МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное образовательное учреждение высшего образования «Санкт-петербургский государственный морской технический университет»

ФАКУЛЬТЕТ ЦИФРОВЫХ ПРОМЫШЛЕННЫХ ТЕХНОЛОГИЙ

Кафедра Киберфизических систем

ЛАБОРАТОРНАЯ РАБОТА

По дисциплине «Программирование»

Выполнил: Гузненков Владислав Сергеевич группа 20121

Проверил: Поделенюк Павел Петрович

Санкт-Петербург

2024

Оглавление

[1. Цели и формулировка задачи 3](#_Toc187676633)

[2. Результаты работы 4](#_Toc187676634)

[2.1. Реализация программы с использованием функционального программирования языка Python 4](#_Toc187676635)

[2.1.1. Ход работы 4](#_Toc187676636)

[2.1.2. Демонстрация работы программы 5](#_Toc187676637)

[2.1.3. Листинг кода 5](#_Toc187676638)

[ЗАКЛЮЧЕНИЕ 7](#_Toc187676639)

[Список использованных источников 8](#_Toc187676640)

# Цели и формулировка задачи

Цель работы: Научиться работать со следующими темами:

1. Работа с файлами,
2. Работа с несколькими функциями, а не только с main,
3. Реализации рекурсивной функции.

Формулировка задачи: Даны **N** целых чисел **X1, X2, …, XN**. Требуется расставить между ними знаки **+** и **-** так, чтобы значение получившегося выражения было равно заданному целому **S**.

Входные данные: считываются с файла через пробел, в следующем порядке:

N X1 X2 ... XN S  
**2 <= N <= 30**, **0 <= Xi <= 5\*107**, **-109 <= S <= 109**

Необходимо вывести в файл полученное равенство с расставленными знаками, либо **no solution**, если не найдено решения. Если найдено несколько решений, то необходимо вывести **любое**.

Пример:

Input:

3 2 3 4 3

Output:

2-3+4=3

# Результаты работы

## **Реализация программы с использованием функционального программирования языка Python**

Основная идея алгоритма была в том, чтобы программа искала нужное уравнение методом проб и ошибок с помощью рекурсивной функции, постоянно заходя в новую функцию, подставляя «+» или «-», таким образом, перебираются все возможные уравнения и их результаты, в итоге выводится то, которое дает нужную нам сумму в виде строчного формата данных.

## Ход работы

Функция find\_sum: эта функция работает рекурсивно и принимает на вход:

* исходные числа, результат, который нужно получить,
* порядковый номер числа, с которым работает функция (0, если не задано),
* текущая строка, состоящая из чисел и «+» и «-» между ними (пустая строка, если не задана),
* текущее значение выражения (0, если не задано).

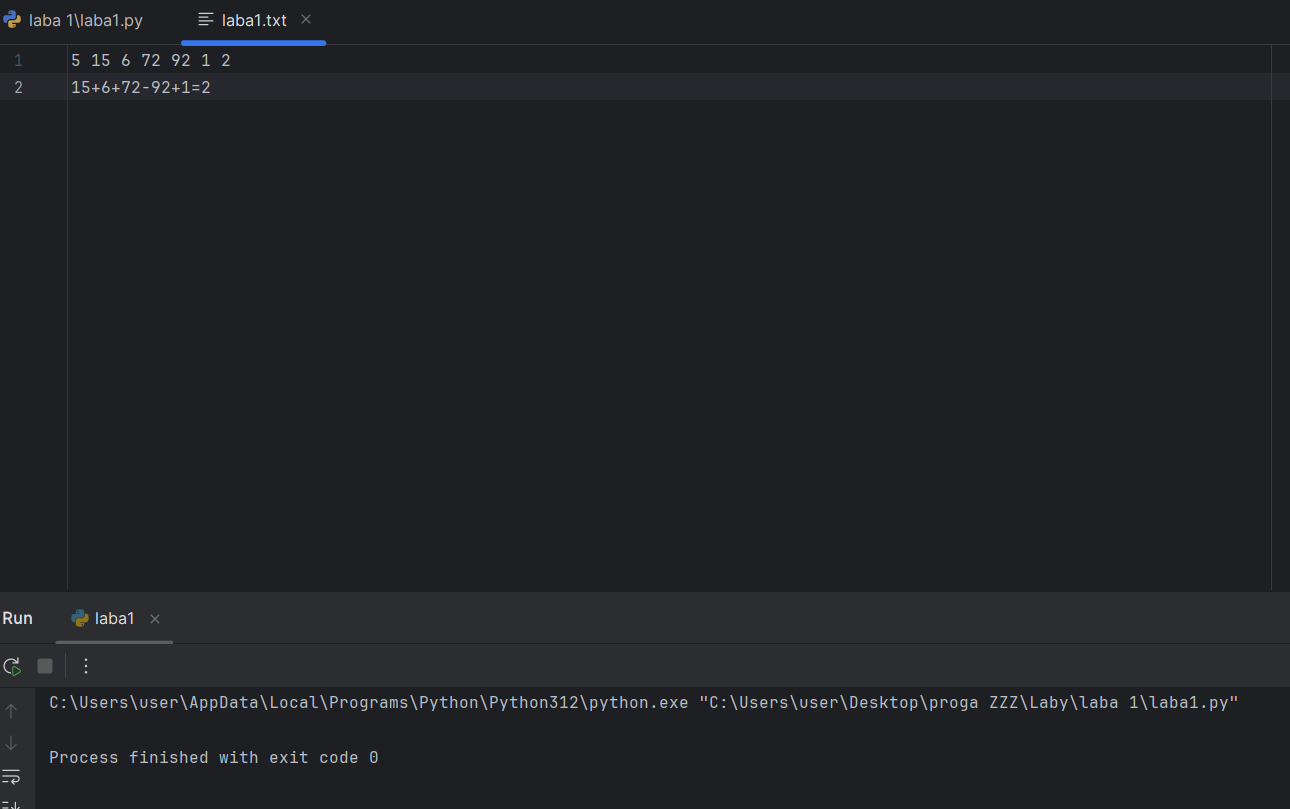
В этой функции выход из рекурсии реализован в виде условия: если конечное значение выражения равно нужному нам, то функция возвращает строку с «+» и «-», а если не равно, то возвращает None.

