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:

Бай М.О.

К3141

Проверил:

Афанасьев А.В.

Санкт-Петербург

2024 г.

# Содержание отчета

[Содержание отчета 2](#_Toc177344361)

[Задачи по варианту 3](#_Toc177344362)

[Задача №1. Ввод-вывод 3](#_Toc177344363)

[Задача №2. Число Фибоначчи 9](#_Toc177344364)

[Задача №3. Еще про числа Фибоначчи 12](#_Toc177344365)

[Задача №4. Тестирование ваших алгоритмов 15](#_Toc177344366)

[Вывод 17](#_Toc177344367)

# 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 ≤ a, b ≤ 109 . Выход: единственное целое число — результат сложения a + b.
2. Задача a + b2. В данной задаче требуется вычислить значение a + b2. Вход: одна строка, которая содержит два целых числа a и b. Для этих чисел выполняются условия −109 ≤ a, b ≤ 109. Выход: единственное целое число — результат сложения a + b2.
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≤ a, b ≤109.
   * Формат выходного файла. Выходной файл единственное целое число — результат сложения a + b.
4. Выполните задачу a+b2 с использованием файлов аналогично предыдущему пункту.

Задача a+b

Листинг кода:

num = input()  
splitnum = num.split()  
if -10\*\*9 <= int(splitnum[0]) <= 10\*\*9+1:  
 if -10\*\*9 <= int(splitnum[1]) <= 10\*\*9+1:  
 print(int(splitnum[0]) + int(splitnum[1]))  
 else: print('Error')  
else: print('Error')

num = input()

splitnum = num.split()

if -10\*\*9 <= int(splitnum[0]) <= 10\*\*9+1:

if -10\*\*9 <= int(splitnum[1]) <= 10\*\*9+1:

print(int(splitnum[0]) + int(splitnum[1]))

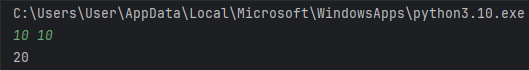
else: print('Error')

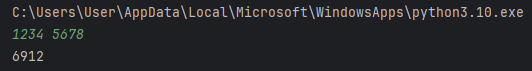
else: print('Error')

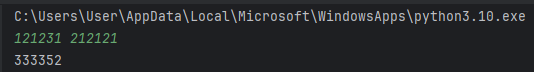
Текстовое объяснение решения:

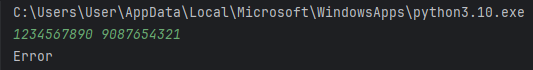
Принимаем строку, которую разделяем по пробелу и присваиваем значения переменной “splitnum’. Проверяем, удовлетворяют ли числа условию задачи (быть в диапазоне от -10\*\*9 до 10\*\*9). Если удовлетворяют, то выводим их сумму, вызывая целочисленные значения по коду элемента, если нет – выводим ошибку.

Результат работы кода, учитывая случаи несоответствия изначальных чисел условию:









Вывод по задаче:

В данной задаче данные вводятся из строки и вывод так же сделан через консоль.

Задача a + b2

Листинг кода:

num = input()  
splitnum = num.split()  
if -10\*\*9 <= int(splitnum[0]) <= 10\*\*9+1:  
 if -10\*\*9 <= int(splitnum[1]) <= 10\*\*9+1:  
 print(int(splitnum[0]) + int(splitnum[1])\*\*2)  
 else:  
 print('Error')  
else:  
 print('Error')

num = input()

splitnum = num.split()

if -10\*\*9 <= int(splitnum[0]) <= 10\*\*9+1:

if -10\*\*9 <= int(splitnum[1]) <= 10\*\*9+1:

print(int(splitnum[0]) + int(splitnum[1])\*\*2)

else:

print('Error')

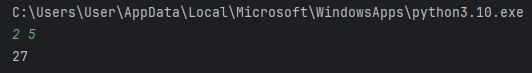
else:

