Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика**

ОТЧЕТ

по домашней работе

Дисциплина: Прикладная теория цифровых автоматов

Вариант: 19

Студент ИУ6-44Б (Группа) _____ 30.05.22 (Подпись, дата) С.С. Швецов (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) Ю.И. Бауман (И.О. Фамилия)

Москва, 2022

Задание

Разместить на шахматной доске 8 ферзей так, чтобы они не напали друг на друга. Подробно описать алгоритм решения.

Алгоритм решения данной задачи был реализован на языке программирования "Ruby". Данный язык был выбран мной потому, что "Ruby" предоставляет огромное количество методов обработки массивов и даст возможность вставлять собственные куски кода в виде блока в различные циклы, что позволяет избегать дублирование кода для различных методов, отличающихся в одну строку.

Код программы:

```
class ChessDesk

  def initialize
    @desk = [[0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0],]
    @head_of_desk = ["  a b c d e f g h",
                     "1 ",
                     "2 ",
                     "3 ",
                     "4 ",
                     "5 ",
                     "6 ",
                     "7 ",
                     "8 "]
    @positions = []
  end

  def draw_desk
    if @positions.empty?
      puts @head_of_desk[0]
      @head_of_desk.shift
      @head_of_desk.each { |elem| print elem; 8.times { print "# " }; puts "" }.unshift("  a b c d e f g h")
    else
      @positions.map! { |elem| elem.reverse }.sort!
      puts @head_of_desk[0]
      for i in 1..8
        print @head_of_desk[i]
        @positions[i - 1][1].times { print "# " }
        print "F "
        (7 - @positions[i - 1][1]).times { print "# " }
        puts ""
      end
```

Продолжение кода программы:

```
    end
end

def add_position(position)
    @positions.push(position)
    self.change_desk { |elem| elem = elem + 1 }
end

def remove_position
    self.change_desk { |elem| elem = elem - 1 }
    @positions.pop
end

def change_desk
    @desk[@positions.last[1]].map! { |elem| yield elem }
    @desk.map { |elem| elem[@positions.last[0]] = yield elem[@positions.last[0]] }
    i = @positions.last[0]
    j = @positions.last[1]
    # i - столбец
    # j - строка
    while (i > 0) and (j > 0)
        i -= 1
        j -= 1
    end
    while ((i <= 7) and (j <= 7))
        @desk[j][i] = yield @desk[j][i]
        i += 1
        j += 1
    end
    i = @positions.last[0]
    j = @positions.last[1]
    while (i > 0) and (j < 7)
        i -= 1
        j += 1
    end
    while ((i <= 7) and (j >= 0))
        @desk[j][i] = yield @desk[j][i]
        i += 1
        j -= 1
    end
end

def set_first_position(position)
    array = position.split("").map { |elem| elem.to_i != 0  ? elem = elem.to_i - 1 : elem = elem.ord - 97 }
    self.add_position(array)
end

def search_combination(string_number)
    if string_number == @positions.first[1]
        string_number += 1
    end
    for i in 0..7
        if string_number == 8
            return
        end
        if @desk[string_number][i] == 0
            self.add_position([i, string_number].to_a)
```

Продолжение кода программы:

```
            self.search_combination(string_number + 1)
            if string_number == @positions.first[1]
               string_number -= 1
            end
          end
        end
        if @positions.length == 8
          return
        elsif i == 7
          self.remove_position
        end
      end
end


puts "Перед вами шахматная доска"
puts "Задача: разместить 8 ферзей на шахматной доске таким образом, чтобы они не могли напасть друг на друга"
chess_desk = ChessDesk.new
chess_desk.draw_desk
puts "Введите стартовой положение ферзя на доске"
puts "Пример формата ввода: 'a1'"
print "Ввод: "
input_position = gets.chomp
chess_desk.set_first_position(input_position)
chess_desk.search_combination(0)
puts "Вариант решения"
chess_desk.draw_desk
```

Был реализован класс, поля которого содержат три массива. @desk - массив весов, который отражает пустые клетки в ходе выполнения алгоритма и клетки, перекрывающиеся поставленной фигурой. @positions - массив, хранящий координаты каждой размещенный фигуры. На момент создания объекта массив пуст. Во время поиска нужной расстановки элементы могут как добавляться в массив так и удаляться. @head_of_desk - массив строк, использующийся для вывода информации.

В начале алгоритма в консоль выводится сообщение, предлагающее ввести пользователю стартовую позицию размещения первого ферзя, относительно которого и будет осуществлен поиск нужной комбинации.

Далее вызывается метод set_first_position. Здесь введенная пользователем строка-ячейка преобразуется в числовые координаты, адресующие в массиве весов нужный элемент, относительно которого будут рассчитываться веса. Далее внутри этого метода вызывается метод

add_position с аргументом - массивом, состоящим из двух элементов и задающим позицию размещения первого ферзя.

