось в зависимости от данных, что доказывает эффективность написанной программы и ее линейную сложность.

Вывод

В ходе выполнения лабораторной работы я повторил основы ввода-вывода данных в Python, как через консоль, так и через файлы. Разработал эффективные алгоритмы для вычисления чисел Фибоначчи и их последней цифры, оптимизировав использование времени и памяти. Также приобрел опыт в измерении времени выполнения программ, что важно для оценки эффективности алгоритмов.