Министерство образования и науки Российской Федерации

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

(СПбГМТУ)

Факультет Цифровых Промышленных Технологий

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2**

Выполнил:

Студент группы 20121

Лосев М.А

Поверил:

Поделенюк П.П.

Санкт-Петербург

2022 г.

**ЦЕЛЬ РАБОТЫ:**

1. Работа с текстовыми файлами

2. Работа с локальными и глобальными переменными

3.Работа с несколькими функциями, а не только с main

4.Нисходящий метод разработки программ

5.Анализ ходов

**ХОД РАБОТЫ:**

1.Открываем текстовый файл, считываем данные и закрываем его во избежание ошибок в дальнейшем.

2.Напишем функцию “count\_variants” для подсчета всех вариантов расстановки всех заданных фигур на доске

2.1.Напишем функцию “get\_new\_fig\_arr” , которая, используя функции “cell\_is\_good”, “cell\_is\_free”, “cell\_is\_attacked” заполняет доску всеми заданными фигурами, учитывая специфику фигуры биг Ромб

2.2 Напишем функцию “good\_move\_new\_figures”, которая возвращает все возможные варианты перестановки текущих фигур на доске

3.Записываем в файл "output.txt" ответ на каждое найденное решение.

**РЕЗУЛЬТАТЫ РАБОТЫ:**

**Входные данные(содержимое файла):**



**Результат работы программы:**



**Результат работы программы(output.txt):**



**ВЫВОД:**

Научились работать с текстовыми файлами, глобальными и локальными переменными.Научились использовать различные функции, а не только main.Изучили нисходящий метод разработки программ и анализ данных.

**ЛИСТИНГ:**

from datetime import datetime  
import time  
  
  
def cell\_is\_free(x: int, y: int, old\_fig\_arr: list, new\_fig\_arr: list) -> bool:  
 return (x, y) not in (old\_fig\_arr + new\_fig\_arr)  
  
  
def cell\_is\_attacked(x: int, y: int, old\_fig\_arr: list, new\_fig\_arr: list) -> bool:  
 attack\_mod\_delta\_x = (0, 1, 2, 3)  
 attack\_mod\_delta\_y = (0, 1, 2, 3)  
 l = []  
 for fig in (old\_fig\_arr + new\_fig\_arr):  
 flag = False  
 mod\_delta\_x = abs(fig[0] - x)  
 mod\_delta\_y = abs(fig[1] - y)  
 if (  
 mod\_delta\_x == 1 and mod\_delta\_y == 1) or (  
 mod\_delta\_x == 0 and mod\_delta\_y == 2) or (  
 mod\_delta\_x == 2 and mod\_delta\_y == 0) or (  
 mod\_delta\_x == 3 and mod\_delta\_y in [  
 1, 2, 3]) or (  
 mod\_delta\_x in [  
 1, 2, 3] and mod\_delta\_y == 3) or (  
 mod\_delta\_x == 2 and mod\_delta\_y == 2):  
 flag = False  
 elif (mod\_delta\_x in attack\_mod\_delta\_x  
 and mod\_delta\_y in attack\_mod\_delta\_y):  
 flag = True  
 l.append(flag)  
 if True in l:  
 return True  
 else:  
 return False  
  
  
def cell\_is\_good(x: int, y: int, old\_fig\_arr: list, new\_fig\_arr: list) -> bool:  
 return cell\_is\_free(x, y, old\_fig\_arr, new\_fig\_arr) \  
 and not cell\_is\_attacked(x, y, old\_fig\_arr, new\_fig\_arr)  
  
  
def get\_new\_fig\_arr(n: int, new\_fig\_count: int, old\_fig\_arr: list) -> list:  
 new\_fig\_arr = []  
  
 for y in range(n):  
 for x in range(n):  
 if cell\_is\_good(x, y, old\_fig\_arr, new\_fig\_arr):  
 new\_fig\_arr.append((x, y))  
  
 if len(new\_fig\_arr) == new\_fig\_count:  
 return new\_fig\_arr  
 return None  
  
  
def good\_move\_last\_new\_fig(n: int, old\_fig\_arr: list, new\_fig\_arr: list) -> bool:  
 last\_x, last\_y = new\_fig\_arr.pop()  
  
 for y in range(last\_y, n):  
 for x in range(n):  
 if y > last\_y or x > last\_x:  
 if cell\_is\_good(x, y, old\_fig\_arr, new\_fig\_arr):  
 new\_fig\_arr.append((x, y))  
 return True  
 return False  
  
