Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

**Лабораторная работа.**

**«Задача о восьми ферзях»**

Выполнил студент гр. РИС-24-2б

Молочко Артём Анатольевич

Проверил:

Доц. Каф. ИТАС

Ольга Андреевна Полякова

(оценка) (подпись)

(дата)

г. Пермь, 2025

Постановка задачи:

Дано:

Шахматная доска размером 8×8 клеток и восемь ферзей.

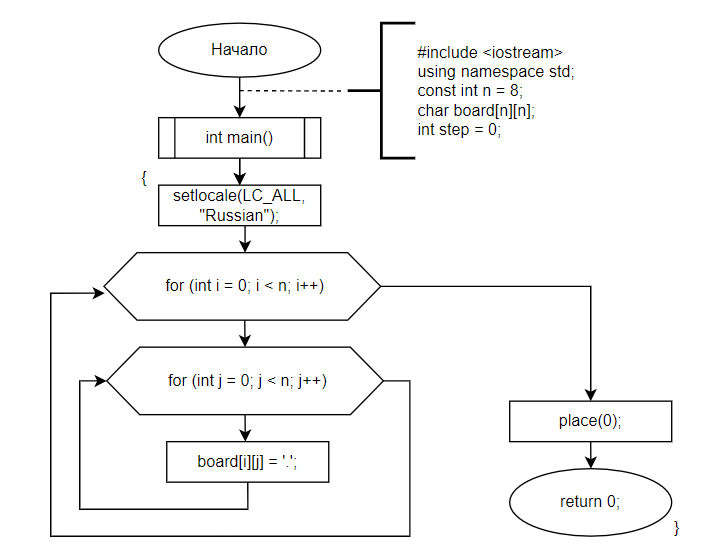
Требуется:

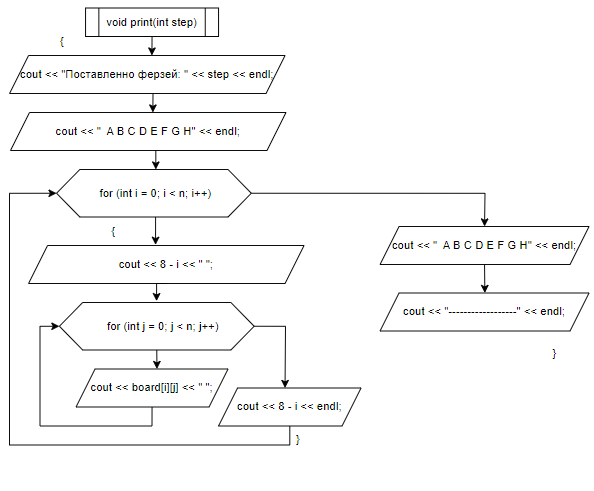
Расположить всех восьми ферзей на доске таким образом, чтобы ни один из них не атаковал другого. В шахматах ферзь может ходить по горизонтали, вертикали и диагонали, поэтому ни в одном ряду, столбце или диагонали не должно быть более одного ферзя.

