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**

Дисциплина: **Программирование**

Группа: **20121**

Тип работы: **Курсовой проект**

Цель: Анализ ходов фигуры на шахматной доске

При работе над данным лабораторным проектом будут затронуты следующие вопросы:

1. Работа с основами функционального программирования языка Python
2. Работы с основами ООП языка Python
3. Разработка классов и UML диаграмма
4. Работы с пакетами Python
5. Создание GUI приложения с использованием tkinter (или customtkinter) + pygame (+OpenGl).

Формулировка задания:

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. Требуется найти одно решение для визуализации и все возможные решения для вывода в файл. Если решение не найдено, то необходимо вывести соответствующее сообщение.

1) Необходимо создать UML диаграмму взаимодействия классов. При необходимости можно делать несколько UML диаграмм. Для диаграммы создается табличка взаимодействия (Имя класса :: Имя базового класса :: Описание). Также для каждого класса создается таблица методов и атрибутов (Методы и атрибуты :: Описание). При описании методов обязательно прописывать тип параметров, а также выходной тип данных. При описании атрибутов необходимо прописывать его тип.

2) Необходимо составить программу с использованием функционального программирования языка Python, где:

1. 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 стоящей на доске фигуры (фигуры стоят правильно). Координаты отсчитываются от 0 до N-1. 1 <= N <= 20.
2. Выходные данные 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.
3. Выходные данные на консоль — это доска N*N, где фигура обозначается #, ее ходы обозначаются *, а пустые клетки обозначаются 0.

3) Необходимо составить программу с использованием ООП языка Python, а также разработать интерфейса при помощи пакетов tkinter (или customtkinter) + pygame (+OpenGl).

Ваша задача реализовать любой из предложенных вариантов:

1. tkinter (или customtkinter) + pygame(+OpenGl)).
   1.1. tkinter (или customtkinter)
      1.1.1. На интерфейсе есть два поля ввода и кнопка. В первом поле вводится размер доски N, во втором – количество фигур, которые необходимо расставить с помощью алгоритма L, и с помощью кнопки создается новое окно.

      Необходимо проверять правильность ввода данных.
   1.2. pygame (+OpenGl)
      1.2.1. В данном интерфейсе есть шахматная доска и кнопка. На созданной шахматной доске с помощью ПКМ/ЛКМ необходимо расставить/убрать стоящие на доске K фигур. Необходимо визуализировать саму фигуру и ее ходы (визуализацию фигуры и ее ходов выбираете сами, но так, чтобы они отличались). Также необходимо проверить, чтобы поставленная вами фигура не находилась под боем.

      С помощью кнопки создается новое окно.
      1.2.2. В данном интерфейсе есть шахматная доска и кнопка. На созданной шахматной доске показываются фигуры, расставленные пользователем и найденные алгоритмом, (нужно показать любое

найденное решение). Визуализация пользовательских и найденных алгоритмом фигур должна отличаться (визуализация ходов одинаковая).

Если решение не было найдено, то вывести сообщение об этом и закрыть данный интерфейс.

С помощью кнопки происходит вывод данных 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. Порядок, в котором фигуры перечислены в решении, не имеет значения.

2. tkinter (или customtkinter) + pygame(+OpenGl)).

2.1. tkinter (или customtkinter)

2.1.1. На интерфейсе есть три поля ввода и кнопка. В первом поле вводится размер доски N, во втором – количество фигур, которые необходимо расставить с помощью алгоритма L, в третьем – количество стоящих на доске фигур K. С помощью кнопки создается новое окно.

Необходимо проверять правильность ввода данных.

2.1.2. В данном интерфейсе есть K полей ввода и кнопка. В каждое поле необходимо ввести числа x и y (через пробел) - координаты уже стоящей на доске фигуры. С помощью кнопки создается новое окно.

Необходимо проверять правильность ввода данных.

2.2. pygame (+OpenGl)

2.2.1. В данном интерфейсе есть шахматная доска и кнопка. На созданной шахматной доске показываются фигуры, расставленные пользователем и найденные алгоритмом, (нужно

показать любое найденное решение). Визуализация пользовательских и найденных алгоритмом фигур должна отличаться (визуализация ходов одинаковая).

Если решение не было найдено, то вывести сообщение об этом и закрыть данный интерфейс.

С помощью кнопки происходит вывод данных 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. Порядок, в котором фигуры перечислены в решении, не имеет значения.

Дополнительная информация:

1. **Обязательное использование аннотации и комментариев**
2. **Проверка ввода данных.**

Требование к оформлению КП:

1. Титульный лист
2. Оглавление
3. Цели и формулировка задачи
4. Разработка классов и UML диаграмма
5. Результаты работы

4.1. Реализация программы с использованием функционального программирования языка Python

4.1.1. Ход работы

4.1.2. Демонстрация работы программы

4.1.3. Листинг кода