В методе add_position координаты добавляются в массив @positions. Далее вызывается метод change_desk с параметром в виде блока, хранящего код, который будет вызван внутри этого метода. Ему передается элемент матрицы, который инкрементируется внутри этого блока. Таким образом, помечаются все элементы шахматной доски, куда ферзь может переместиться за 1 ход.

Затем из основной программы вызывается метод search_combination. Методу в качестве аргумента передается номер строки, в которой осуществляется поиск незанятой клетки. Если такая клетка существует, вызывается метод add_position и на доску выставляется новый ферзь. Далее метод search_position вызывается рекурсивно. В аргументе указывается номер следующей строки. То есть поиск свободных клеток осуществляется построчно с верхнего края доски вниз. Если свободная клетка не найдена, то последний выставленный ферзь удаляется и осуществляется возврат управления в предыдущий фрейм активации, где дальнейшие действия происходят по такой же схеме. Когда размещено все 8 ферзей, рекурсия прекращается и осуществляется возврат управления основной программе.

Далее вызывается метод draw_desk, отрисовывающий доску и найденное размещение ферзей.

```
PS C:\Users\User\Desktop\Ruby\theory of digital automata> ruby ChessTask.rb
Перед вами шахматная доска
Задача: разместить 8 ферзей на шахматной доске таким образом, чтобы они не могли напасть друг на друга
  a b c d e f g h
1 # # # # # # # #
2 # # # # # # # #
3 # # # # # # # #
4 # # # # # # # #
5 # # # # # # # #
6 # # # # # # # #
7 # # # # # # # #
8 # # # # # # # #
Введите стартовой положение ферзя на доске
Пример формата ввода: 'a1'
Ввод: d5
Вариант решения
  a b c d e f g h
1 # F # # # # # #
2 # # # # # F # #
3 F # # # # # # #
4 # # # # # # F #
5 # # # F # # # #
6 # # # # # # # F
7 # # F # # # # #
8 # # # # F # # #
PS C:\Users\User\Desktop\Ruby\theory of digital automata>
```

Рисунок 1 - Результат работы программы

q0 – Приглашение пользователя осуществить ввод данных
q1 – Установка начальных условий поиска комбинации
q2 – Поиск позиции и размещение на доске второй фигуры
q3 – Поиск позиции и размещение на доске третьей фигуры
q4 – Поиск позиции и размещение на доске четвертой фигуры
q5 – Поиск позиции и размещение на доске пятой фигуры
q6 – Поиск позиции и размещение на доске шестой фигуры
q7 – Поиск позиции и размещение на доске седьмой фигуры
q8 – Поиск позиции и размещение на доске восьмой фигуры
q9 – Вывод найденной схемы в консоль
A – Фигура выставлена
B – Свободная позиция не найдена. Удаление фигуры

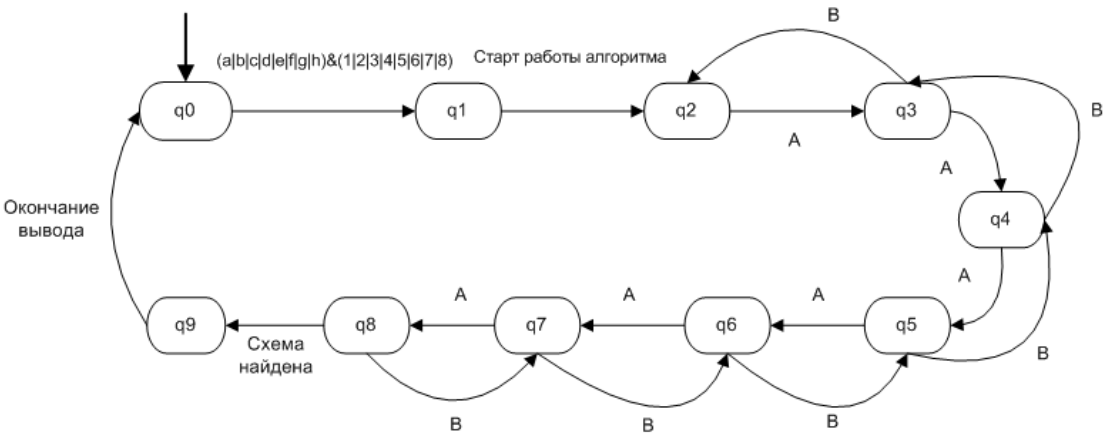

Рисунок 2 - Автомат в виде графа с расписанными обозначениями различных состояний и сигналов

Вывод: В ходе данной домашней работы был реализован алгоритм решения задачи о размещении 8 ферзей на шахматной доске, написанный на языке Ruby, а так же представлен автомат в виде ориентированного графа, и описаны его состояния при подаче различных сигналов.