МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Санкт-Петербургский государственный морской технический университет»  
 (СПБГМТУ)

|  |
| --- |
| ФАКУЛЬТЕТ ЦИФРОВЫХ ПРОМЫШЛЕННЫХ ТЕХНОЛОГИЙ  КАФЕДРА КИБЕРФИЗИЧЕСКИХ СИСТЕМ |

КУРСОВОЙ ПРОЕКТ

по дисциплине “программирование”

|  |
| --- |
|  |

|  |
| --- |
| Выполнил студент группы 20121 |
| Романова Елена Сергеевна |
| Проверил |
| Минин М.С. |

Санкт-Петербург

2023 год.

содержание

[Цели 3](#_Toc138379815)

[Формулировка задачи 4](#_Toc138379816)

[Разработка классов и UML диаграмма 7](#_Toc138379817)

[Результаты работы 9](#_Toc138379818)

[Заключение 15](#_Toc138379819)

[Список используемых источников 16](#_Toc138379820)

# Цели

**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 задачи

**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 ни одна 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)

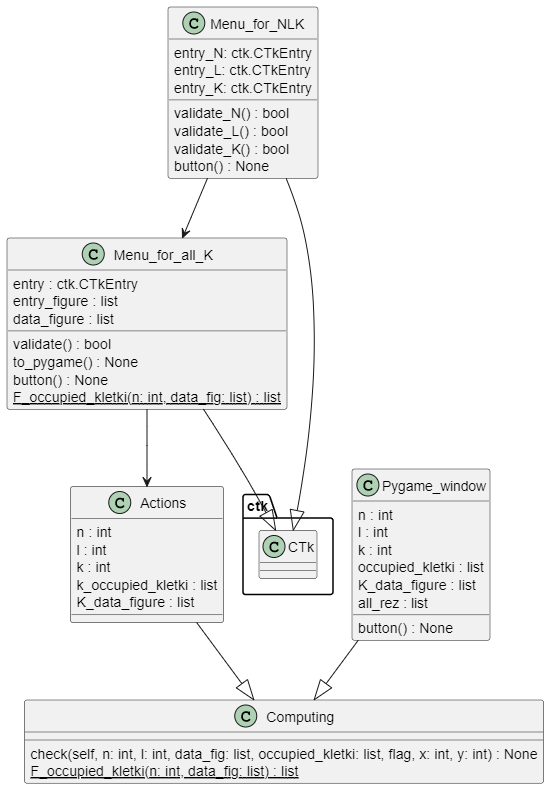
2.2.1. В данном интерфейсе есть шахматная доска и кнопка. На созданной шахматной доске показываются фигуры, расставленные пользователем и найденные алгоритмом, (нужно показать любое найденное решение). Визуализация пользовательских и найденных алгоритмом фигур должна отличаться (визуализация ходов одинаковая). 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.

# Разработка классов и UML диаграмма



Пояснения:

*Menu\_for\_NLK (ctk.CTk)* - класс, отвечающий за первое окно CustomTkinter, которое содержит поля ввода N, L, K, содержащий кнопку для перехода к следующему окну (кнопка становится доступной/активной после ввода всех значений).

*Menu\_for\_all\_K(ctk.CTk)* - класс, отвечающий за второе окно CustomTkinter. В окне находится фрейм с полосой прокрутки, содержащий K полей ввода для каждой из стартовых фигур, и кнопка.

*Computing* – класс, содержащий в себе основные методы взаимодействия с координатами фигур.

*Actions* – класс, отвечающий за нахождение и передачу значений первого решения классу Pygame\_window.

*Pygame\_window* – класс, отвечающий за финальное окно Pygame с визуализацией первого решения и кнопкой вывода решений в файл.

# Результаты работы

Была выполнена только вторая реализация.

