Санкт-Петербургский государственный морской технический университет

**Лабораторная работа №1**

**за Ⅰ семестр**

Выполнил: Студен 1 курса Факультета Цифровых Промышленных Технологий группы 20121 Поташев Р.В.

Преподаватель: Поделенюк. П. П.  
Кафедра прикладной   
математики и математического моделирования

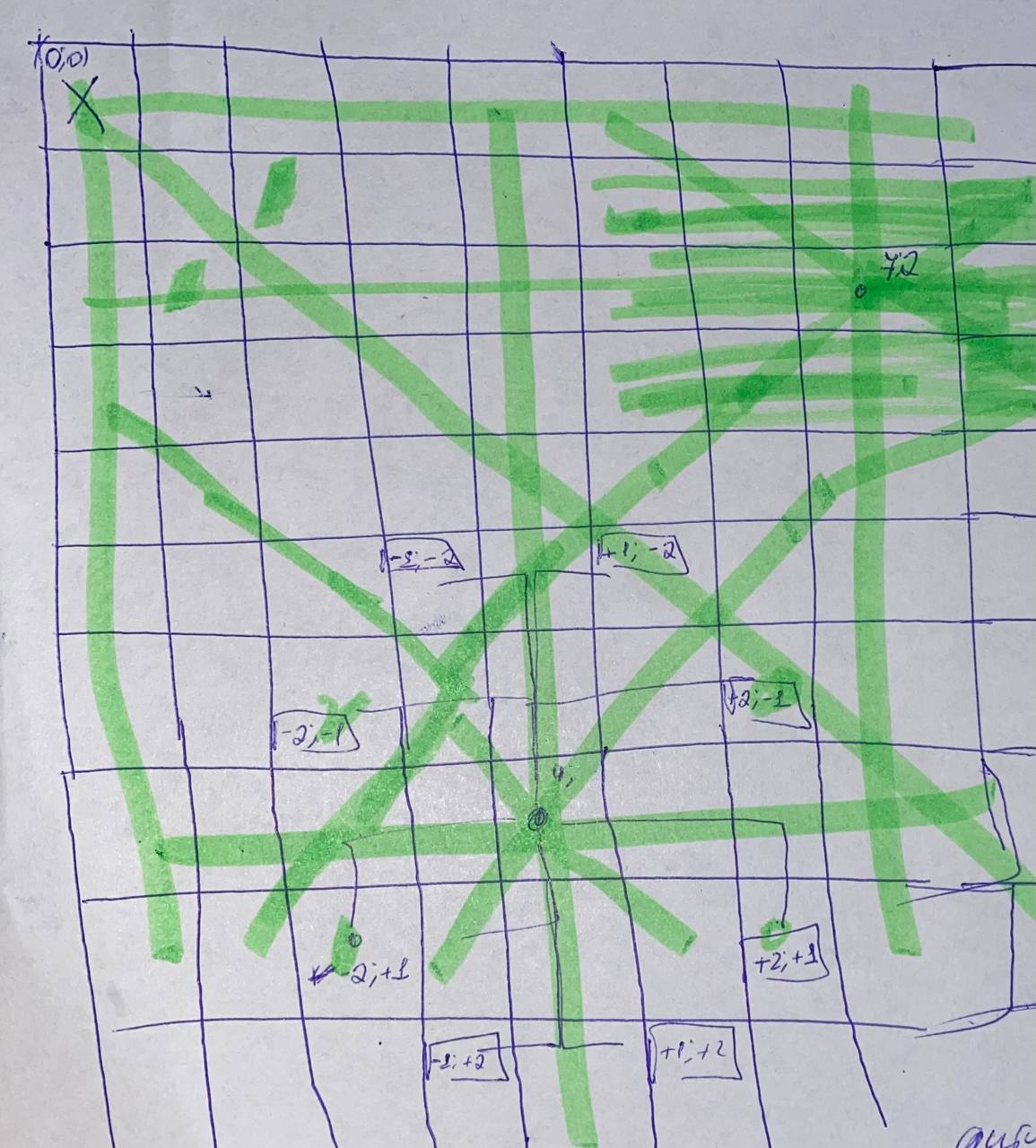
Санкт-Петербург   
2023

**Цель работы:**

Научиться работать с текстовыми файлами, локальными и глобальными переменными, с несколькими функциями и научиться использовать нисходящему методу разработки программ.

