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

Выполнил:

Афаккир Мухсин

Группа К3140

Проверил:

Афанасьев А. В.

Санкт-Петербург

2024 г.

# Содержание отчета

Оглавление

[Содержание отчета 2](#_Toc177731738)

[Задачи по варианту 3](#_Toc177731739)

[Задание № 1.1. a+b 3](#_Toc177731740)

[Задание №1.2. a+b2 4](#_Toc177731741)

[Задание №1.3. a+bс использованием файлов 5](#_Toc177731742)

[Задание №1.4. a+b2 с использованием файлов 6](#_Toc177731743)

[Задание №2. Число Фибоначчи. 7](#_Toc177731744)

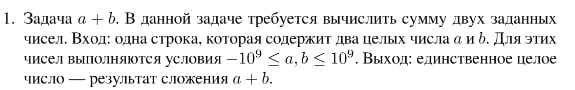
[Задание №3. Еще про числа Фибоначчи 9](#_Toc177731745)

[Вывод: 11](#_Toc177731746)

# Задачи по варианту

## Задание № 1.1. a+b

Текст задачи.



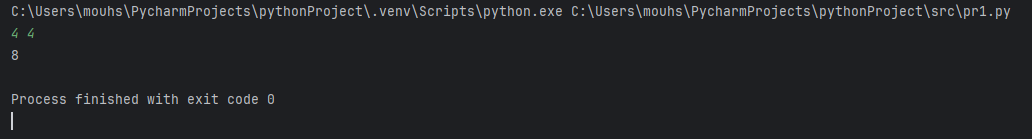
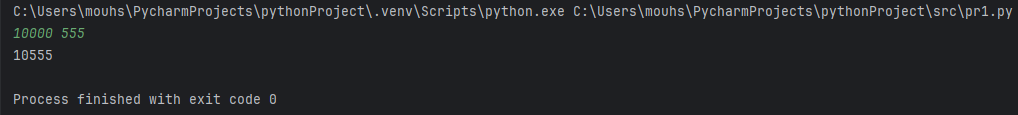
Код:

inputDataStr = input()  
minInput = -10 \*\* 9  
maxInput = 10 \*\* 9  
  
result = "Error input!"  
  
a, b = map(int, inputDataStr.split())  
  
if minInput <= a <= maxInput and minInput <= b <= maxInput:  
 result = a + b  
  
  
print(result)

Текстовое объяснение решения.

1. Заданы левая и правая границы интервала допустимых значений (minInput и maxInput).
2. Считываются два числа, введенных пользователем.
3. Числа проверяются на соответствие интервалу, если условие выполнено, то выводится их сумма, если условие не выполнено, то выводится соответствующее сообщение.

Примеры работы кода:

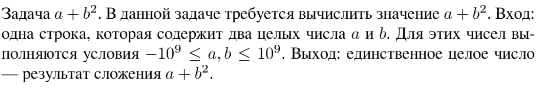


Вывод по задаче:

В решении задачи использованы переменные, if-else конструкции и операторы сравнения. Код выполняет поставленную задачу.

## Задание №1.2. a+b2

Текст задачи.



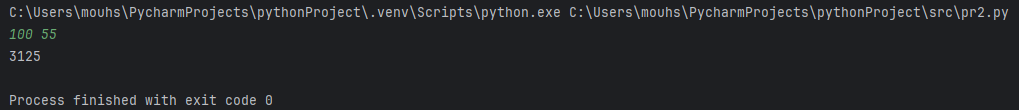
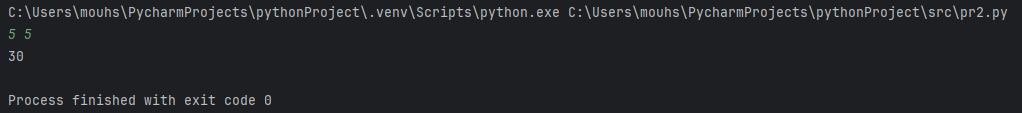
Код:

inputDataStr = input()  
minInput = -10 \*\* 9  
maxInput = 10 \*\* 9  
  
result = "Error input!"  
  
  
a, b = map(int, inputDataStr.split())  
  
if minInput <= a <= maxInput and minInput <= b <= maxInput:  
 result = a + b\*\*2  
  
  
print(result)

Текстовое объяснение решения.