  
def good\_complete\_new\_fig\_arr(n: int, new\_fig\_count: int, old\_fig\_arr: list, new\_fig\_arr: list) -> bool:  
 last\_x, last\_y = new\_fig\_arr[-1]  
 added\_fig\_arr = []  
  
 for y in range(last\_y, n):  
 for x in range(n):  
 if y > last\_y or x > last\_x:  
 if cell\_is\_good(  
 x,  
 y,  
 old\_fig\_arr,  
 new\_fig\_arr + added\_fig\_arr):  
 added\_fig\_arr.append((x, y))  
 if len(new\_fig\_arr + added\_fig\_arr) == new\_fig\_count:  
 new\_fig\_arr += added\_fig\_arr  
 # print(new\_fig\_arr)  
 return True  
 return False  
  
  
def good\_move\_new\_figures(n: int, new\_fig\_count: int, old\_fig\_arr: list, new\_fig\_arr: list) -> bool:  
 while True:  
 while not good\_move\_last\_new\_fig(n, old\_fig\_arr, new\_fig\_arr):  
 if len(new\_fig\_arr) == 0:  
 return False  
  
 if len(new\_fig\_arr) == new\_fig\_count or good\_complete\_new\_fig\_arr(  
 n, new\_fig\_count, old\_fig\_arr, new\_fig\_arr):  
 return True  
  
  
def print\_variant(n: int, old\_fig\_arr: list, new\_fig\_arr: list) -> None:  
 for y in range(n):  
 s = ""  
 for x in range(n):  
 if (x, y) in old\_fig\_arr:  
 s += 'O '  
 elif (x, y) in new\_fig\_arr:  
 s += 'n '  
 elif cell\_is\_attacked(x, y, old\_fig\_arr, new\_fig\_arr):  
 s += '# '  
 else:  
 s += '- '  
 print(s)  
 print('\n')  
  
def count\_variants(n: int, new\_fig\_count: int, old\_fig\_arr: list) -> int:  
 new\_list = []  
 res = 0  
 new\_fig\_arr = get\_new\_fig\_arr(n, new\_fig\_count, old\_fig\_arr)  
  
 if new\_fig\_count == 0:  
 print\_variant(n, old\_fig\_arr, old\_fig\_arr)  
 res += 1  
  
 if new\_fig\_arr:  
 if res == 0:  
 print\_variant(n, old\_fig\_arr, new\_fig\_arr)  
 res += 1  
 for line in (old\_fig\_arr + new\_fig\_arr):  
 new\_list.append('(' + str(line[0]) + ',' + str(line[1]) + '),')  
 new\_list.append('\n')  
 while good\_move\_new\_figures(n, new\_fig\_count, old\_fig\_arr, new\_fig\_arr):  
 res += 1  
 if new\_fig\_arr == [(3, 2), (4, 4)]:  
 print\_variant(n, old\_fig\_arr, new\_fig\_arr)  
 for line in (old\_fig\_arr + new\_fig\_arr):  
 new\_list.append('(' + str(line[0]) + ', ' + str(line[1]) + ') ')  
 new\_list.append('\n')  
 new\_list[-1] = new\_list[-1][0:-2]  
 if res == 0:  
 new\_list.append('No solution')  
 with open(r"output.txt", "w") as file:  
 for line in (new\_list):  
 file.write(line)  
 file.close()  
 return res  
  
  
'''Получаем переменные из файла'''  
  
  
def get\_data\_from\_file(filename: str) -> tuple:  
 flag = True  
 old\_fig\_arr = []  
 with open(filename, "r") as file1:  
 for line in file1:  
  
 if flag:  
 n = int(line.strip().split(' ')[0])  
 l = int(line.strip().split(' ')[1])  
 k = int(line.strip().split(' ')[2])  
 flag = False  
 elif flag == False:  
 a = line.strip().split(' ')  
 a = [int(item) for item in a]  
 old\_fig\_arr.append(tuple(a))  
 file1.close()  
 return n, l, k, old\_fig\_arr  
  
  
def main():  
 n, l, k, old\_fig\_arr = get\_data\_from\_file("p2.txt")  
 # print(n, l, k, old\_fig\_arr)  
  
 print("Размер доски: ", n, " ,Нужно поставить фигур: ", l, " ,Уже стоят фигур: ", len(old\_fig\_arr), "\n",)  
 start\_time = time.time()  
 print()  
 res = count\_variants(n, l, old\_fig\_arr)  
 print("Количество решений: ", res)  
  
 print("Время работы: ", (time.time() - start\_time))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()