**Ход работы:**

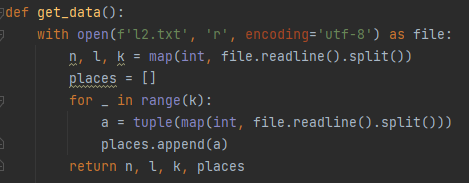
Сначала проанализируем возможные ходы фигур, расположенных в разных областях шахматной доски.



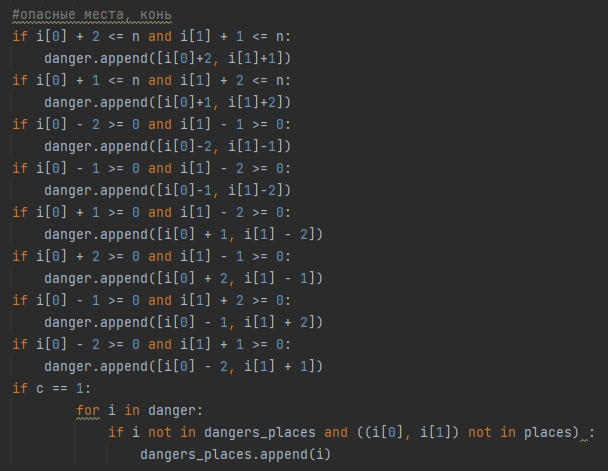
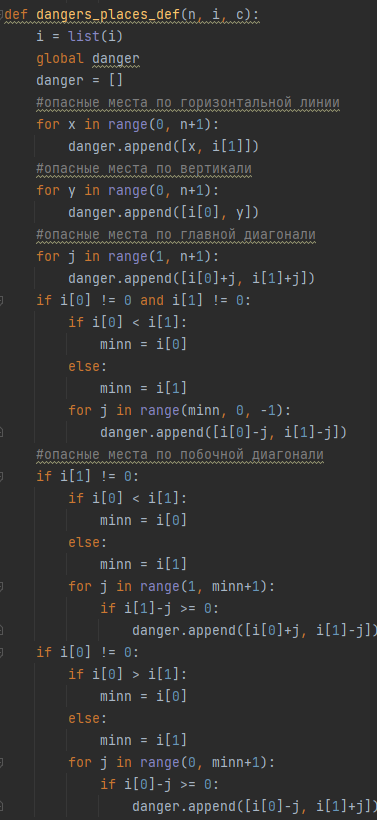
Сначала нам надо найти опасные ходы у уже расставленных фигур, затем надо найти список с свободными ходами. После чего перебираем список свободных ходов и пытаемся расставить нужное количество фигур. Для этого надо взять одну возможную фигуры, найти список опасных ходов для нее и проверить попадает ли уже расставленная фигура в этот список, если нет, то можем ставить эту фигуры на шахматную доску.

**Результат работы**

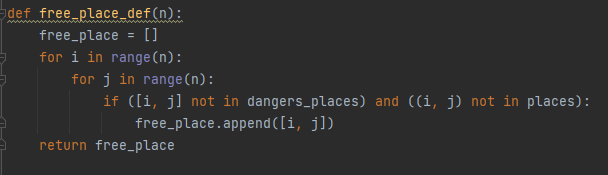
Создаем функцию “get\_data()”, считывающую данные с файла, в котором заданы начальные параметры.



