**Лабораторная работа №1**

**Цель:** При работе над данной лабораторной работой затронуть следующие вопросы:

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:**

**Задача**: 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: BishopTwoCellsFigure. Под боем у этой фигуры находятся 8 клеток (по 2 на каждой диагонали)

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 стоящих на доске фигуры (фигуры стоят правильно). Координаты отсчитываются от 0 до N-1. 1 <= N <= 20.

Алгоритм должен быть оптимизирован, для работы с доской 20

Выходные данные идёт 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 фигуры обозначаются #, клетки под её боем обозначаются \*, а пустые клетки обозначаются 0.

**Решение**:

def safe(board,row,col,N): # Проверяет, находится ли клетка под боем другой фигуры  
 for i in range(-2,3): # Клетки под боем находим в промежутке от -2 до +2, остальную доску брать в расчёт смысла нет  
 for j in range(-2,3):  
 if abs(i)==abs(j) and (i!=0 or j!=0): # Проверяем только диагональные клетки  
 r,c=row+i,col+j # Координаты клетки под боем  
 if 0<=r<N and 0<=c<N and board[r][c]==1: return False # Если клетка под боем  
 return True # Если клетка безопасна  
  
def place\_figures(board,figures,placed,L,N): # Размещаем фигуры на доске  
 if placed==L: return [figures.copy()] # Если все нужные фигуры размещены, возвращаем копию текущего расположения фигур  
 solutions=[] # Список для хранения возможных решений  
 for i in range(N): # Строки  
 for j in range(N): # Столбцы  
 if board[i][j]==0 and safe(board,i,j,N): # Проверка безопасности клетки  
 board[i][j]=1 # Устанавливаем фигуру  
 figures.append((i,j)) # Добавляем координаты в список  
 solutions.extend(place\_figures(board,figures,placed+1,L,N)) # Рекурсивно размещаем следующую фигуру  
 figures.pop() # Убираем последнюю добавленную фигуру  
 board[i][j]=0 # Убираем фигуру с клетки  
 return solutions # Все найденные решения  
  
def print\_board(board,N): # Выводит доску в консоль  
 print('Изначальная доска:')  
 for i in range(N): # Строки  
 for j in range(N): # Столбцы  
 if board[i][j]==1: print('#',end=' ') # Если клетка занята  
 elif board[i][j] == 2: print('\*',end=' ') # Если клетка под боем  
 else: print('0',end=' ') # Если клетка пуста  
 print() # Переход на новую строку  
  
with open('input.txt','r') as file:  
 N,L,K=map(int,file.readline().strip().split())  
 board=[[0]\*N for \_ in range(N)] # Создаем доску  
 current\_figures=[] # Список координат размещенных фигур  
 for \_ in range(K): # Непоставленные фигуры  
 x,y=map(int,file.readline().strip().split())  
 board[x][y]=1 # Отмечаем, что клетка занята  
 current\_figures.append((x,y))  
  
 for (x,y) in current\_figures: # Перебираем фигуры, идентифицируя клетки под боем  
 for i in range(-2,3): # Промежуток нахождения клеток под боем  
 for j in range(-2,3):  
 if abs(i)==abs(j) and (i!=0 or j!=0): # Проверяем только диагональные клетки  
 r,c=x+i,y+j # Координаты клетки под боем  
 if 0<=r<N and 0<=c<N: board[r][c]=2 # Метим клетку под боем  
  
 solutions=place\_figures(board,current\_figures,0,L,N) # Функция для размещения фигур  
 c=0 # Счётчик количества решений  
 with open('output.txt','w') as out\_file:  
 if not solutions: out\_file.write('no solutions\n') # Если решений нет  
 else:  
 for solution in solutions: # Перебираем все найденные решения  
 formatted\_solution = ', '.join(f'({x},{y})' for (x,y) in solution) # Форматируем решение  
 c+=1  
 out\_file.write(f'{formatted\_solution}\n') # Записываем решение в файл  
  
 print\_board(board, N) # Выводим доску в консоль  
 print('Всего решений:',c//(K+L))

**Пример:**

|  |  |
| --- | --- |
| Input:  20 0 1  8 8 | Output:  (8,8) |
| Input:  10 3 3  0 0  1 7  9 8 | Output:  Всего решений: 83863 |

**Вывод:** В ходе лабораторной работы затронули следующие 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.