1. Ранее заданы левая и правая границы интервала допустимых значений (minInput и maxInput).
2. Считываются два числа, введенных пользователем.
3. Числа проверяются на соответствие интервалу, если условие выполнено, то выводится сумма первого числа и квадрата второго, если условие не выполнено, то выводится соответствующее сообщение.

Примеры работы кода:

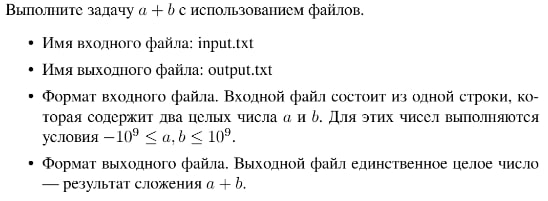


Вывод по задаче:

В решении задачи так же использованы переменные, if-else конструкции и операторы сравнения, в отличии от первой задачи, здесь потребовалось возвести одно из чисел во вторую степень. Код выполняет поставленную задачу.

## Задание №1.3. a+bс использованием файлов

Текст задачи:



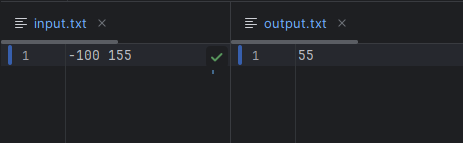
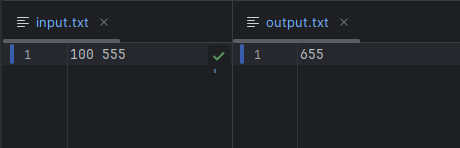
Код:

inputFile = open('input.txt', 'r')  
outputFile = open('output.txt', 'w')  
minInput = -10 \*\* 9  
maxInput = 10 \*\* 9  
  
result = "Error input!"  
  
  
a, b = map(int, inputFile.readline().split())  
  
if minInput <= a <= maxInput and minInput <= b <= maxInput:  
 result = a + b  
  
  
outputFile.write(str(result))  
  
inputFile.close()  
outputFile.close()

Текстовое объяснение решения.

1. Ранее заданы левая и правая границы интервала допустимых значений (minInput и maxInput).
2. Открывается файл input.txt с входными данными.
3. Считывается первая строка входного файла, из неё берутся числа a и b.
4. Числа проверяются на соответствие интервалу, если условие выполнено, то в переменную result вводится значение суммы чисел в строковом виде, открывается файл output.txt и в него записывается переменная result. Если условие не выполнено, то выводится соответствующее сообщение.

Пример работы кода:



Вывод по задаче:

Эта задача подобна первой, но требует умения работы с файлами (открытия, считывания и записи данных). Код выполняет поставленную задачу.

## Задание №1.4. a+b2 с использованием файлов

Текст задачи.



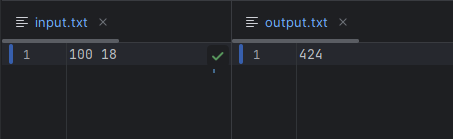
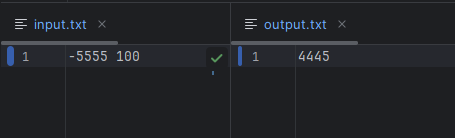
Код:

inputFile = open('input.txt', 'r')  
outputFile = open('output.txt', 'w')  
minInput = -10 \*\* 9  
maxInput = 10 \*\* 9  
  
result = "Error input!"  
  
  
a, b = map(int, inputFile.readline().split())  
  
if minInput <= a <= maxInput and minInput <= b <= maxInput:  
 result = a + b\*\*2  
  
outputFile.write(str(result))  
  
inputFile.close()  
outputFile.close()

Текстовое объяснение решения.