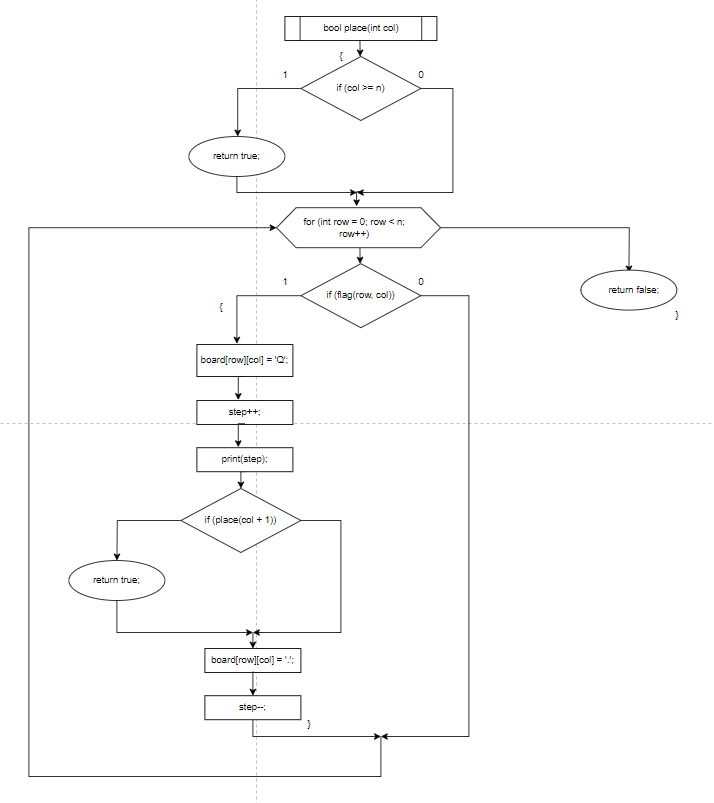
Анализ задачи:

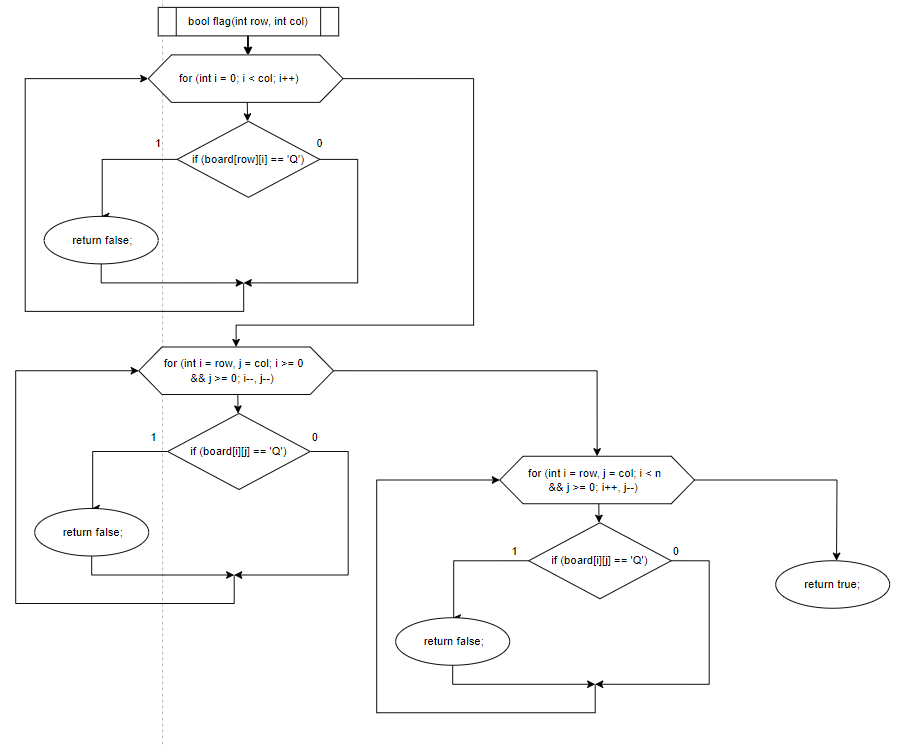
* В процессе решения ферзи поочередно размещаются в каждом столбце.
* Для каждой попытки размещения ферзя в столбце, проверяется, не будет ли он атаковать уже расставленных ферзей, с помощью функции flag, которая проверяет строку, столбец и диагонали.
* Если ферзя можно поставить в клетку, то алгоритм рекурсивно продолжает попытки размещения ферзей в следующем столбце.
* Если размещение ферзя приводит к тупику (не удается расставить всех ферзей), то происходит откат на шаг назад: ферзь удаляется, и поиск продолжается в другом месте.