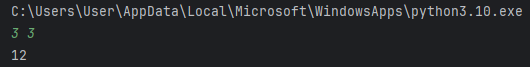
print('Error')

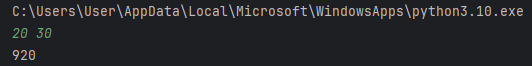
Текстовое объяснение решения:

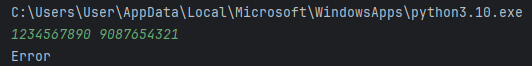
Принимаем строку, которую разделяем по пробелу и присваиваем значения переменной “splitnum’. Проверяем, удовлетворяют ли числа условию задачи (быть в диапазоне от -10\*\*9 до 10\*\*9). Если удовлетворяют, то выводим их сумму, вызывая целочисленные значения по коду элемента, при этом возводим второй элемент в квадрат, если нет – выводим ошибку.

Результат работы кода, учитывая случаи несоответствия изначальных чисел условию:









Вывод по задаче:

Данная задаче не отличается от предыдущей, за исключением того, что вторую переменную возводим в квадрат.

Задача a+b с использованием файлов

Листинг кода:

num = open('input.txt').read()  
splitnum = num.split()  
if -10\*\*9 <= int(splitnum[0]) <= 10\*\*9+1:  
 if -10\*\*9 <= int(splitnum[1]) <= 10\*\*9+1:  
 a = (int(splitnum[0]) + int(splitnum[1]))  
 open('output.txt', 'w').write(str(a))  
 else:  
 print('Error')  
else:  
 print('Error')

num = open('input.txt').read()

splitnum = num.split()

if -10\*\*9 <= int(splitnum[0]) <= 10\*\*9+1:

if -10\*\*9 <= int(splitnum[1]) <= 10\*\*9+1:

a = (int(splitnum[0]) + int(splitnum[1]))

open('output.txt', 'w').write(str(a))

else:

print('Error')

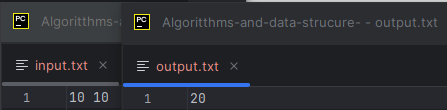
else:

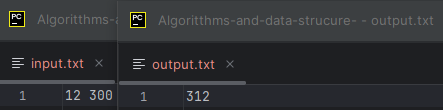
print('Error')

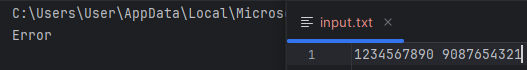
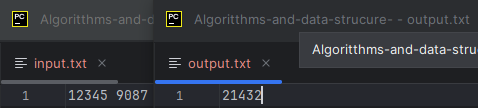
Текстовое объяснение решения:

Принимаем строку, которую разделяем по пробелу и присваиваем значения переменной “splitnum’. Проверяем, удовлетворяют ли числа условию задачи (быть в диапазоне от -10\*\*9 до 10\*\*9). Если удовлетворяют, то присваиваем их сумму переменной ‘a’, вызывая целочисленные значения по индексу элемента, если нет – выводим ошибку. Открываем файл output.txt и записываем в него значение переменной ‘a’ в строковом формате. Если изначальные числе не удовлетворяют условию – выводим ошибку.

Результат работы кода, учитывая случаи несоответствия изначальных чисел условию:







Вывод по задаче:

В данной задаче используется работа с файлами.

Задача a + b2 с использованием файлов

Листинг кода:

num = open('input.txt').read()  
splitnum = num.split()  
if -10\*\*9 <= int(splitnum[0]) <= 10\*\*9+1:  
 if -10\*\*9 <= int(splitnum[1]) <= 10\*\*9+1:  
 a = (int(splitnum[0]) + int(splitnum[1])\*\*2)  
 open('output.txt', 'w').write(str(a))  
 else:  
 print('Error')  
else:  
 print('Error')

num = open('input.txt').read()

splitnum = num.split()

if -10\*\*9 <= int(splitnum[0]) <= 10\*\*9+1:

if -10\*\*9 <= int(splitnum[1]) <= 10\*\*9+1:

a = (int(splitnum[0]) + int(splitnum[1])\*\*2)

open('output.txt', 'w').write(str(a))

else:

print('Error')