1. Ранее заданы левая и правая границы интервала допустимых значений (minInput и maxInput).
2. Открывается файл input.txt с входными данными.
3. Считывается первая строка входного файла, из неё берутся числа a и b.
4. Числа проверяются на соответствие интервалу, если условие выполнено, то в переменную result вводится значение суммы первого числа и квадрата второго в строковом виде, открывается файл output.txt и в него записывается переменная result. Если условие не выполнено, то выводится соответствующее сообщение.

Пример работы кода:

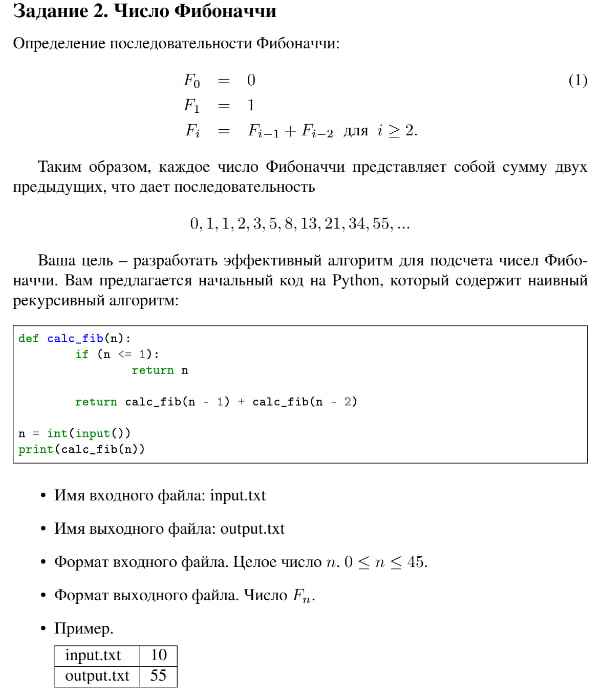


Вывод по задаче:

Эта задача подобна второй, но требует умения работы с файлами (открытия, считывания и записи данных). Код выполняет поставленную задачу.

## Задание №2. Число Фибоначчи.

Текст задачи:



Код:

import time  
import tracemalloc  
  
t\_start = time.perf\_counter()  
tracemalloc.start()  
  
inputFile = open('input.txt', 'r')  
outputFile = open('output.txt', 'w')  
  
result = "Error input!"  
  
  
n = int(inputFile.readline())  
  
if 0 <= n <= 45:  
 if n == 0 or n == 1:  
 result = n  
 else:  
 a, b = 0, 1  
 for i in range(n -1):  
 a, b = b, a + b  
  
 result = b  
  
outputFile.write(str(result))  
  
inputFile.close()  
outputFile.close()  
  
current, peak = tracemalloc.get\_traced\_memory()  
  
current = current / 10\*\*3  
peak = peak / 10\*\*3  
  
  
print("Time used: ", (time.perf\_counter() - t\_start))  
print("Current memory usage is: " + str(current))  
print("Peak memory usage was: " + str(peak))  
  
tracemalloc.stop()

Текстовое объяснение решения.

1) Импорт модуля time с последующим объявлением переменной-счетчика для отслеживания времени работы алгоритма.

2) Импортируйте модуль tracemalloc, а затем отследите память, чтобы отслеживать текущее использование памяти алгоритмом и пиковое использование памяти.

3) Открывается файл input.txt с входными данными.

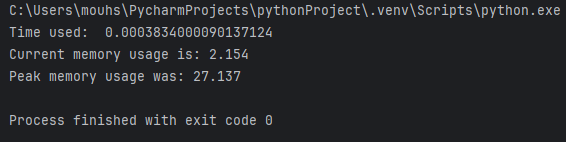
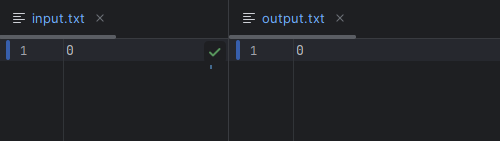
4) Считывается первая строка входного файла, из неё берется число a и проверяется на соответствие интервалу, если условие не выполняется, отображается сообщение об ошибке ввода.

