

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

Выполнила:

Туманова Нелли Алексеевна

Группа К3141

Проверил:

Афанасьев А.В.

Санкт-Петербург

2024 г.

Содержание отчета

Оглавление

Содержание отчета	2
Задачи по варианту	3
Задание №1. Ввод-вывод	3
Задача №1.	3
Задача №2.	4
Задача №3.	5
Задача №4.	6
Задание №2. Числа Фибоначчи	7
Задание №3. Числа Фибоначчи: последняя цифра.....	10
Задание №4. Проверка времени работы заданий №2, 3.....	12
Вывод.....	12

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 $-10^9 \leq a, b \leq 10^9$. Выход: единственное целое число — результат сложения $a + b$.

```
for _ in range(3):
    a, b = map(int, input().split())

    if -(10 ** 9) <= a <= (10 ** 9) and -(10 ** 9) <= b <= (10 ** 9):
        print(a + b)
        break
    else:
        print('Incorrect numbers. Try again.')
```

Этот код выполняет следующие действия:

1. Ввод значений:

Программа ожидает ввода двух целых чисел (a и b) от пользователя, разделённых пробелом. Значения считываются из стандартного ввода и преобразуются в целые числа с помощью функции `map`.

2. Проверка диапазона:

Далее выполняется проверка, находятся ли введённые числа в заданном диапазоне. Конкретно, код проверяет, что каждое из чисел (a и b) не выходит за пределы диапазона от -10^9 до 10^9 . Если хотя бы одно из чисел не соответствует этому условию, программа просит ввести корректные данные (на это дается еще 3 попытки).

3. Сложение чисел:

Если оба числа находятся в допустимом диапазоне, программа вычисляет их сумму и выводит результат на экран с помощью функции `print`.

Таким образом, код обеспечивает ввод и обработку двух целых чисел с проверкой их корректности, после чего выводит сумму этих чисел, если все условия выполнены.

Задача не требует проверки времени выполнения и затрат памяти.

```
12 25
37
Для продолжения нажмите любую клавишу . . .
```

```
130 61
191
Для продолжения нажмите любую клавишу . . .
```

Вывод по задаче:

В процессе решения задачи, я вспомнила, как писать код на python, а также, каким образом производится ввод и запись значений в переменные.

Задача №2.

Задача $a + b^2$. В данной задаче требуется вычислить значение $a + b^2$. Выход: 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 $a + b^2$.

```
for _ in range(3):
    a, b = map(int, input().split())

    if -(10 ** 9) <= a <= (10 ** 9) and -(10 ** 9) <= b <= (10 ** 9):
        print(a + b ** 2)
        break
    else:
        print('Incorrect numbers. Try again.')
```

Этот код выполняет следующие действия:

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

Если оба числа находятся в допустимом диапазоне, программа вычисляет квадрат второго числа и складывает первое число с этим значением.

Таким образом, код обеспечивает ввод и обработку двух целых чисел с проверкой их корректности, после чего выводит сумму первого числа и квадрата второго, если все условия выполнены.

Задача так же не требует проверки времени выполнения и затрат памяти.

```
12 25
637
Для продолжения нажмите любую клавишу . . .

130 61
3851
Для продолжения нажмите любую клавишу . . .
```

Вывод по задаче:

Задача аналогична предыдущей => закрепляет усвоенный материал.

Задача №3.

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

```
with open('input.txt') as file:
    a, b = map(int, file.readline().split())

if -(10 ** 9) <= a <= (10 ** 9) and -(10 ** 9) <= b <= (10 ** 9):
    with open('output.txt', 'w+') as file:
        file.write(str(a + b))
else:
    print('Incorrect numbers. Try again.')
```

Этот код выполняет следующие действия:

1. Чтение из файла:

Сначала он открывает файл с именем input.txt в режиме чтения (по умолчанию в функции open). Программа считывает первую строку файла и разбивает её на две части, которые преобразует в целые числа a и b с помощью функции map. Метод with open('input.txt') as file: автоматически закроет файл после завершения блока.