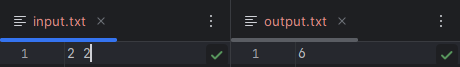
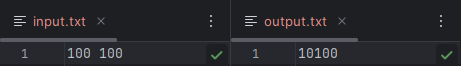
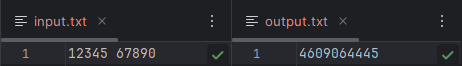
else:

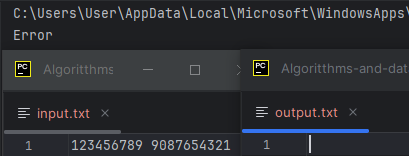
print('Error')

Текстовое объяснение решения:

Принимаем строку, которую разделяем по пробелу и присваиваем значения переменной “splitnum’. Проверяем, удовлетворяют ли числа условию задачи (быть в диапазоне от -10\*\*9 до 10\*\*9). Если удовлетворяют, то присваиваем их сумму переменной ‘a’, вызывая целочисленные значения по индекс элемента и возводя вторую переменную в квадрат, если нет – выводим ошибку. Открываем файл output.txt и записываем в него значение переменной ‘a’ в строковом формате. Если изначальные числе не удовлетворяют условию – выводим ошибку.

Результат работы кода, учитывая случаи несоответствия изначальных чисел условию:

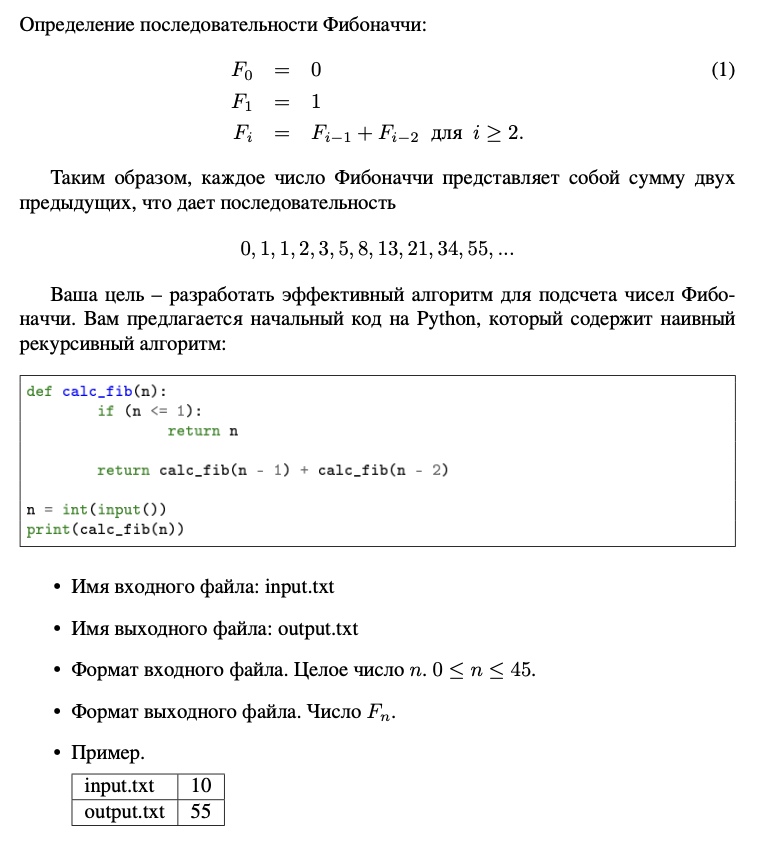
  



Вывод по задаче:

В данной задаче используется работа с файлами.