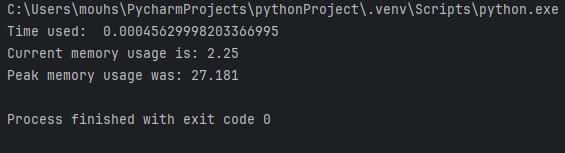
5) Подсчитывает конкретное число Фибоначчи.

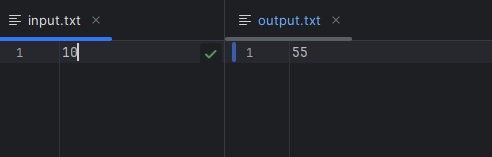
6) После подсчета в переменную result вводится число Фибоначчи, открывается файл output.txt и в него записывается переменная result.

7) Крайней командой выводится время выполнения алгоритма, текущее использование памяти и пиковое использование памяти.

Пример работы кода:







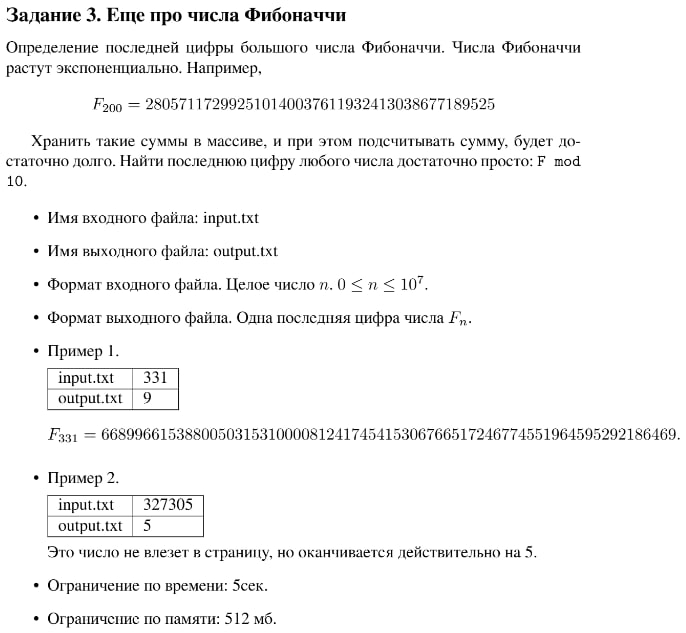
|  |  |  |
| --- | --- | --- |
|  | Время выполнения | использование памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | Время работы алгоритма: 0.0003834000090137124 | Текущее использование памяти: 2.154  Пиковое использование памяти: 27.137 |
| Пример из задачи | Время работы алгоритма: 0.0004562999823366995 | Текущее использование памяти: 2.25  Пиковое использование памяти: 27.181 |
| Верхняя граница диапазона значений входных данных из текста задачи | Время работы алгоритма: 0.0004290000069886446 | Текущее использование памяти: 3.562  Пиковое использование памяти: 27.181 |

Вывод по задаче:

При решении задачи потребовалось применить полученные ранее знания по работе с файлами, разобрать алгоритм для подсчета числа Фибоначчи и посчитать время работы алгоритма. Код выполняет поставленную задачу.

