**Практическая работа №9 Рекурсивные функции**

*Боев., ИСП.23.1А*

Теория

[Рекурсия в программировании: понятие, суть, примеры - плюсы и минусы рекурсивных функций и алгоритмов](https://practicum.yandex.ru/blog/rekursiya-v-programmirovanii/?utm_source=yandex&utm_medium=cpc&utm_campaign=Yan_Perfmax_RF_Prog_Resk_b2c_Gener_Regular_Blog_tCPA-direct-boos_460&utm_content=sty_search:s_none:cid_118128484:gid_5541029364:pid_54290267715:aid_16844097890:crid_0:rid_54290267715:p_1:pty_premium:mty_:mkw_:dty_desktop:cgcid_0:rn_%D0%9C%D0%BE%D1%81%D0%BA%D0%B2%D0%B0%20%D0%B8%20%D0%9C%D0%BE%D1%81%D0%BA%D0%BE%D0%B2%D1%81%D0%BA%D0%B0%D1%8F%20%D0%BE%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D1%8C:rid_1&utm_term=%D1%80%D0%B5%D0%BA%D1%83%D1%80%D1%81%D0%B8%D1%8F&yclid=16399711170202697727)

Задание

1. Определить исходные данные и результат.
2. Написать программу для реализации рекурсивной функции по своему варианту.

def F(n):  
 if n == 1:  
 return 1  
 elif n % 2 == 0:  
 return n + F(n - 1)  
 elif n > 1 and n % 2 != 0:  
 return 2 \* F(n - 2)  
  
num = float(input())  
result = F(num)  
print(f"F({num}) =", result)

1. Протестировать на различных исходных данных, заполнить таблицу:

|  |  |  |
| --- | --- | --- |
| № теста | Исходные данные n | Результат |
| 1 | 1 | 1 |
| 2 | 3 | 2 |
| 3 | 2 | 3 |
| 4 | 7 | 8 |
| 5 | 8 | 16 |
| 6 | 14 | 78 |
| 7 | 56 | 134217784 |
| 8 | 124 | 2305843009213694076 |
| 9 | 345 | 5986310706507378352962293074805895248510699696029696 |
| 10 | 999 | 1636695303948070935006594848413799576108321023021532  3947416456840480668982023372774416350461629520785754  43342063780035504608628272942696526664263794688 |

1. Ответить на вопрос задачи.

C:\Users\vladi\PycharmProjects\pythonProject\.venv\Scripts\python.exe "C:\Users\vladi\Desktop\ДЗ на 11.03.2025\Task.py"

26

**F(26) = 4122**

Process finished with exit code 0

1. Найти и описать пример задачи на рекурсию с собеседований на вакансию программиста.

**1. Поиск подходящих слагаемых**

**Условие:**  
Дан список чисел и число k. Нужно определить, можно ли выбрать два числа из списка так, чтобы их сумма равнялась k.

**Пример:**

numbers = [10, 15, 3, 7]

k = 17

# Ответ: True (10 + 7 = 17)

**Решение:**

def find\_sum(numbers, k):

if len(numbers) < 2:

return False

elif numbers[0] + numbers[-1] == k:

return True

else:

return find\_sum(numbers[1:], k) or find\_sum(numbers[:-1], k)

**2. Задача о Ханойской башне**

**Условие:**  
Необходимо перенести n дисков с одного стержня на другой, используя третий стержень как вспомогательный. За один раз можно перенести только один диск, при этом нельзя класть больший диск на меньший.

**Решение:**

def hanoi(n, source, target, auxiliary):

if n == 1:

print(f"Переместить диск 1 с {source} на {target}")

return

hanoi(n-1, source, auxiliary, target)

print(f"Переместить диск {n} с {source} на {target}")

hanoi(n-1, auxiliary, target, source)

**3. Генерация всех возможных комбинаций скобок**

**Условие:**  
Сгенерировать все правильные комбинации n пар скобок.

**Пример:**

n = 3

# Ответ: ["((()))", "(()())", "(())()", "()(())", "()()()"]

**Решение:**

def generate\_parentheses(n):

def backtrack(s='', left=0, right=0):

if len(s) == 2 \* n:

result.append(s)

return

if left < n:

backtrack(s+'(', left+1, right)

if right < left:

backtrack(s+')', left, right+1)

result = []

backtrack()

return result

**4. Задача о коммивояжере**

**Условие:**  
Найти кратчайший путь, проходящий через все города по одному разу с возвращением в исходную точку.

**Решение:**

def tsp(graph, start, current\_path=[]):

if len(current\_path) == len(graph):

return current\_path + [start]

shortest\_path = None

for city in graph:

if city not in current\_path:

path = tsp(graph, start, current\_path + [city])

if path:

if not shortest\_path or sum(graph[path[i]][path[i+1]] for i in range(len(path)-1)) < sum(graph[shortest\_path[i]][shortest\_path[i+1]] for i in range(len(shortest\_path)-1)):

shortest\_path = path

return shortest\_path

**5. Задача о расстановке ферзей**

**Условие:**  
Расставить n ферзей на шахматной доске n×n так, чтобы ни один ферзь не находился под ударом другого.

**Решение:**

def solve\_n\_queens(n):

def can\_place(pos, ocuppied\_columns):

for i in range(len(ocuppied\_columns)):

if pos == ocuppied\_columns[i] or \

pos == ocuppied\_columns[i] + (len(ocuppied\_columns) - i) or \

pos == ocuppied\_columns[i] - (len(ocuppied\_columns) - i):

return False

return True

def put\_queen(ocuppied\_columns, target\_row):

if target\_row == n:

return [ocuppied\_columns]

result = []

for column in range(n):

if can\_place(column, ocuppied\_columns):

result += put\_queen(ocuppied\_columns + [column], target\_row + 1)

return result

1. Оформить отчет по выполненной работе.

Варианты

|  |  |
| --- | --- |
| № | Задача |
| 1 |  |