## [Задача №2. Число Фибоначчи](#_Задача_№2._Число)



Листинг кода:

f1 = 1  
f2 = 1  
i = 0  
  
n = int(open('input.txt').read())  
if n == 0:  
 open('output.txt', 'w').write(str(0))  
elif n == 1 or n == 2:  
 open('output.txt', 'w').write(str(1))  
elif 0 <= n <= 45:  
 while i < n - 2:  
 f\_s = f1+f2  
 f1 = f2  
 f2 = f\_s  
 i = i + 1  
  
 open('output.txt', 'w').write(str(f\_s))  
else:  
 print('Error')

f1 = 1

f2 = 1

i = 0

n = int(open('input.txt').read())

if n == 0:

open('output.txt', 'w').write(str(0))

elif n == 1 or n == 2:

open('output.txt', 'w').write(str(1))

elif 0 <= n <= 45:

while i < n - 2:

f\_s = f1+f2

f1 = f2

f2 = f\_s

i = i + 1

open('output.txt', 'w').write(str(f\_s))

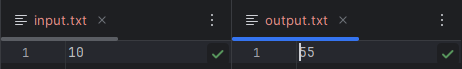
else:

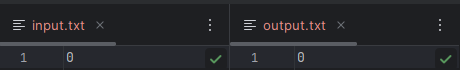
print('Error')

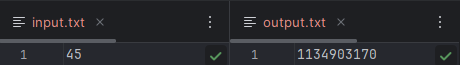
Текстовое объяснение решения:

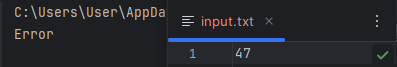
Задаем переменные ‘f1’ и ‘f2’ и записываем в них первые два числа последовательности Фибоначчи. Создаем вспомогательную переменную ‘i’. Открываем файл и принимаем число в целочисленном формате в переменную ‘n’. Если изначальное значение не соответствует условию – выводим ошибку. Далее делаем проверку исходного значения чисел 1 и 2, и если исходное значение равно 0, то открываем файл output.txt и записываем значение в него 0 в строковом формате. Если исходное значение равно 1 или 2, то открываем файл output.txt и записываем значение в него 1 в строковом формате. Иначе запускаем цикл while, где ‘i’ (счетчик чисел) сравнивается с исходным числом, из которого мы вычитаем 2 т.к. первые два числа уже записаны. Задаем переменную ‘f\_s’, которая равна сумме двух предыдущих чисел последовательности. Присваиваем переменной ‘f1’ значение ‘f2’ а переменной ‘f2’ значение ‘f\_s’. Добавляем к счетчику чисел ‘i’ единицу. После завершения цикла открываем файл output.txt и записываем значение в него значение ‘f\_s’ в строковом формате.

Результат работы кода, учитывая случаи несоответствия изначальных чисел условию:





