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

Курсовая работа

по теме:

«Рекурсия. Лабораторная работа №2»

Руководитель научной работы

|  |
| --- |
| Петров Петр Петрович |
| Подпись (Дата)  « » |

# Содержание

[Содержание 2](#_Toc125381650)

[Список исполнителей 3](#_Toc125381651)

[Цель работы 4](#_Toc125381652)

[Формулировка задания 4](#_Toc125381653)

[Ход работы 5](#_Toc125381654)

[Результат работы 6](#_Toc125381655)

[Приложение А 7](#_Toc125381656)

[Листинг кода: 7](#_Toc125381657)

# Список исполнителей

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

# Цель работы

Ключевые слова: РЕКУРСИЯ, ФАЙЛ, PYTHON, ШАХМАТЫ, ОПТИМИЗАЦИЯ

* Работа с текстовыми файлами
* Работа с локальными и глобальными переменными
* Работа с несколькими функциями, не только с main
* Нисходящий метод разработки программ
* Анализ ходов

# Формулировка задания

Дана квадратная шахматная доска размером N x N. На доске уже размещено K фигур. Фигуры размещены так, что находятся не под боем друг друга.

Необходимо расставить на доске еще L фигур так, чтобы никакая из фигур на доске не находилась под боем любой другой фигуры. **Необходимо найти все возможные решения.**

Вы работаете с фигурой: \*(Задание по вариантам)

Входные данные в файле input.txt. На первой строке файла записаны три числа: N L K (через пробел). Далее следует K строк, содержащих числа x и y (через пробел) - координаты уже стоящей на доске фигуры (фигуры стоят правильно). Координаты отсчитываются от 0 до N-1. 1 <= N <= 20.

**Ваш алгоритм должен быть оптимизирован, для работы с доской 20!!**

Выходные данные в файл output.txt. **На каждое найденное решение необходимо записать в файл одну строку**. Строка состоит из пар (x,y) - координаты фигур на доске. **В решение следует вывести координаты всех фигур, находящихся на доске.** Каждую фигуру необходимо записать в виде пары координат, разделенных запятой и обрамленных скобками. Координаты отсчитываются от 0 до N-1. Порядок, в котором фигуры перечислены в решении, не имеет значения. Если не было найдено ни одного решения, в файл необходимо записать no solutions.

Выходные данные на консоль — это доска N\*N, где фигура обозначается #, ее ходы обозначаются \*, а пустые клетки обозначаются 0

# Ход работы

Составные элементы:

* Файл ввода: input.txt
* Файл вывода: output.txt
* Основной скрипт на Python: Вторая лаба.py
  + Функция **main**(out = output)
  + Функция **ultra\_mega\_king\_dragon\_figure**(x, y, desk)
  + Функция **rec**(desk: list, figur\_lst: list, n, x\_now, y\_now)

Функция **main** имеет один аргумент по умолчанию out – принимающий в себя значение глобальной переменной output – файла вывода. В теле этой функции происходят считывание файла ввода, преобразование данных в удобный для взаимодействия вид, расстановка фигур поставленных изначально, вызов рекурсивной функции и последующая запись ответа.

Функция **ultra\_mega\_king\_dragon\_figure** принимает на вход x и y – , координаты по которым располагаются изначальные фигуры, а также desk – матрица (шахматная доска). Вывод функции является матрица с расположенными на ней фигурами (#) и ячейками атаки фигур (\*).

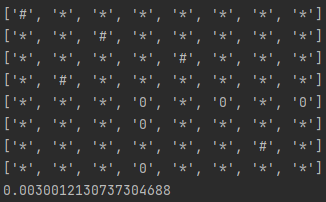
Функция **rec** является рекурсивной функцией в которой, углубляясь – я нахожу все возможные варианты расположения n-ой фигуры (где n – глубина рекурсии). На каждом уровне рекурсии функция проходит по абстрактному полю и проверяет ячейку на возможность постановки в неё фигуры (проходит по списку поставленных фигур с оператором if). Когда же достигнута нужная глубина рекурсии (n = 0 значит не осталось фигур для расстановки) происходит запись координат поставленных фигур в файл output.txt

Листинг скрипта: см. Приложение А

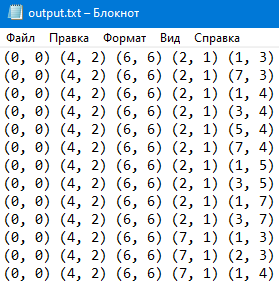
# Результат работы

По завершению работы программа выводит в консоль доску согласно первой строке файла output.txt и время выполнения или же ничего, если есть решения или его нет соответственно.