Блок-схема









Визуализация рекурсии.

place(0)->Попытки разместить ферзя в строках первого столбца(col = 0)

|

|--place(1)->Попытки разместить ферзя во втором столбце(col = 1)

|

|--place(2)->Попытки разместить ферзя в третьем столбце(col = 2)

|

|--place(3)->Попытки разместить ферзя в четвертом столбце(col = 3)

|

... продолжаем до 8 столбцов

|

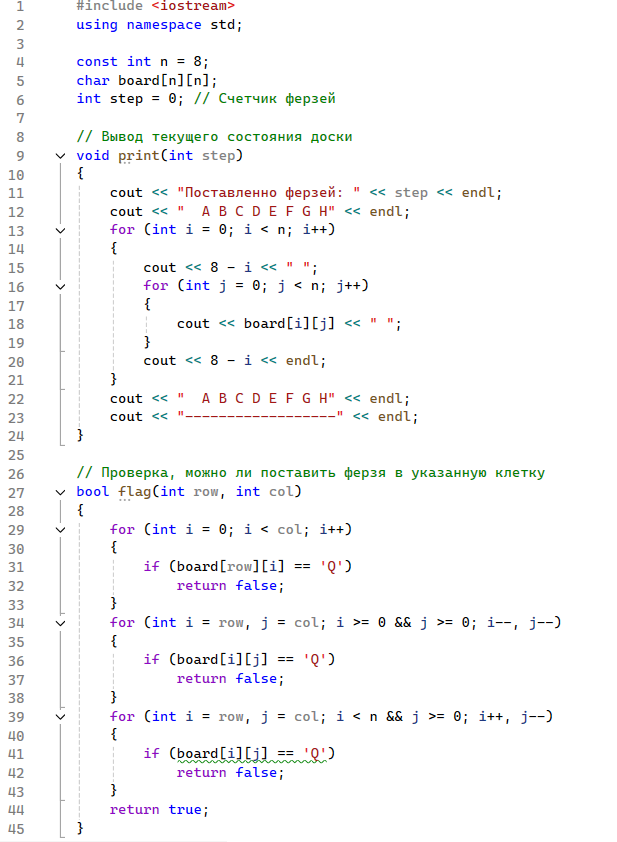
|--шаг назад(если ферзь не ставится)->возвращаемся на уровень выше

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. place(0)
   * Пробуем row = 0, col = 0 ✅
   * **Ферзь установлен на (0,0)**
   * Рекурсивный вызов: place(1)
2. place(1)
   * Пробуем row = 0, col = 1 ❌ (бьется ферзем (0,0))
   * Пробуем row = 1, col = 1 ❌ (бьется ферзем (0,0))
   * Пробуем row = 2, col = 1 ✅
   * **Ферзь установлен на (2,1)**
   * Рекурсивный вызов: place(2)
3. place(2)
   * Пробуем row = 0, col = 2 ❌ (бьется ферзем (0,0))
   * Пробуем row = 1, col = 2 ❌ (бьется ферзем (2,1))
   * Пробуем row = 2, col = 2 ❌ (бьется ферзем (2,1))
   * Пробуем row = 3, col = 2 ✅
   * **Ферзь установлен на (3,2)**
   * Рекурсивный вызов: place(3)
4. place(3)
   * Пробуем row = 0, col = 3 ❌ (бьется ферзем (0,0))
   * Пробуем row = 1, col = 3 ✅
   * **Ферзь установлен на (1,3)**
   * Рекурсивный вызов: place(4)

... и т.д

1. **Если в какой-то момент невозможно поставить ферзя в col, возвращаемся на шаг назад**
   * Если place(3) не нашло места, откатываем place(2), убираем ферзя и пробуем следующую строку.
   * Если place(2) не найдет решения, откатываем place(1), и так далее...
2. **Базовый случай:**
   * Если col == 8, значит, все 8 ферзей расставлены, и решение найдено.
   * Иначе идем назад меняя расположение предыдущих ферзей.

Код программы