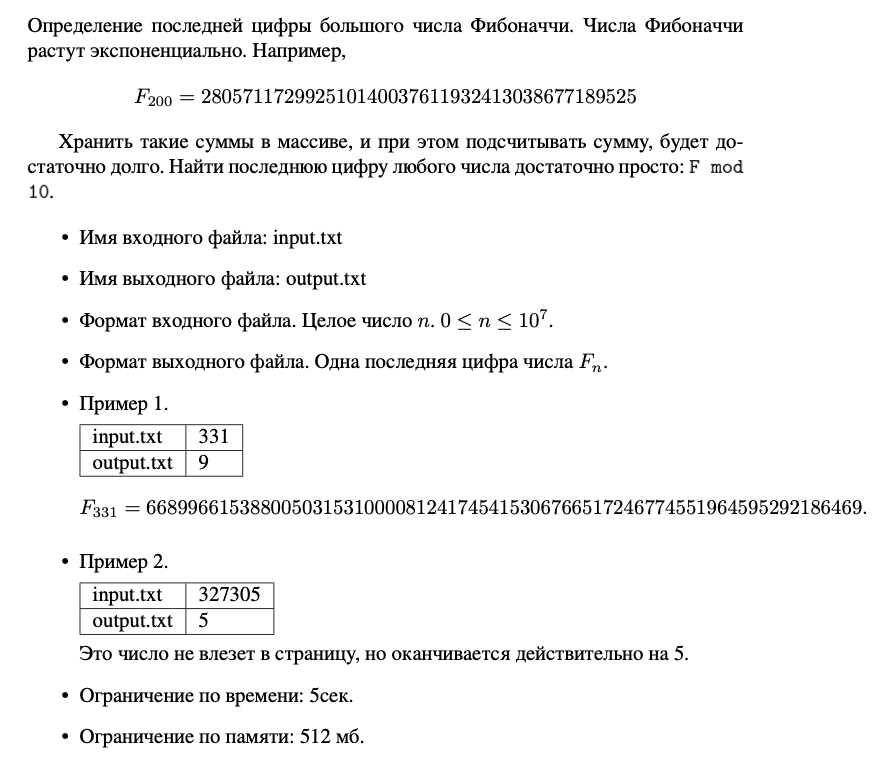




Вывод по задаче:

Скорость выполнения алгоритма растет линейно: чем больше мы подаём на вход число n, тем дольше выполняется программа. В данной задаче использованы навыки работы с файлами, каскадными условиями и циклом ‘while’.

## [Задача №3. Еще про числа Фибоначчи](#_Задача_№3._Еще)



Листинг кода:

import time  
t\_start = time.perf\_counter()  
  
f1 = 1  
f2 = 1  
i = 0  
  
  
n = int(open('input.txt').read())  
  
if n == 0:  
 open('output.txt', 'w').write(str(0))  
 print(time.perf\_counter() - t\_start)  
elif n == 1 or n == 2:  
 open('output.txt', 'w').write(str(1))  
 print(time.perf\_counter() - t\_start)  
elif 0 <= n <= 10\*\*7+1:  
 while i < n - 2:  
 f\_s = f1+f2  
 f1 = f2%10  
 f2 = f\_s%10  
 i = i + 1  
 open('output.txt', 'w').write(str(f\_s % 10))  
 print(time.perf\_counter() - t\_start)

import time

t\_start = time.perf\_counter()

f1 = 1

f2 = 1

i = 0

n = int(open('input.txt').read())

if n == 0:

open('output.txt', 'w').write(str(0))

print(time.perf\_counter() - t\_start)

elif n == 1 or n == 2:

open('output.txt', 'w').write(str(1))

print(time.perf\_counter() - t\_start)

elif 0 <= n <= 10\*\*7+1:

while i < n - 2:

f\_s = f1+f2

f1 = f2%10

f2 = f\_s%10

i = i + 1

open('output.txt', 'w').write(str(f\_s % 10))

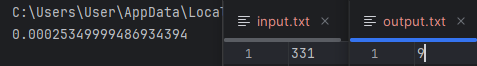
print(time.perf\_counter() - t\_start)

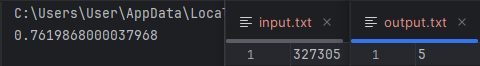
Текстовое объяснение решения:

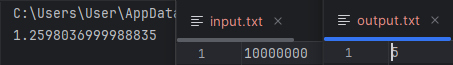
Для замера времени работы программы импортируем библиотеку time. Создаем переменную t\_start, в которую записываем время, пройденное от начала запуска программы. В конце берем время работы программы с помощью time.perf\_counter() и вычитаем из него время начала. Выводим время работы в секундах.