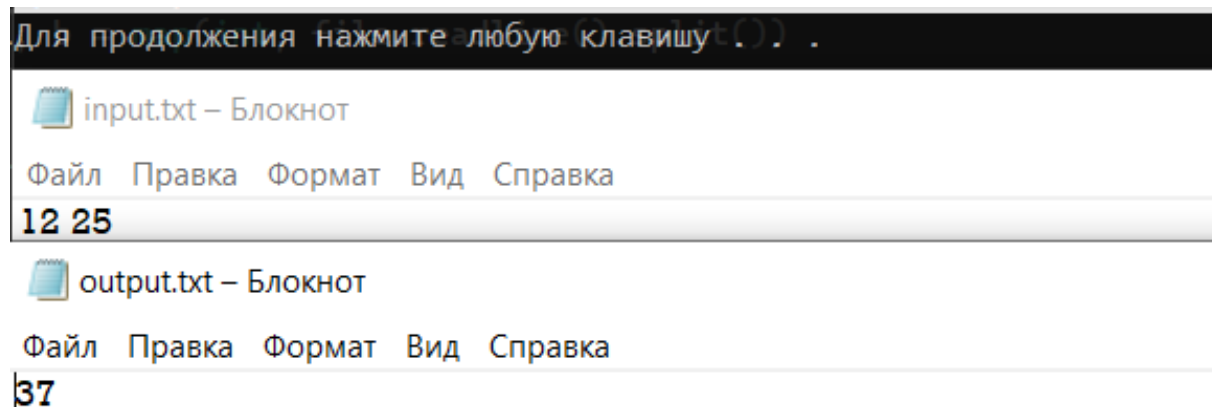
2. Проверка диапазона:

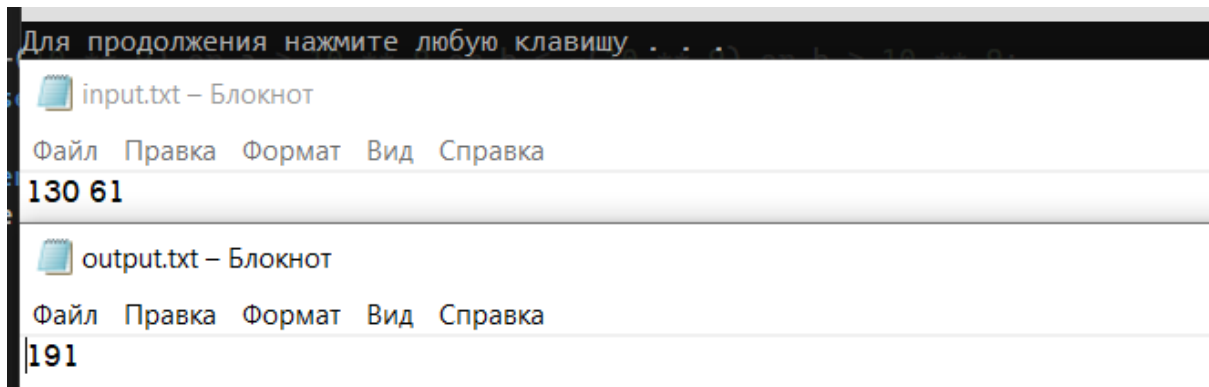
Если число не из диапазона, программа выведет сообщение и завершит работу.

3. Запись результата в файл:

Далее, если оба числа в допустимом диапазоне, программа открывает файл с именем output.txt в режиме записи ('w+'). Это позволяет не только записывать в файл, но и создавать его, если он не существовал ранее. В файл записывается сумма a и b в виде строки.

Таким образом, процесс можно кратко описать так: программа считывает два целых числа из файла, проверяет их на соответствие заданному диапазону, и, если данные валидны, записывает их сумму в другой файл. Задача не требует проверки времени выполнения и затрат памяти.





Вывод по задаче:

В процессе выполнения я вспомнила, как работать с файлами в python: открывать их, читать и записывать данные.

Задача №4.

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

```
with open('input.txt') as file:
    a, b = map(int, file.readline().split())

if -(10 ** 9) <= a <= (10 ** 9) and -(10 ** 9) <= b <= (10 ** 9):
    with open('output.txt', 'w+') as file:
        file.write(str(a + b ** 2))
else:
    print('Incorrect numbers. Try again.')
```

Этот код выполняет следующие действия:

1. Чтение из файла: аналогично задаче №3.
2. Проверка диапазона: аналогично задаче №3.
3. Запись результата в файл: аналогично задаче №3.

Таким образом, процесс можно кратко описать так: программа считывает два целых числа из файла, проверяет их на соответствие заданному диапазону, и, если данные валидны, записывает сумму первого числа и квадрата второго в другой файл.