**Задание №3. Еще про числа Фибоначчи**

Текст задачи:



Код:

import time  
import tracemalloc  
  
  
t\_start = time.perf\_counter()  
tracemalloc.start()  
  
  
inputFile = open('input.txt', 'r')  
outputFile = open('output.txt', 'w')  
  
minInput = 0  
maxInput = 10 \*\* 7  
  
result = "Error input!"  
  
  
n = int(inputFile.readline())  
  
if minInput <= n <= maxInput:  
 if n == 0 or n == 1:  
 result = n  
 else:  
 a, b = 0, 1  
 for i in range(n -1):  
 a, b = b, a + b  
  
 result = b % 10  
  
outputFile.write(str(result))  
  
inputFile.close()  
outputFile.close()  
  
current, peak = tracemalloc.get\_traced\_memory()  
  
current = current / 10\*\*3  
peak = peak / 10\*\*3  
  
print("Time used: ", (time.perf\_counter() - t\_start))  
print("Current memory usage is: " + str(current))  
print("Peak memory usage was: " + str(peak))  
  
tracemalloc.stop()

Текстовое объяснение решения.

1) Импорт модуля time с последующим объявлением переменной-счетчика для отслеживания времени работы алгоритма.

2) Импортируйте модуль tracemalloc, а затем отследите память, чтобы отслеживать текущее использование памяти алгоритмом и пиковое использование памяти.

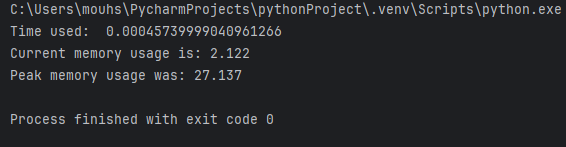
3) Открывается файл input.txt с входными данными.

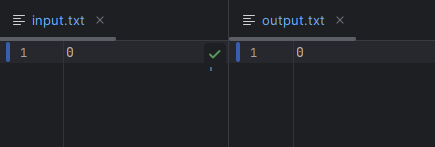
4) Считывается первая строка входного файла, из неё берется число a и проверяется на соответствие интервалу, если условие не выполняется, то код завершается преждевременно.

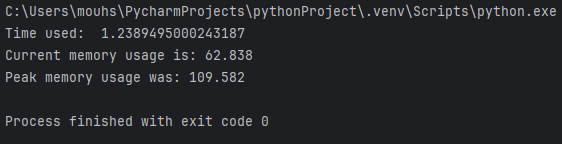
5) Ниже приведено описание кода, который вычисляет только последние цифры чисел Фибоначчи, это позволяет обрабатывать более крупные числа.

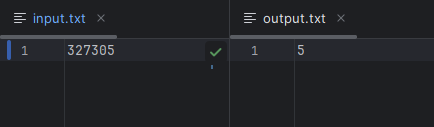
6) После подсчета в переменную result вводится последняя цифры числа Фибоначчи, открывается файл output.txt и в него записывается переменная result.

7) Крайней командой выводится время выполнения алгоритма, текущее использование памяти и пиковое использование памяти.









|  |  |  |
| --- | --- | --- |
|  | Время выполнения | использование памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | Время работы алгоритма: 0.00039770000148564577 | Текущее использование памяти: 2.122  Пиковое использование памяти: 27.137 |
| Пример из задачи | Время работы алгоритма: 0.0005326999817043543 | Текущее использование памяти: 2.202  Пиковое использование памяти: 27.183 |
| Пример из задачи | Время работы алгоритма: 1.238949000243187 | Текущее использование памяти: 62.838  Пиковое использование памяти: 109.582 |
| Верхняя граница диапазона значений входных данных из текста задачи | Время работы алгоритма: 197.36573100008536 | Текущее использование памяти: 978.446  Пиковое использование памяти: 1482.994 |

Вывод по задаче:

Задача потребовала значительной модификации функции, написанной для задания №3. В этом коде так же использовались логические конструкции, работа с файлами и создание функции. Код выполняет поставленную задачу.

**Вывод:**

При выполнении этой лабораторной работы я получил опыт в использовании логических конструкций, вспомнил, как работать с файлами, получил навык работы с модулем time для того, чтобы оценить скорость работы алгоритма на наглядном примере. Также я ознакомился с рекурсивным и итеративным алгоритмами, которые используются для вычисления чисел Фибоначчи.