Задаем переменные ‘f1’ и ‘f2’ и записываем в них первые два числа последовательности Фибоначчи. Создаем вспомогательную переменную ‘i’. Открываем файл и принимаем число в целочисленном формате в переменную ‘n’. Если изначальное значение не соответствует условию – выводим ошибку. Далее делаем проверку исходного значения чисел 1 и 2, и если исходное значение равно 0, то открываем файл output.txt и записываем значение в него 0 в строковом формате. Берем время работы программы с помощью time.perf\_counter() и вычитаем из него время начала. Выводим время работы в секундах. Если исходное значение равно 1 или 2, то открываем файл output.txt и записываем значение в него 1 в строковом формате. Берем время работы программы с помощью time.perf\_counter() и вычитаем из него время начала. Выводим время работы в секундах. Иначе запускаем цикл while, где ‘i’ (счетчик чисел) сравнивается с исходным числом, из которого мы вычитаем 2 т.к. первые два числа уже записаны. Задаем переменную ‘f\_s’, которая равна сумме двух предыдущих чисел последовательности. Присваиваем переменной ‘f1’ последнюю цифру переменной ‘f2’ а переменной ‘f2’ последнюю цифру переменной ‘f\_s’. Добавляем к счетчику чисел ‘i’ единицу. После завершения цикла открываем файл output.txt и записываем значение в него последнюю цифру значение ‘f\_s’ путем остатка от деления на 10 в строковом формате.

Результат работы кода:







|  |  |
| --- | --- |
|  | Время выполнения, c |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.00023700000019744039 |
| Пример из задачи | 0.00045789999421685934 |
| Пример из задачи | 0.03813089997856878 |
| Верхняя граница диапазона значений входных данных из текста задачи | 1.1880753000150435 |

Вывод по задаче:

Скорость выполнения алгоритма растет линейно: чем больше мы подаём на вход число n, тем дольше выполняется программа. В данной задаче использованы навыки работы с файлами, каскадными условиями и циклом ‘while’. Т.к. значения в данной задаче больше, чем в прошлой,

## [Задача №4. Тестирование ваших алгоритмов](#_Задача_№4._Тестирование)

Текст задачи:

Вам необходимо протестировать время выполнения вашего алгоритма в Задании 2 и Задании 3.

Листинг кода:

import time  
t\_start = time.perf\_counter()  
  
//Код программы

print(time.perf\_counter() - t\_start)

import time

t\_start = time.perf\_counter()

//Код программы

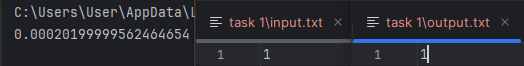
print(time.perf\_counter() - t\_start)

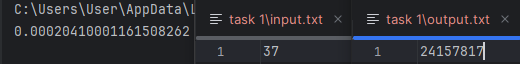
Текстовое объяснение решения:

Для замера времени работы программы импортируем библиотеку time. Создаем переменную t\_start, в которую записываем время, пройденное от начала запуска программы. В конце берем время работы программы с помощью time.perf\_counter() и вычитаем из него время начала. Выводим время работы в секундах.

Результат работы кода:

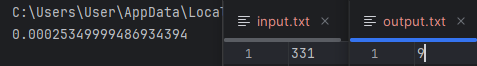


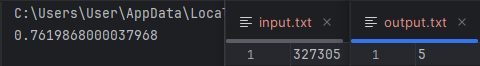


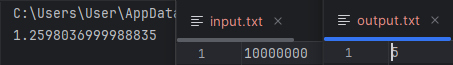












Вывод по задаче:

1. Проверка времени выполнения задачи полезна для оценивания оптимизации кода. Так же с ее помощью можно оценить скорость выполнения кода и сопоставить ее с временными ограничениями в условии задачи.

# 

# Вывод

В ходе выполнения работы были получены знания по выполнению задач на ввод/вывод, работу с циклами ‘if’ и ‘while’, работу с файлами и последовательностью Фибоначчи. Была изучена библиотека time для измерения времени работы алгоритма и применена на практике для оптимизации программ. Также была показана на практике линейная зависимость времени работы от входных данных: чем больше числа на входе, тем больше итераций выполняет программа, следовательно, тем дольше она получает результат.