1. **Ход работы:**

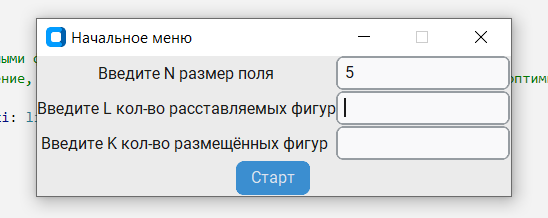
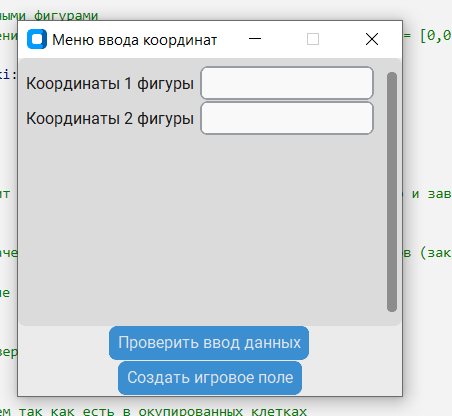
При запуске программы первым появляется окно класса *Menu\_for\_NLK*, которое отвечает за считывание и проверку значений N(размер поля), L(количество фигур, которые необходимо расставить с помощью программы), K(количество фигур, за которыми уже закреплены координаты).

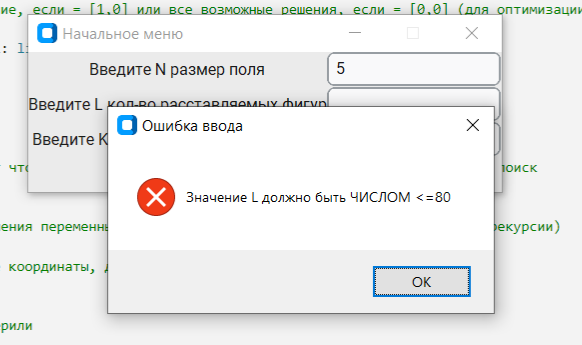
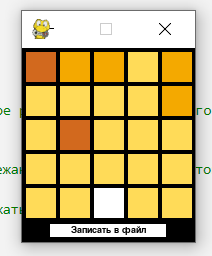
Если все значения были введены корректно, данные передаются классу *Menu\_for\_all\_K* после нажатия на кнопку, при этом создаётся новое окно для ввода координат уже расставленных фигур и закрывается первое окно. В этом классе прописаны проверки на правильность введенных значений для координат. А также есть 2 кнопки. Первая 'Проверить ввод данных' проводит завершающую проверку введенных значений и меняет значение переменной flag, благодаря чему теперь при нажатии на кнопку 'Создать игровое поле' программа переходит к главной вычислительной части, передавая значения классу *Actions*.

Этот класс наследуется от класса *Computing*, который содержит основные методы взаимодействия с координатми фигур. Он, используя метод check родительского класса находит первое возможное решение и передает полученные данные классу *Pygame\_window.*

Класс *Pygame\_window* создает окно pygame на котором находится игровое поле, отражающее первое решение, и кнопка, которая, с помощью методов родительского класса *Computing* вычисляет все возможные решения для введенных данных и записывает из в файл.