Задача не требует проверки времени выполнения и затрат памяти.

Для продолжения нажмите любую клавишу . . .

 input.txt – Блокнот

Файл Правка Формат Вид Справка

12 25

 output.txt – Блокнот

Файл Правка Формат Вид Справка


637

Для продолжения нажмите любую клавишу . . .

 input.txt – Блокнот

Файл Правка Формат Вид Справка

130 61

 output.txt – Блокнот

Файл Правка Формат Вид Справка

3851|

Вывод по задаче:

Задача аналогична предыдущей => закрепляет усвоенный материал.

Задание №2. Числа Фибоначчи

Разработать эффективный алгоритм для подсчета чисел Фибоначчи.

```
def fib(n):
    last1, last2 = 0, 1
    for _ in range(n):
        last1, last2 = last2, last1 + last2

    return last1

import time
time_start = time.perf_counter()

with open('input.txt') as file:
    n = int(file.readline())

if 0 <= n <= 45:
    with open('output.txt', 'w+') as file:
        file.write(str(fib(n)))

    print(f'TIME {time.perf_counter() - time_start} microsec.')
else:
    print('Incorrect numbers. Try again.')
```

Этот код реализует вычисление n-го числа Фибоначчи с помощью простого итеративного метода:

1. Определение функции:

Функция `fib(n)` принимает одно целое число `n` как аргумент. Это значение указывает, какое по счёту число Фибоначчи нужно вычислить.

В теле функции проверяется, положительное ли число. Если нет – генерируется исключение с сообщением "The number must be positive".

Если `n` положительное, создаются две переменные `last1` и `last2`, которые инициализируются значениями 0 и 1 соответственно. Они будут хранить последние два числа последовательности Фибоначчи.

Цикл вычисления чисел Фибоначчи:

Цикл `for _ in range(n)` проходит от 0 до `n-1` (всего `n` итераций). На каждой итерации происходит обновление значений `last1` и `last2`. Переменная `last1` получает значение `last2`, а `last2` обновляется до суммы прежних `last1` и `last2`. Таким образом, на каждой итерации переменная `last2` хранит текущее число Фибоначчи, а `last1` — предыдущее число.

Возврат результата:

После завершения цикла функция возвращает значение переменной `last1`, которая на последней итерации содержит `n`-е число Фибоначчи.

2. Измерение времени выполнения:

С помощью модуля `time` начинается отсчёт времени. В переменную `time_start` при помощи `time.perf_counter()` фиксируется текущее время.

3. Чтение входных данных:

Открывается файл `input.txt` для чтения.

Из файла считывается первое число и преобразуется в целое, после чего это значение сохраняется в переменную `n`.

4. Проверка диапазона:

После чтения значения `n` программа выполняет проверку, чтобы убедиться, что число не больше 45. Если `n` не соответствует этим условиям, программа выводит сообщение о том, что данные введены некорректно, и завершится.

5. Запись результата:

После проверки диапазона открывается файл `output.txt` в режиме записи `w+`. Если файл не существует, он будет создан.

Вычисленное `n`-е число Фибоначчи (результат функции `fib(n)`) преобразуется в строку и записывается в файл.

6. Измерение времени выполнения:

В конце выводится время, затраченное на выполнение программы, в микросекундах: от текущего времени отнимается `time_start` – время на момент начала выполнения программы.

Таким образом, программа эффективно вычисляет число Фибоначчи и сохраняет результат в файл, всё это время контролируя диапазон входного значения и фиксируя время выполнения.

```

TIME 0.0005044999998062849 microsec.
Для продолжения нажмите любую клавишу . . .

input.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
0

output.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
0

TIME 0.0005703000060748309 microsec.
Для продолжения нажмите любую клавишу . . .

input.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
10

output.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
55

TIME 0.0007615000067744404 microsec.
Для продолжения нажмите любую клавишу . . .

input.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
45

output.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
1134903170

```

	Время работы (микросекунды)
Нижняя граница диапазона значений входных данных из текста задачи ($n = 0$)	0.0005
Пример 1 из задачи ($n = 10$)	0.00057
Верхняя граница диапазона значений входных данных из текста задачи ($n = 45$)	0.00076

Вывод по задаче:

Программа, вычисляющая n-е число Фибоначчи итеративно, оказывается эффективнее рекурсивной по нескольким важным причинам:

1. Дублирование вычислений:

В рекурсивной реализации функция может многократно вычислять одно и то же значение. Например, для вычисления `fib(5)` сначала вычисляется `fib(4)` и `fib(3)`, но для `fib(4)` снова будет вызван `fib(3)` и `fib(2)`, и так далее.