Далее, в коде изображено два случая: рекурсивный вход с «+числом» и рекурсивный вход с «-числом»

После функции идет работа с файлом. Сначала берутся исходные данные с файла, преобразовываются в количество чисел, список из этих чисел и итоговую сумму. Потом запускается рекурсия и по её итогу либо выводится «no solution», либо строка с «+» и «-» + «=нужная сумма».

## Демонстрация работы программы

Примерно так выглядит итог работы программы (выражение верно):



## Листинг кода

1. def find\_sum(numbers: list, target: int, index: int = 0, cur\_str: str = '', cur\_sum: int = 0) -> str or None:  
    *"""  
    Рекурсивная функция, которая выводит правильное равентсво через + или -.  
    :param numbers: Числа для равенства  
    :param target: нужное значение равенства  
    :param index: порядковый номер числа, с которым работает рекурсия.  
    :param cur\_str: Текущее состояние равенства  
    :param cur\_sum: текущее значение равенства  
    :return: выводит результат который равен либо строке, либо ничему (None)  
    """* if index == len(numbers): # выход из рекурсии  
    if cur\_sum == target:  
    return cur\_str # если нашлось решение  
    return None # если не нашлось решение  
     
    # рекурсия со знаком +  
    str\_pos = find\_sum(numbers, target, index + 1, cur\_str + f"+{numbers[index]}", cur\_sum + numbers[index])  
     
    if str\_pos:  
    return str\_pos  
     
    # рекурсия со знаком -  
    str\_neg = find\_sum(numbers, target, index + 1, cur\_str + f"-{numbers[index]}", cur\_sum - numbers[index])  
     
    if str\_neg:  
    return str\_neg  
     
    return None  
     
     
   with open("laba1.txt", "r") as f: # открываем файл и считываем с него информацию  
    massive = list(map(int, f.readline().split(" ")))  
     
   n = massive[0]  
   numbers = massive[1:-1]  
   s = massive[-1]  
     
   stroka = find\_sum(numbers, s) # запускаем функцию  
     
   output = '' # то что будет записываться в файл  
   if stroka and stroka[0] != "-":  
    res = f"{stroka}={s}"  
    if res[0] == "+": # если самый первый знак "+", то убираем его и записываем без него  
    output = res[1:]  
   else:  
    output = "no solution" # если функция вывела None  
   with open('laba1.txt', 'a') as f: # записываем ответ в исходный файл, не удаляя из него содержимое  
    f.write('\n' + output)

# ЗАКЛЮЧЕНИЕ

Были приобретены важные навыки работы с рекурсивными алгоритмами, которые действуют по методу проб и ошибок, и работы с файлами, а точнее запись и прочтение информации с файла.

# Список использованных источников

<https://pythonru.com/osnovy/rekursiya-python> - **Рекурсивная функция в python**

<https://pythonworld.ru/tipy-dannyx-v-python/fajly-rabota-s-fajlami.html> - **Файлы. Работа с файлами.**

<https://www.youtube.com/watch?v=jvFULnNpNLg> - **41 Рекурсия в Python. Рекурсивная функция Часть 1**

<https://www.youtube.com/watch?v=rzGCxtZdMuM> - **42 Рекурсия в Python. Рекурсивная функция Часть 2**

<https://pythonru.com/osnovy/globalnye-peremennye-python> **- Руководство по глобальным переменным**