# ОТЧЕТ: ПОСЛЕДОВАТЕЛЬНОСТЬ РАКАМАНА

## A. ТИТУЛЬНАЯ СТРАНИЦА

**ФИО студента:** Ефремов Николай Александрович

**Номер студенческого билета:** 245111

**Название предмета и код:** Алгоритмы и Структуры Данных и Парадигмы Программирования

**Номер лабораторной работы:** Лабораторная работа №3

**Номер группы:** ИД24-2

**Дата сдачи:** 25.11.2025

**Название задачи:** Последовательность Ракамана (Task #8)

## B. ОПИСАНИЕ ЗАДАНИЯ

**Задача №8 - Последовательность Ракамана**

Реализовать популярную визуализацию последовательности Ракамана через полуокружности на числовой прямой. Последовательность определяется следующим образом:

- a(0 = 0
- a(n) = a(n-1) - n, если результат положителен и не встречался ранее иначе a(n)
- = a(n-1) + n **Основные требования:**

Реализовать визуализацию последовательности через полуокружности на числовой прямой

Реализовать автоматическое масштабирование для адаптации к размерам экрана

Визуализировать другие известные натуральные последовательности (опционально) Реализовать управление программой с клавиатуры **Дополнительные требования (реализованы):**

- Приостановка и возобновление анимации
- Вывод текущего числа последовательности
- Градиентная окраска дуг
- Конфигурационный файл для управления настройками
- ООП подход с разделением ответственности между классами

## C. ДОСТИГНУТАЯ СЛОЖНОСТЬ И РЕАЛИЗАЦИЯ

**Выполненные основные требования:**

✓ Генерация последовательности Ракамана с проверкой уникальности чисел

✓ Визуализация через полуокружности с чередованием направлений (вверх/вниз)
✓ Отрисовка числовой прямой с маркерами для каждого значения
✓ Автоматическое масштабирование под размер окна
✓ Управление клавиатурой (SPACE - пауза, R - сброс, ESC - выход)

**Реализованная дополнительная функциональность:**
✓ Градиентная окраска дуг (плавный переход от синего к красному)
✓ Вывод текущего числа последовательности в углу экрана
✓ Конфигурационный файл JSON для управления всеми параметрами
✓ Счетчик прогресса анимации (Arc: n/m)
✓ Плавная анимация с контролем FPS

**Оригинальные расширения:**
✓ Множественные цветовые схемы в конфигурации
✓ Параметризация всех визуальных элементов (толщина линий, размеры маркеров)
✓ Сегментирование полуокружностей для плавности отрисовки
✓ Методы для легкого расширения (добавления новых визуализаций)

## D. ОБЪЯСНЕНИЕ ПРОЕКТИРОВАНИЯ ПРОГРАММЫ

**Структура ООП:**

1. **RecamanSequence** - генерирует последовательность Ракамана
   - Методы: generateNext(), generateTerms(), getMaxValue()
   - Отвечает за математику и логику последовательности
2. **ArcVisualizer** - отвечает за отрисовку дуг
   - Методы: interpolateColor(), drawArc()
   - Отвечает за визуальное представление на экране
3. **RecamanVisualization** - главный координатор
   - Методы: calculateScaling(), handleEvents(), update(), draw(), run()
   - Управляет взаимодействием, анимацией и основным циклом программы **Организация кода:**
   - Разделение ответственности: каждый класс отвечает за свою область
   - Генерация логики (RecamanSequence) отделена от визуализации (ArcVisualizer)
   - Главный класс (RecamanVisualization) координирует работу
   - Конфигурация вынесена в отдельный файл config.json
   - Используются словари для гибкого управления параметрами **Математическое исследование:**

**Последовательность Ракамана:**

- Определена в 1991 году Бернардо Ракаманом
- Каждое число появляется максимум один раз (инъективность)
- Неизвестно, содержит ли все натуральные числа (открытая гипотеза)
- Число 852655 не найдено даже после 10^230 членов
- Максимальные значения растут непредсказуемо (требует масштабирования) **Управление настройками:**

Конфигурационный файл config.json содержит:

- windowWidth, windowHeight - размеры окна maxTerms -
- количество членов последовательности animationSpeed -
- FPS анимации
- arcSettings - параметры дуг (толщина, цвета, сегменты) autoScaling -
- автоматическое масштабирование с отступами numberLineSettings -
- параметры числовой прямой

## E. ОСНОВНАЯ ЧАСТЬ - ТРЕБОВАНИЯ И РЕШЕНИЯ

**Требование 1 Генерация последовательности Ракамана**

Решение: Реализован класс RecamanSequence, который генерирует последовательность согласно алгоритму. Используется множество (set) для O 1 проверки уникальности чисел. Метод generateNext() вычисляет следующий член, generateTerms() генерирует несколько членов сразу.

Трудности: Необходимо эффективно проверять наличие числа в последовательности решено через множество (set) вместо списка.

**Требование 2 Визуализация через полуокружности**

Решение: Класс ArcVisualizer отрисовывает полуокружности между парами соседних чисел. Каждая дуга генерируется как набор точек на окружности с использованием тригонометрии. Дуги чередуются вверх/вниз для лучшей читаемости.

Трудности: Определение центра и радиуса дуги, преобразование математических координат в экранные, сегментирование для плавности.

**Требование 3 Автоматическое масштабирование**

Решение: Метод calculateScaling() вычисляет масштаб на основе максимального значения в последовательности. Формула: scale = (width - 2*padding) / maxValue. Это позволяет всей визуализации поместиться на экране.

Трудности: Максимальные значения непредсказуемы и могут измениться при добавлении новых членов.