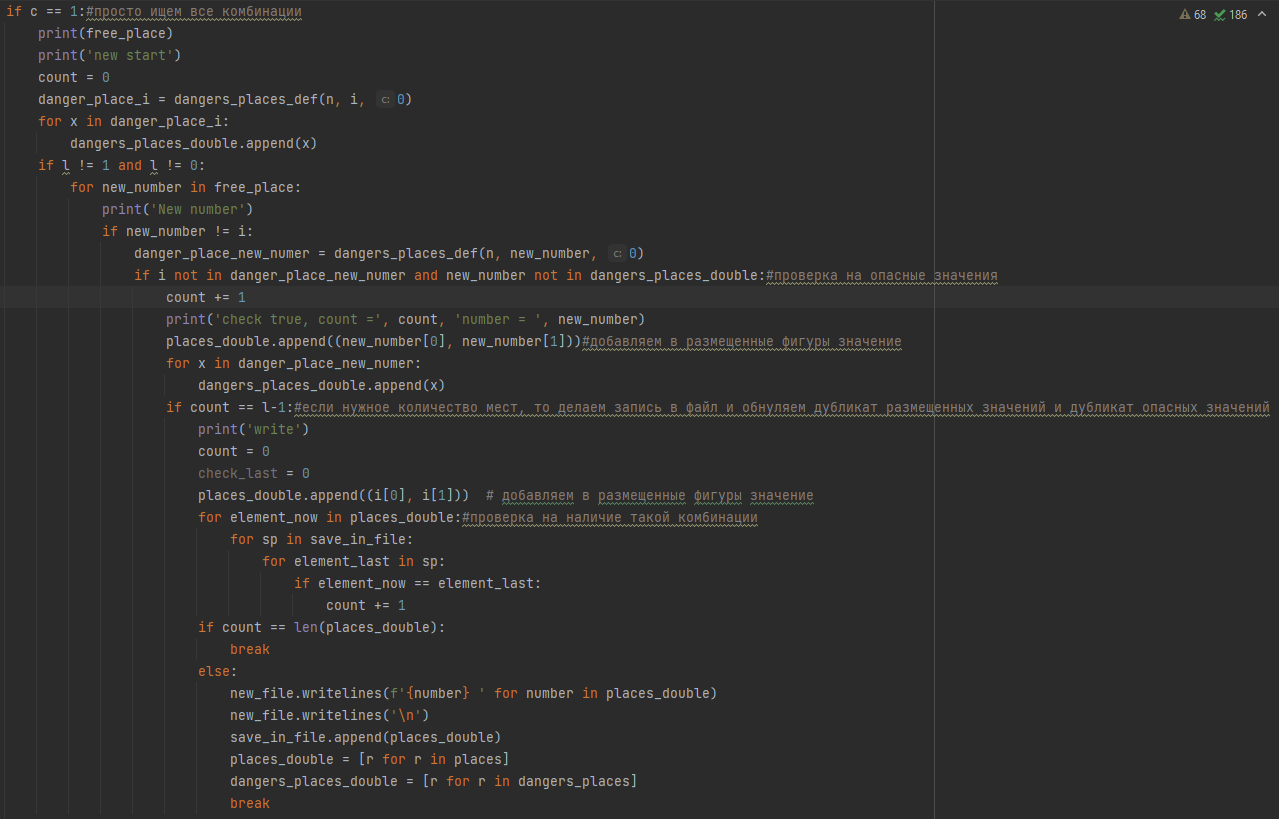
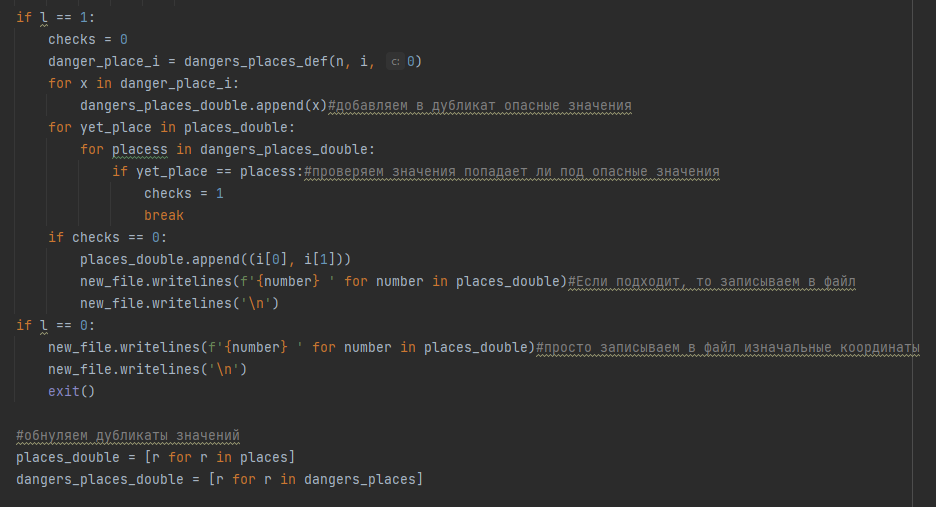
Затем проходимся циклом по списку с уже расставленными фигурами и передаем каждую координату в функцию «dangers\_places\_def(n, i, c)», которая принимает 3 параметра: n – размер доски, i – координату фигуры и c – сигнал для функции о необходимости добавлять найденные «опасные» ходы в общий список с ними.



