Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ МОРСКОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(СПБГМТУ)

|  |
| --- |
|  |

Отчёт по Лабораторной работе №2

|  |
| --- |
|  |

|  |
| --- |
| Выполнил студент группы 20121: Никитин Д В. |
| Проверил: Поделенюк П.П. |

|  |
| --- |
|  |

Санкт-Петербург  
2023год

# ЦЕЛИ РАБОТЫ

При работе над данной лабораторной работой будут затронуты 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

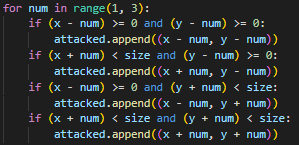
В самом начале из модуля time импортируем функцию time(), которая передаёт время, выраженное в секундах с начала эпохи.

Программа открывает файлы ‘input.txt’ (данные сохраняются в переменной input\_data) и ‘output.txt’ (данные сохраняются в переменной output) с помощью метода open(). Переменная file принимает значения из input\_data в виде списка. Далее вводим переменную firstboard, равную None, которая будет принимать позиции первых поставленных фигур. Переменным N, L и K передаются значения из первого элемента file. N – размер доски, L – количество фигур, которые надо поставить на доску и K – количество фигур, которые уже стоят на доске. Переменная standing\_figure сохраняет в виде кортежей (x, y) координаты из переменной file[1:].

Следующим шагом вызывается функция main(). Переменной atime передаётся функция time(). Далее вызываем функцию chess() в качестве аргументов которой передаём 0, 0, функцию putting\_given\_figure() с аргументами int(N) и standing\_figure, int(L) и standing\_figure.

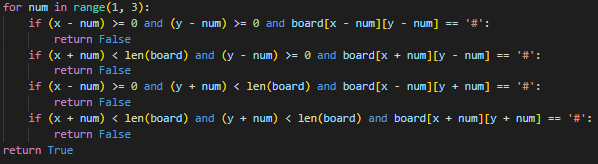
Сперва надо разобраться, что делает функция putting\_given\_figure(). В качестве аргументов принимает размер доски (size: int), координаты фигур (coordinates: list) и возвращает доску с расставленными фигурами. Переменная board создаёт матрицу N на N, состоящую из 0. Далее с помощь цикла for мы достаём x и y из coordinates и на место board[x][y] ставим ‘#’ (фигуру). С помощью цикла for достаём список cell из функции attacked\_cell(), которая принимает за аргументы x, y и size. На месте board[cell[0]][cell[1]] ставим ‘\*’ (атакуемая клетка). Далее функция возвращает board.

Теперь рассмотрим, что делает функция attacked\_cell(). Функция принимает координату x (x: int), y (y: int) и размер доски (size: int), а возвращает список кортежей координат клеток. Переменной attacked передаётся пустой список, там будут хранится координаты клеток, находящихся под ударом. Далее мы с помощью цикла for (Рис. 1) проверяем, существует ли такая клетка, если да, то добавляем её в список attacked. Функция возвращает attacked.

  
Рис 1.

Вернёмся к функции chess(). Она принимает в аргументах первую координату x (xstart: int), y необходимый для того, чтобы мы не проходились по уже пройденным x, y (ystart: int), доску (board: list), количество фигур, которые необходимо поставить (fig\_num: int) и уже стоящие фигуры (standing: list). Сперва мы делаем firstboard глобальной, чтобы можно было её изменить. Далее идёт проверка на количество фигур, которые надо поставить. Если все фигуры поставлены, то мы проверяем, является ли firstboard None’ом, если да, то передаём её значение standing. Потом с помощью цикла for записываем методом write() координаты фигур standing в файл output. Если не все фигуры поставлены, то мы циклом for проходимся от xstart до len(board) по координатам x. Внутри этого цикла мы проходимся циклом for по координатам y от 0 до len(board). Далее следует проверка равняется ли x xstart’у и меньше ли y ystart’а. Это так же нужно, чтобы мы не проходились по уже пройденным клеткам. Если условие выполняется, то с помощью continue мы переходим к следующей итерации. Если нет, то мы проверяем, равен ли элемент board[x][y] нулю и возвращает ли функция under\_attack(), с переданными в качестве аргументов координатами x, y и доски board, True.

Теперь, чтобы идти дальше, необходимо рассмотреть, что делает функция under\_attack(). Она принимает координаты x (x: int), y (y: int) и доску (board: list), а возвращает True или False. Функция, с помощью цикла for (Рис. 2), проверяет существует ли клетка и стоит ли там фигура, так как если стоит, то следовательно клетка x, y находится под ударом. Если да, то возвращает False, если ни на одной клетке не стоит фигуры, то возвращает True.

  
Рис. 2

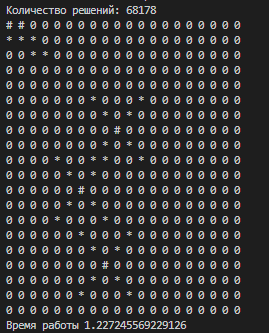
Вернёмся к chess(). Если условия выполнились, то элемент board[x][y] принимает значение ‘#’ (фигура), потом мы вызываем функцию chess, в которую передаём x, y, fig\_num – 1, а так же фигуры, которые сейчас стоят на доске (standing + [tuple([x, y])]). Тем самым создаётся рекурсия. Далее элемент board[x][y] принимает значение 0 (удаляем фигуру).

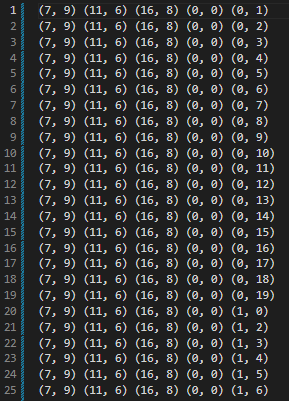
Когда функция chess() заканчивает свою работу, переменная btime принимает разность time() и atime. Далее с помощью метода seek() с переданным аргументом 0, возвращаемся к началу файлу output. Следующим шагом, мы с помощью метода read() (чтобы корректно использовать этот методы, необходимо было вернуться к началу файла) проверяем пуст ли файл. Если нет, то мы опять возвращаемся к началу файла и с помощью f-строк и len(output.readlines()) (функция len() вывод длину списка, который образовал метод readlines()) выводим количество решений. Потом с помощью цикла for и функции putting\_given\_figure() c переданными аргументами int(N) и firstboard выводится первая доска с расставленными фигурами. В конце выводится время работы программы. Если файл пуст, то выводится ‘no solution’, то же записывается в файл output.

Когда функция main() заканчивает свою работу, метод close() закрывает файловые объекты.

# РЕЗУЛЬТАТ РАБОТЫ

Для заданного в файле ‘input.txt’:  


Вывод в консоли будет выглядеть так:  


А файл output будет выглядеть так:  
  
(И так далее)

# ВЫВОД

В результате работы цели были достигнуты в решении присутствует: 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.

# Листинг кода