* 1. **Демонстрация работы программы.**

** **

** **

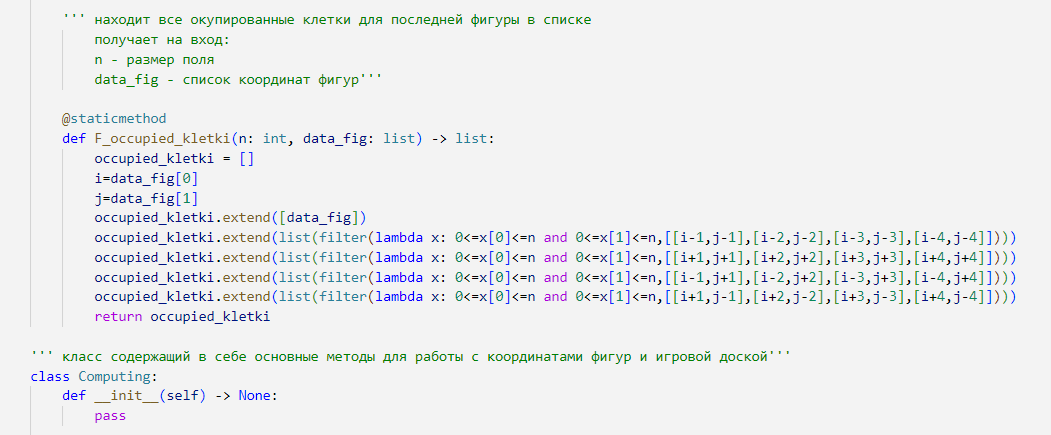
* 1. **Листинг кода.**

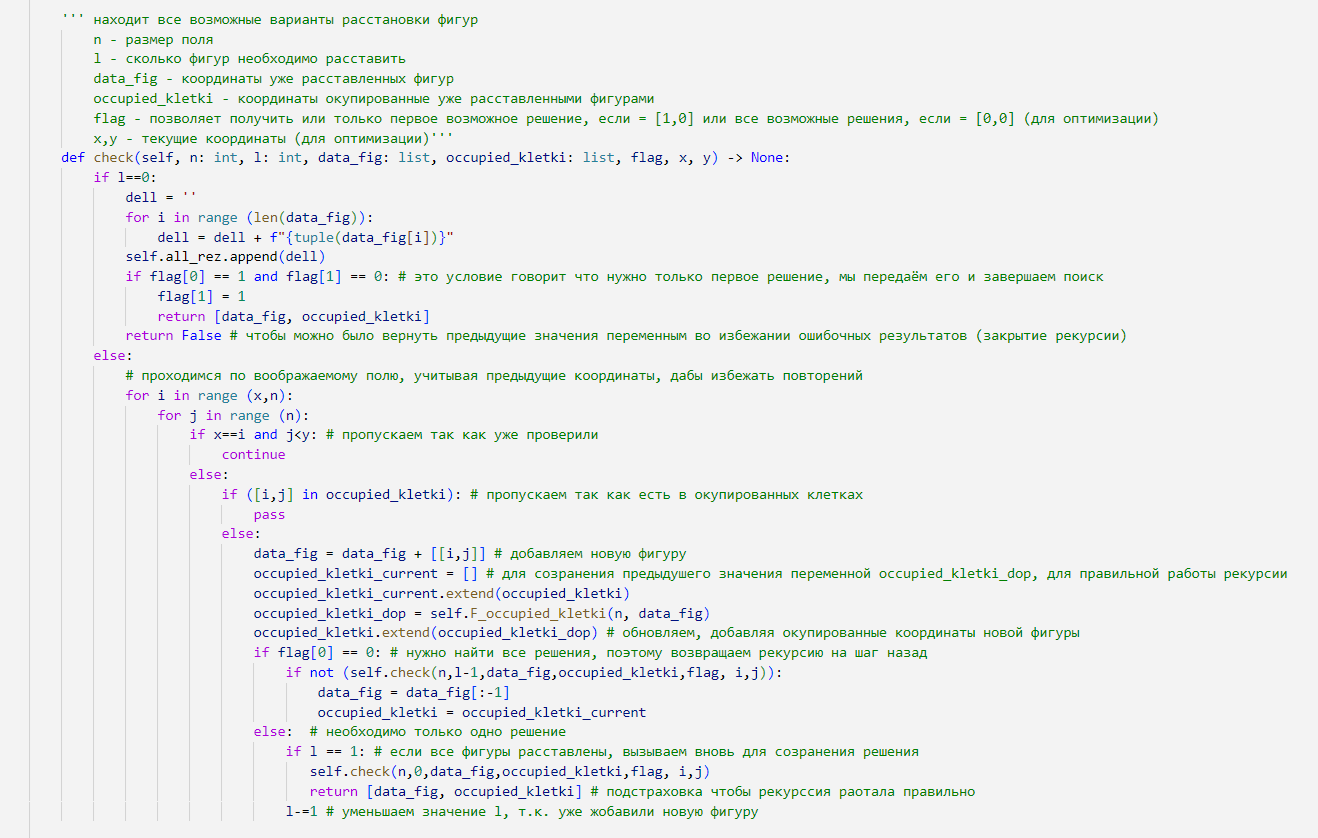


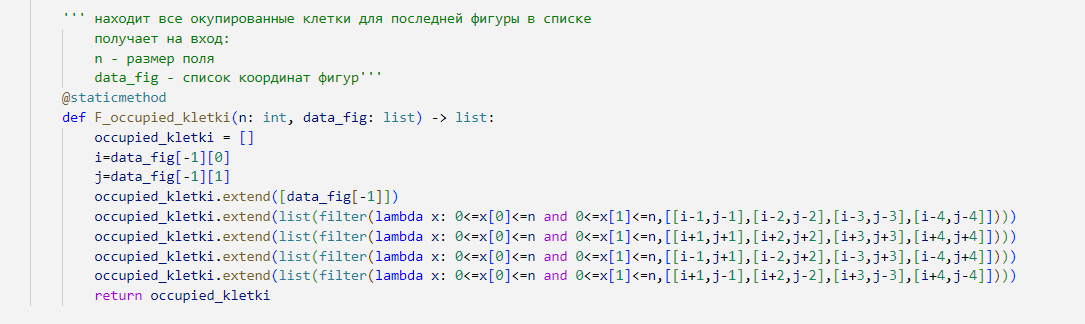


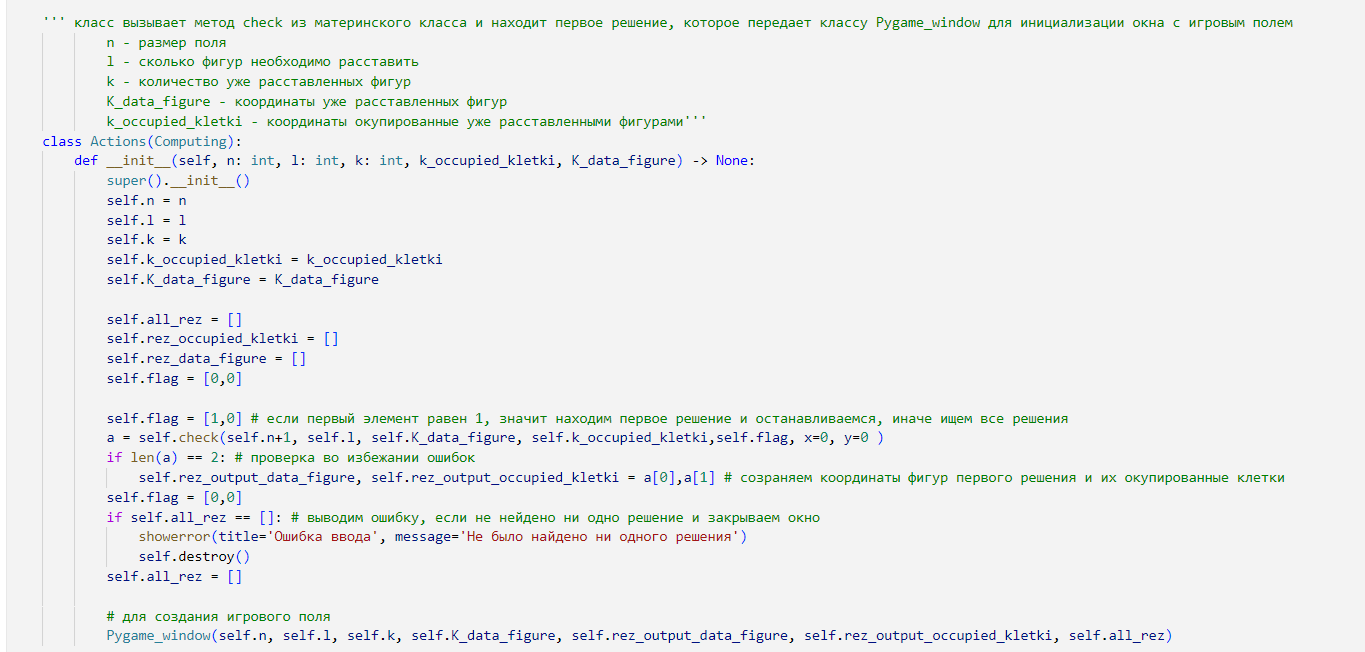






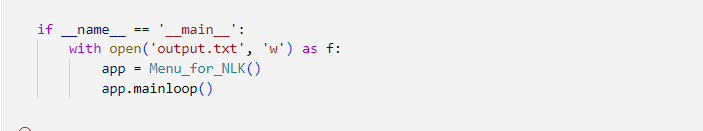












# Заключение

В результате работы над курсовым проектом я:

* укрепила свои знания в области работы с основами функционального программирования языка Python;
* впервые реализовала полноценный проект с использованием основ ООП языка Python;
* научилась работать с UML диаграммами;
* научилась создавать GUI с использованием customtkinter и pygame;
* попрактиковалась правильно оформлять код с помощью docstring;

# Список используемых источников

* <https://customtkinter.tomschimansky.com/documentation/windows/toplevel/>
* <https://github.com/pygame/pygame>
* <https://docs-python.ru/standart-library/modul-typing-python/>
* <https://docs-python.ru/standart-library/modul-functools-python/funktsija-partial-modulja-functools/>