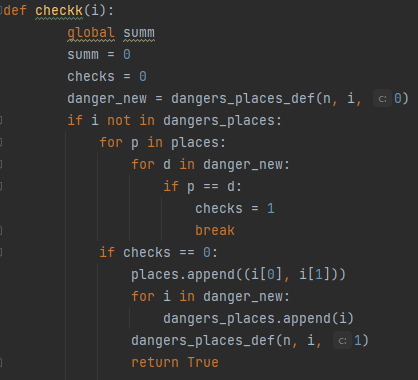
После чего методом исключения, находим свободные места на шахматной доске.



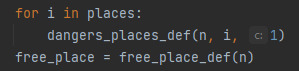
Затем проходимся по свободным значениям и запускаем функцию «check(i, 1)», которая ищет все возможные расстановки фигур. Сначала она вычисляет опасные ходы и записывает их в дубликат опасных ходов, а затем начинает подбирать нужное количество фигур, которые можно разместить по заданным условиям и проверяет на наличие уже составленной такой комбинации.



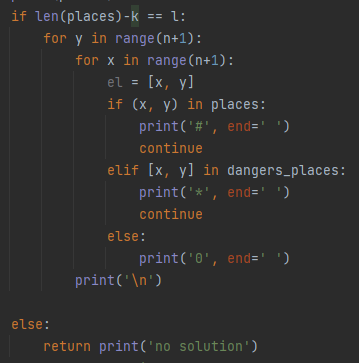
После этого проходимся по списку со свободными местами и передаем эти координаты в функцию «checkk(i, 0)»(c = 0 находим решение для вывода шахматной доски), которая вызывает функцию «dangers\_places\_def(n, i, c)» с параметром c = 0, то есть функция не должна записывать найденные опасные места для конкретной координаты в общий массив. То есть функция сначала находит «опасные» места, а потом проверяет входит ли координата в «опасный» список и попадает ли хотя бы одна из уже размещенных фигур в потенциально опасный список, то есть если разместить новую фигуру в данном месте. Если попадает, то значение переменной «checks» становится «1» и цикл завершается, если нет, то новая фигура добавляется на поле и список «опасных» ходов дополняется.



При этом функция запускается из функции «main()» и циклом, а если была добавлена фигура функция возвращает «True» и значение переменной «summ» увеличивается на 1.