Пример вывода с решением: см. Изображение экрана 1 и Изображение экрана 2

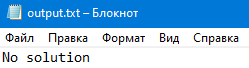


Изображение экрана



Изображение экрана

Пример вывода с отсутствующим решением: см. Изображение экрана 3



Изображение экрана

# Приложение А

Листинг кода:

|  |  |
| --- | --- |
|  | import copy |
|  | **import** **time** |
|  |  |
|  | output = open("output.txt", mode="w+", encoding="utf-8") # Глобальные переменные - файлы |
|  | inpt = open("input.txt", mode="r+", encoding="utf-8") |
|  | **def** **ultra\_mega\_king\_dragon\_figure**(x, y, desk): |
|  | # добавляю по вертикали |
|  | table = copy.deepcopy(desk) |
|  | **for** i **in** range(len(table)): |
|  | table[i][x] = "\*" |
|  | # добавляю по горизонтали |
|  | **for** i **in** range(len(table)): |
|  | table[y][i] = "\*" |
|  | # ячейка слева |
|  | **if** x > **0**: |
|  | table[y][x - **1**] = "\*" |
|  | # ячейка справа |
|  | **if** x < len(table) - **1**: |
|  | table[y][x + **1**] = "\*" |
|  | # ячейка сверху |
|  | **if** y > **0**: |
|  | table[y - **1**][x] = "\*" |
|  | # ячейка снизу |
|  | **if** y < len(table) - **1**: |
|  | table[y + **1**][x] = "\*" |
|  | # слева-сверху |
|  | **if** y > **0** **and** x > **0**: |
|  | table[y - **1**][x - **1**] = "\*" |
|  | # справа-сверху |
|  | **if** y > **0** **and** x < len(table) - **1**: |
|  | table[y - **1**][x + **1**] = "\*" |
|  | # слева-снизу |
|  | **if** y < len(table) - **1** **and** x > **0**: |
|  | table[y + **1**][x - **1**] = "\*" |
|  | # справа-снизу |
|  | **if** y < len(table) - **1** **and** x < len(table) - **1**: |
|  | table[y + **1**][x + **1**] = "\*" |
|  | table[y][x] = "#" |
|  | **return** table |
|  | **def** **rec**(desk: list, figur\_lst: list, n, x\_now, y\_now): |
|  | **if** n == **0**: |
|  | **print**(\*figur\_lst, file=output) |
|  | **return** |
|  | new\_figurs = list() |
|  | i = y\_now |
|  | j = x\_now |
|  | **while** True: |
|  | **if** i == len(desk) - **1** **and** j == len(desk) - **1**: |
|  | **break** |
|  | **if** j == len(desk) - **1**: |
|  | i += **1** |
|  | j = **0** |
|  | **else**: |
|  | j += **1** |
|  | ability = True |
|  | **for** e **in** figur\_lst: |
|  | **if** (j == e[**0**] **or** i == e[**1**]) **or** (abs(j - e[**0**]) <= **1** **and** abs(i - e[**1**]) <= **1**): |
|  | ability = False |
|  | **if** ability: |
|  | new\_figurs.append(tuple([j, i])) |
|  |  |
|  | """ for i in range(len(desk)): |
|  | for j in range(len(desk)): |
|  | ability = True |
|  | for e in figur\_lst: |
|  | if (j == e[0] or i == e[1]) or (abs(j - e[0]) <= 1 and abs(i - e[1]) <= 1): |
|  | ability = False |
|  | if ability: |
|  | new\_figurs.append(tuple([j, i]))""" |
|  |  |
|  | **for** e **in** new\_figurs: |
|  | rec(desk, figur\_lst + [e], n - **1**, e[**0**], e[**1**]) |
|  | **def** **main**(out = output): |
|  | start = time.time() |
|  |  |
|  |  |
|  | """for e in matrix: |
|  | print(e)""" |
|  |  |
|  | lines = inpt.readlines() # считали строки |
|  | lines = [e.strip("**\n**" ).split(" ") **for** e **in** lines] # Сформатировали строки |
|  |  |
|  | figures = [] |
|  | s = int(lines[**0**][**0**]) # размечикс |
|  | c = int(lines[**0**][**1**]) # скок надо поставить |
|  | **for** e **in** lines[**1**:]: |
|  | figures.append((int(e[**0**]), int(e[**1**]))) |
|  | matrix = [["0" **for** \_ **in** range(s)] **for** \_ **in** range(s)] |
|  |  |
|  | **for** e **in** figures: |
|  | matrix = ultra\_mega\_king\_dragon\_figure(e[**0**], e[**1**], matrix) |
|  |  |
|  | inpt.close() |
|  |  |
|  | rec(matrix, figures, c, -**1**, **0**) |
|  | out.close() |
|  | out = open("output.txt", mode="r+", encoding="utf-8") |
|  |  |
|  | lines = out.readlines() |
|  | **if** lines == []: |
|  | **print**("No solution", file=out) |
|  | out.close() |
|  | **else**: |
|  | out.close() |
|  | """ |
|  | output = open("output.txt", mode="w+", encoding="utf-8") |
|  | for i in range(len(lines)): |
|  | lines[i] = lines[i][:-2] |
|  | output.write(lines[i] + "\n") |
|  | """ |
|  | out = open("output.txt", mode="r+", encoding="utf-8") |
|  |  |
|  | lines = out.readline().replace("(", "").replace(")", "").replace(",", "").split(" ") # считали строки |
|  | #lines.pop(-1) |
|  | #print(lines) |
|  |  |
|  | matrix\_to = [["0" **for** \_ **in** range(s)] **for** \_ **in** range(s)] |
|  | **for** i **in** range(**0**, len(lines), **2**): |
|  | matrix\_to = ultra\_mega\_king\_dragon\_figure(int(lines[i]), int(lines[i+**1**]), matrix\_to) |
|  |  |
|  | out.close() |
|  |  |
|  | **for** e **in** matrix\_to: |
|  | **print**(e) |
|  |  |
|  | end = time.time() - start |
|  | **print**(end) |
|  | # ура победа... |
|  |  |
|  | **if** \_\_name\_\_ == "\_\_main\_\_": |
|  | main() |