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 №2

По дисциплине «Программирование»

Выполнил: Зорин П. С.

Проверил:

Санкт-Петербург

2024

* **Цели и формулировка задачи**
* Работа с текстовыми файлами,
* Работа с локальными и глобальными переменными,
* Работа с несколькими функциями, а не только с main,
* Нисходящий метод разработки программ,
* Анализ ходов.
* **Результаты работы**



Файл output.txt, в котором записины все координаты.



* **Ход работы**

Программа предназначена для размещения определенного числа верблюдов на шахматной доске размером N x N с учетом ограничений. Функция safe проверяет, находится ли заданная клетка под угрозой, учитывая возможные ходы верблюда. Если клетка свободна от угроз, алгоритм продолжает расстановку, используя функцию place\_verblud, которая рекурсивно размещает верблюдов, пока не закончатся оставшиеся фигуры или не будет достигнута заданная конфигурация.

Для отображения угроз от размещенных фигур, используется функция threats, которая создает новую доску, помечая клетки, находящиеся под угрозой. print\_board выводит текущее положение доски(где стоят фигуры и какие клетки под угрозой).

Входные данные:N - доска, L - кол-во фигур, которые надо разместить, K - Фигуры, которые уже находятся на доске, X и Y - координаты фигур, которые уже стоят на доске. В конце, все найденные решения записываются в файл output.txt, а общее число решений и доска выводится на экран.

* **Демонстрация работы программы**

Ввод данных:



Результат:





* **Листинг кода**

def safe(board, x, y, N):  
 # Находиться ли клетка под угрозой  
 verblud\_moves = [  
 (3, 1), (3, -1), (-3, 1), (-3, -1),  
 (1, 3), (1, -3), (-1, 3), (-1, -3)  
 ]  
  
 for dx, dy in verblud\_moves:  
 if 0 <= x + dx < N and 0 <= y + dy < N and board[x + dx][y + dy] == 1:  
 return False  
 return True  
  
def place\_verblud(board, N, remaining, solutions, current\_solution):  
 if remaining == 0:  
 solutions.append(current\_solution.copy())  
 return  
  
 for i in range(N):  
 for j in range(N):  
 if board[i][j] == 0 and safe(board, i, j, N):  
 # Установить верблюда  
 board[i][j] = 1  
 current\_solution.append((i, j))  
  
 # Расстановка оставшихся верблюдов  
 place\_verblud(board, N, remaining - 1, solutions, current\_solution)  
  
 # Удалить верблюда  
 board[i][j] = 0  
 current\_solution.pop()  
  
def threats(board, N):  
 # Доска для угроз  
 threat\_board = [['0'] \* N for \_ in range(N)]  
  
 for i in range(N):  
 for j in range(N):  
 if board[i][j] == 1: # Если на доске есть фигура  
 threat\_moves = [  
 (3, 1), (3, -1), (-3, 1), (-3, -1),  
 (1, 3), (1, -3), (-1, 3), (-1, -3)  
 ]  
 for dx, dy in threat\_moves:  
 if 0 <= i + dx < N and 0 <= j + dy < N:  
 threat\_board[i + dx][j + dy] = '\*'  
  
 return threat\_board  
  
def print\_board(board, N):  
 threat\_board = threats(board, N)  
  
 for i in range(N):  
 row = ''  
 for j in range(N):  
 if board[i][j] == 1:  
 row += '# '  
 elif threat\_board[i][j] == '\*':  
 row += '\* '  
 else:  
 row += '0 '  
 print(row.strip())  
  
# Ввод данных  
N, L, K = map(int, input("N(размер доски), L(фигуры для размещения), K(фигуры на доске): ").split())  
  
board = [[0] \* N for \_ in range(N)]  
positions = []  
  
for \_ in range(K):  
 x, y = map(int, input("Координаты фигуры: ").split())  
 board[x][y] = 1  
 positions.append((x, y))  
  
solutions = []  
place\_verblud(board, N, L, solutions, positions)  
  
# Запись всех решений в файл output.txt  
with open('output.txt', 'w') as f:  
 if not solutions:  
 f.write("no solutions\n")  
 else:  
 for solution in solutions:  
 f.write(', '.join(f"({x},{y})" for x, y in solution) + '\n')  
  
# Вывод доски  
print\_board(board, N)  
print("Общее количество решений:", len(solutions))

**ЗАКЛЮЧЕНИЕ**

**Список использованных источников**