Потом если было добавлено нужное количество фигур, то начинается перебор всех возможных значение на доске и если какое-то значение попадает в список с расставленными фигурами или «опасными» ходами, то выводится «#» или «\*», соответственно, если значение не попадает не в один из списков, то выводится «0». Если нужное количество новых фигур набрано не было, то выводится «no solution»



**Выводы**

Выполнив лабораторную работу я научился работать с текстовыми файлами, локальными и глобальными переменными, с несколькими функциями и научиться использовать нисходящему методу разработки программ.

def checkk(i, c): # на вход получает одно из значений координат из списка "не опасных" точек на шахматной доске

global dangers\_places, places, n, count, k, new\_file, dangers\_places\_double, places\_double, l, free\_place, save\_in\_file

if c != 1:#добавляем для выводы на шахматную доску

summ = 0

checks = 0

danger\_new = dangers\_places\_def(n, i, 0)

if i not in dangers\_places: # проверка входит ли данная координата в "опасный список"

for p in places: # проверка будет ли входить одна из уже размещенных фигур в новый "опасный" список

for d in danger\_new:

if p == d:

checks = 1 # если да, то checks = 1

break

if checks == 0: # если нет, то новая фигура, то новая фигура добавляется в список фигур и обновляется "опасный" список

places.append((i[0], i[1]))

dangers\_places\_def(n, i, 1)

return True

if c == 1:#просто ищем все комбинации

count = 0

danger\_place\_i = dangers\_places\_def(n, i, 0)

for x in danger\_place\_i:

dangers\_places\_double.append(x)

if l != 1 and l != 0:

for new\_number in free\_place:

if new\_number != i:

danger\_place\_new\_numer = dangers\_places\_def(n, new\_number, 0)

if i not in danger\_place\_new\_numer and new\_number not in dangers\_places\_double:#проверка на опасные значения

count += 1

places\_double.append((new\_number[0], new\_number[1]))#добавляем в размещенные фигуры значение

for x in danger\_place\_new\_numer:

dangers\_places\_double.append(x)

if count == l-1:#если нужное количество мест, то делаем запись в файл и обнуляем дубликат размещенных значений и дубликат опасных значений

count = 0

check\_last = 0

places\_double.append((i[0], i[1])) # добавляем в размещенные фигуры значение

for element\_now in places\_double:#проверка на наличие такой комбинации

for sp in save\_in\_file:

for element\_last in sp:

if element\_now == element\_last:

count += 1

if count == len(places\_double):

break

else:

new\_file.writelines(f'{number} ' for number in places\_double)

new\_file.writelines('\n')

save\_in\_file.append(places\_double)

places\_double = [r for r in places]

dangers\_places\_double = [r for r in dangers\_places]

break

if l == 1:

checks = 0

danger\_place\_i = dangers\_places\_def(n, i, 0)

for x in danger\_place\_i:

dangers\_places\_double.append(x)#добавляем в дубликат опасные значения

for yet\_place in places\_double:

for placess in dangers\_places\_double:

if yet\_place == placess:#проверяем значения попадает ли под опасные значения

checks = 1

break

if checks == 0:

places\_double.append((i[0], i[1]))

new\_file.writelines(f'{number} ' for number in places\_double)#Если подходит, то записываем в файл

new\_file.writelines('\n')

if l == 0:

new\_file.writelines(f'{number} ' for number in places\_double)#просто записываем в файл изначальные координаты

new\_file.writelines('\n')

exit()

#обнуляем дубликаты значений

places\_double = [r for r in places]

dangers\_places\_double = [r for r in dangers\_places]

def dangers\_places\_def(n, i, c): # поиск опасных "координат", переменная отвечает за добавление найденных опасных ходов в общий список. Если с = 1, то будут добавлены, если с = 0 - нет

# везде есть проверка на то, чтобы значения были в пределах доски

i = list(i)

global danger

danger = []

# опасные места по горизонтальной линии

for x in range(0, n + 1):

danger.append([x, i[1]])

# опасные места по вертикали

for y in range(0, n + 1):

danger.append([i[0], y])