Итеративный метод, напротив, проходит по всем необходимым значениям один раз, сохраняя только последние два значения для вычисления следующего, что значительно уменьшает количество операций.

⇒ Рекурсивная реализация имеет экспоненциальную временную сложность $O(2^n)$ из-за дублирования вычислений, в то время как итеративный подход имеет линейную временную сложность $O(n)$, что делает его более эффективным для больших значений n .

3. Использование памяти:

Рекурсивная реализация использует стек для хранения состояния каждого вызова функции. Если вызвать `fib(n)` с большим n , это может привести к переполнению стека.

Итеративный подход использует фиксированное количество переменных для хранения промежуточных результатов (в данном случае только две переменные), что делает его более эффективным с точки зрения использования памяти.

Задание №3. Числа Фибоначчи: последняя цифра.

Разработать эффективный алгоритм для подсчета последней цифры чисел Фибоначчи.

```
def fib(n):
    last1, last2 = 0, 1
    for _ in range(n):
        last1, last2 = last2, (last1 + last2) % 10

    return last1

import time
time_start = time.perf_counter()

with open('input.txt') as file:
    n = int(file.readline())

if 0 <= n <= 10 ** 7:
    with open('output.txt', 'w+') as file:
        file.write(str(fib(n)))

print(f'TIME {time.perf_counter() - time_start} microsec.')
```

```
else:  
    print('Incorrect numbers. Try again.')
```

Этот код выполняет следующие действия:

1. Определение функции: аналогично заданию №2.

Переменные last1 и last2 инициализируются значениями 0 и 1, что соответствует последним цифрам первых чисел Фибоначчи.

В цикле на каждой итерации происходит обновление переменных: новое значение для last1 становится текущим значением last2, а для last2 — суммой двух предыдущих значений по модулю 10. Это и есть шаг вычисления последней цифры числа Фибоначчи, т.к. $(a + b) \% 10 = a \% 10 + b \% 10$

2. Измерение времени выполнения: аналогично заданию №2.

3. Чтение входных данных: аналогично заданию №2.

4. Проверка диапазона: аналогично заданию №2.

5. Запись результата: аналогично заданию №2.

6. Измерение времени выполнения: аналогично заданию №2.

```
TIME 0.0005067999882157892 microsec.  
Для продолжения нажмите любую клавишу . . .
```

input.txt – Блокнот

Файл Правка Формат Вид Справка

0

output.txt – Блокнот

Файл Правка Формат Вид Справка

0

```
TIME 0.0009304999839514494 microsec.  
Для продолжения нажмите любую клавишу . . .
```

input.txt – Блокнот

Файл Правка Формат Вид Справка

331

output.txt – Блокнот

Файл Правка Формат Вид Справка

9

```
TIME 0.02078260001144372 microsec.  
Для продолжения нажмите любую клавишу . . .
```

input.txt – Блокнот

Файл Правка Формат Вид Справка

327305

output.txt – Блокнот

Файл Правка Формат Вид Справка

5

```
TIME 0.6053433000051882 microsec.
```

```
Для продолжения нажмите любую клавишу . . .
```

input.txt – Блокнот

Файл Правка Формат Вид Справка

10000000

output.txt – Блокнот

Файл Правка Формат Вид Справка

5

	Время работы (микросекунды)
Нижняя граница диапазона значений входных данных из текста задачи ($n = 0$)	0.0005
Пример 1 из задачи ($n = 331$)	0.00093
Пример 2 из задачи ($n = 327305$)	0.02
Верхняя граница диапазона значений входных данных из текста задачи ($n = 10^7$)	0.6

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.

В процессе написания кода, я вспомнила принципы работы с модулем `time`.

Вывод

Лабораторная 0 позволяет вспомнить, как осуществлять ввод-вывод в консоль и текстовые файлы, работать с модулем `time` для анализа времени работы алгоритма и писать алгоритмы для рекурсивного и итеративного нахождения n -ого числа Фибоначчи.