Министерство науки и высшего образования Российской Федерации

(Минобрнауки)

Санкт-Петербургский Государственный Морской Технический Университет

(СПбГМТУ)

Работу выполнил

Студент СПбГМТУ

|  |
| --- |
| Кудрявцев Леонид Константинович |
| Подпись (Дата)  « » |

ОТЧЕТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Лабораторная работа

по теме:

«Рекурсивные функции. Лабораторная работа №1»

Научную работу проверил

|  |
| --- |
| Поделенюк Павел Петрович |
| Подпись (Дата)  « » |

# Содержание

[Содержание 2](#_Toc124800337)

[Список исполнителей 3](#_Toc124800338)

[Цель работы 4](#_Toc124800339)

[Формулировка задания 4](#_Toc124800340)

[Ход работы 5](#_Toc124800341)

[Приложение А 6](#_Toc124800342)

[Листинг кода: 6](#_Toc124800343)

# Список исполнителей

|  |  |  |  |
| --- | --- | --- | --- |
| Студент СПбГМТУ | |  | | --- | |  |   Подпись, дата | Кудрявцев Леонид Константинович, единоличный исполнитель лабораторной работы |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Цель работы

Ключевые слова: РЕКУСРИЯ, ФАЙЛЫ, PYTHON, ФУНКЦИЯ, ДРЕВО РЕШЕНИЙ

* Базовая работа с файлами. Запись и чтение
* Создание функций вне тела main, главной программы
* Создать древовидную рекурсивную функцию

# Формулировка задания

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**. Решение, только через рекурсию!

Пример:

Input:

3 2 3 4 3

Output:

2-3+4=3

# Ход работы

Составные элементы:

* Файл ввода: input(lab1).txt
* Файл вывода: output(1lab).txt
* Основной скрипт на Python: First lab.py
  + Функция: main
  + Функция: rec

Функция **main** не принимает никаких данных и не возвращает, является основным телом программы

Функция **rec**(arr: list, c: int, n: int, sum: int, signs: list, sign: str) принимает на вход переменные: arr (список чисел между которыми нужно поставить знак), c (искомый результат выражения), n (счётчик глубины рекурсии), sum (текущая сумма), signs (список расставленных знаков), sign (знак текущей рекурсии)

Основная идея программы – древовидное решение, перебор каждого возможного варианта расположения знаков. Реализуется древо решению в моей программе посредствам рекурсивной функции **rec**(arr: list, c: int, n: int, sum: int, signs: list, sign: str), а именно последовательным вызовом функции с передачей ей знака «+» или «−».

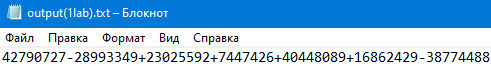
Функция углубляется, расставляя знаки и передавая получившуюся сумму в следующий вызов, когда же достигается базовый случай (количество расставленных знаков равняется: количеству чисел − 1) внутри функции происходит проверка равенства суммы (sum) и искомого числа (c), если проверка удачна функция возвращает накопленные ею на каждом уровне знаки

В главном теле скрипта, в функции **main** происходит считывание входных данных из файла input(lab1).txt, вызов рекурсивной функции согласно полученным данным и запись вывода в файле output(1lab).txt

Листинг скрипта: см. Приложение А

# Результат работы

По завершению работы программы в файл вывода в одну строку печатается получившиеся выражение с расставленными знаками + и – (см. )



Изображение экрана

Программа не выводит ничего в консоль по ходу своего выполнения

# Приложение А

Листинг кода:

|  |  |
| --- | --- |
|  | def rec(arr: list, c: int, n: int, sum: int, signs: list, sign: str): |
|  | **if** n == len(arr) - **1** **and** sum + arr[n] == c: # 1 из 3 базовых случаев |
|  | signs[-**1**] = "+" |
|  | **return** signs |
|  | **elif** n == len(arr) - **1** **and** sum - arr[n] == c: # 2 из 3 базовых случаев |
|  | signs[-**1**] = "-" |
|  | **return** signs |
|  | **elif** n == len(arr) - **1**: # 3 из 3 базовых случаев |
|  | signs[-**1**] = "\*" # \* будет знаком ложности |
|  | **return** signs |
|  | **else**: |
|  | bf = rec(arr, c, n+**1**, sum + arr[n], signs, "+") # сумма с плюсом |
|  | **if** bf[n+**1**] != "\*": # проверяет получилась ли сумма если знак + |
|  | signs = bf |
|  | signs[n] = "+" |
|  | **return** signs # выводим список знаков делая текущий плюсом |
|  | **else**: |
|  | bf = rec(arr, c, n+**1**, sum - arr[n], signs, "-") # если знак - |
|  | **if** bf[n+**1**] != "\*": |
|  | signs = bf |
|  | signs[n] = "-" |
|  | **return** signs # выводим список знаков делая текущий минусом |
|  | **else**: # если стоял знак ложности(не получилась сумма) |
|  | signs = bf |
|  | signs[n] = "\*" |
|  | **return** signs # выводим список знаков делая текущий знаком ложности |
|  | **def** **main**(): |
|  | files = open("input(lab1).txt", "r", encoding="utf-8") # Файл с входящими данными |
|  |  |
|  | sub = files.read().split(" ") # Строка с данными |
|  |  |
|  | n = int(sub[**0**]) # Кол-во чисел |
|  | s = int(sub[-**1**]) # Целевое число |
|  | sub = sub[**1**:-**1**] # Выделяем ряд чисел на вход |
|  | sim = [int(e) **for** e **in** sub] # Вносим все числа в список в формате Int |
|  |  |
|  | files.close() # Обязательно |
|  |  |
|  | zn = ["" **for** \_ **in** range(n)] |
|  | zn = rec(sim, s, **1**, sim[**0**], zn, "") # на выходе получается список со знаками |
|  |  |
|  | otv = "" |
|  | **if** "\*" **in** zn: |
|  | otv = "no solution" # если список забит знаками ложности - no solution |
|  | **else**: # иначе в переменную ответа записываем знак и его число |
|  | **for** i **in** range(n): |
|  | otv += zn[i] |
|  | otv += str(sim[i]) |
|  | otv += "=%i" %s |
|  |  |
|  | # запись в файл |
|  | output = open("output(1lab).txt", "w", encoding="utf-8") |
|  | output.write(otv) |
|  | output.close() # обязательно |
|  |  |
|  | **if** \_\_name\_\_ == "\_\_main\_\_": |
|  | main() |