# опасные места по главной диагонали

for j in range(1, n + 1):

danger.append([i[0] + j, i[1] + j])

if i[0] != 0 and i[1] != 0:

if i[0] < i[1]:

minn = i[0]

else:

minn = i[1]

for j in range(minn, 0, -1):

danger.append([i[0] - j, i[1] - j])

# опасные места по побочной диагонали

if i[1] != 0:

if i[0] < i[1]:

minn = i[0]

else:

minn = i[1]

for j in range(1, minn + 1):

if i[1] - j >= 0:

danger.append([i[0] + j, i[1] - j])

if i[0] != 0:

if i[0] > i[1]:

minn = i[0]

else:

minn = i[1]

for j in range(0, minn + 1):

if i[0] - j >= 0:

danger.append([i[0] - j, i[1] + j])

# опасные места, конь

if i[0] + 2 <= n and i[1] + 1 <= n:

danger.append([i[0] + 2, i[1] + 1])

if i[0] + 1 <= n and i[1] + 2 <= n:

danger.append([i[0] + 1, i[1] + 2])

if i[0] - 2 >= 0 and i[1] - 1 >= 0:

danger.append([i[0] - 2, i[1] - 1])

if i[0] - 1 >= 0 and i[1] - 2 >= 0:

danger.append([i[0] - 1, i[1] - 2])

if i[0] + 1 >= 0 and i[1] - 2 >= 0:

danger.append([i[0] + 1, i[1] - 2])

if i[0] + 2 >= 0 and i[1] - 1 >= 0:

danger.append([i[0] + 2, i[1] - 1])

if i[0] - 1 >= 0 and i[1] + 2 >= 0:

danger.append([i[0] - 1, i[1] + 2])

if i[0] - 2 >= 0 and i[1] + 1 >= 0:

danger.append([i[0] - 2, i[1] + 1])

if c == 1:

for i in danger:

if i not in dangers\_places and ((i[0], i[1]) not in places):

dangers\_places.append(i)

return danger

def free\_place\_def(n):

free\_place = []

for i in range(

n): # перебор всех координат на N-ой доске. Если координата не входит в "опасный" список и в список уже размещенных фигур, то координата добавляется в список свободных мест

for j in range(n):

if ([i, j] not in dangers\_places) and ((i, j) not in places):

free\_place.append([i, j])

return free\_place

def get\_data():

with open(f'input.txt', 'r', encoding='utf-8') as file:

n, l, k = map(int, file.readline().split())

places = []

for \_ in range(k):

a = tuple(map(int, file.readline().split()))

places.append(a)

return n, l, k, places

def main():

global dangers\_places, places, n, count, k, new\_file, dangers\_places\_double, places\_double, l, free\_place, save\_in\_file

n, l, k, places = get\_data()

summ = 0

save\_in\_file = []

dangers\_places = []

for i in places: # заполнение "опасного" списка

dangers\_places\_def(n, i, 1)

free\_place = free\_place\_def(n)

dangers\_places\_double = [i for i in dangers\_places]

places\_double= [i for i in places]

new\_file = open('output.txt', 'w+')

for i in free\_place:

checkk(i, 1)

new\_file.close()

for i in free\_place: # перебор координат из "безопасного" списка и поиск подходящих точек

if summ < l + 1:

if checkk(i, 0):

summ += 1

if summ == l:

break

print('Всего комбинаций: ', len(save\_in\_file))

if len(places) - k >= l:

for y in range(n + 1):

for x in range(n + 1):

el = [x, y]

if (x, y) in places: # если такая же координата есть в списке размещенных фигур, то выводится "#"

print('#', end=' ')

continue

elif [x,

y] in dangers\_places: # если такая же координата есть в списке опасных значений, то выводится "\*"

print('\*', end=' ')

continue

else: # если такой координаты нет, то выводится "0"

print('0', end=' ')

print('\n')

else:

return print('no solution')

if \_\_name\_\_ == '\_\_main\_\_':

main()