**Требование 4 Управление и анимация**

Решение: Метод handleEvents() обрабатывает нажатия клавиш (SPACE, R, ESC). Основной цикл run() вызывает handleEvents(), update(), draw() в порядке, контролируя FPS через clock.tick().

Трудности: Синхронизация между нажатиями клавиш и обновлением анимации, поддержание стабильного FPS.

## F. КОНФИГУРАЦИОННЫЙ ФАЙЛ

```
{
  "windowWidth" : 1200,
  "windowHeight" : 700,
  "backgroundColor" : [15, 15, 25],
  "maxTerms" : 100,
  "animationSpeed" : 60,
  "arcSettings" : {
    "strokeWeight" : 2,
    "colorStart" : [100, 150, 255],
    "colorEnd" : [255, 100, 150],
    "segments" : 50
  },
  "numberLineSettings" : {
    "yPosition" : 0.7,
    "color" : [200, 200, 200],
    "markerSize" : 5
  },
  "autoScaling" : {
    "enabled" : true,
    "padding" : 50
  }
}
```

## G. ПОЛНЫЙ ИСХОДНЫЙ КОД

```
import pygame
import json
import math

class RecamanSequence:
    # Генерация последовательности Ракамана: a(n) = a(n-1) - n, если положительное и
уник     def __init__(self):         self.sequence = [0]         self.seen = {0}
self.currentStep = 1

    def generateNext(self):         lastValue
= self.sequence[-1]         candidate =
lastValue - self.currentStep

        if candidate > 0 and candidate not in self.seen:
            nextValue = candidate         else:
nextValue = lastValue + self.currentStep


self.sequence.append(nextValue)
self.seen.add(nextValue)
```

```
self.currentStep += 1         return
nextValue

    def generateTerms(self, numTerms):
while len(self.sequence) < numTerms:
self.generateNext()

    def getMaxValue(self):         return
max(self.sequence) if self.sequence else 0


class ArcVisualizer:
    # Отрисовка полуокружностей между значениями
последовательности     def __init__(self, config):
self.config = config         self.arcSettings =
config["arcSettings"]

    def interpolateColor(self, index, totalArcs):
# Градиентный переход цвета от начала к концу
ratio = index / max(totalArcs - 1, 1)
colorStart = self.arcSettings["colorStart"]
colorEnd = self.arcSettings["colorEnd"]

        r = int(colorStart[0] + (colorEnd[0] - colorStart[0]) *
ratio)         g = int(colorStart[1] + (colorEnd[1] - colorStart[1])
* ratio)         b = int(colorStart[2] + (colorEnd[2] -
colorStart[2]) * ratio)         return (r, g, b)

    def drawArc(self, surface, startVal, endVal, arcIndex, totalArcs, scale, offsetX,
off         # Рисует дугу между двумя значениями (четные индексы - вверх, нечетные -
вниз)
```

```
        if startVal == endVal:
return

        center = (startVal + endVal) / 2.0
radius = abs(endVal - startVal) / 2.0

        # Чередование направления дуг
direction = 1 if arcIndex % 2 == 0 else -1

        screenCenterX = offsetX + center *
scale         screenCenterY = offsetY
screenRadius = radius * scale

        # Генерация точек полуокружности
points = []         segments =
self.arcSettings["segments"]

        for i in range(segments + 1):
            angle = math.pi * i /
segments             if direction == -1:
angle = -angle

            x = screenCenterX + screenRadius *
math.cos(angle)             y = screenCenterY + screenRadius
* math.sin(angle)             points.append((x, y))

        if len(points) > 1:             color =
self.interpolateColor(arcIndex, totalArcs)
            pygame.draw.lines(surface, color, False, points,
self.arcSettings["strokeWeig


class RecamanVisualization:
    # Главный класс визуализации последовательности
Ракамана     def __init__(self, configPath="config.json"):
with open(configPath, 'r') as f:             self.config =
json.load(f)

        pygame.init()
        self.width = self.config["windowWidth"]
self.height = self.config["windowHeight"]
        self.screen = pygame.display.set_mode((self.width, self.height))
pygame.display.set_caption("Recamán's Sequence Visualization")

        self.sequence = RecamanSequence()
self.visualizer = ArcVisualizer(self.config)

        self.currentArc = 0
        self.maxTerms = self.config["maxTerms"]
self.clock = pygame.time.Clock()
self.running = True         self.paused = False
self.sequence.generateTerms(self.maxTerms)

        self.scale = 1.0
self.offsetX = 0.0
```

```
        self.offsetY = self.height * self.config["numberLineSettings"]["yPosition"]

self.calculateScaling()

    def calculateScaling(self):
        # Автоматическое масштабирование для вмещения всей
последовательности         if not self.config["autoScaling"]["enabled"]:
return

        maxValue = self.sequence.getMaxValue()
padding = self.config["autoScaling"]["padding"]

        if maxValue > 0:
            self.scale = (self.width - 2 * padding) / maxValue
self.offsetX = padding

    def drawNumberLine(self):         lineY = self.offsetY         color =
tuple(self.config["numberLineSettings"]["color"])
pygame.draw.line(self.screen, color, (0, lineY), (self.width, lineY), 2)

        markerSize = self.config["numberLineSettings"]["markerSize"]         for
i, value in enumerate(self.sequence.sequence[:self.currentArc + 1]):
x = self.offsetX + value * self.scale             pygame.draw.circle(self.screen,
color, (int(x), int(lineY)), markerSize)

    def handleEvents(self):         for
event in pygame.event.get():
if event.type == pygame.QUIT:
                self.running = False
elif event.type == pygame.KEYDOWN:
if event.key == pygame.K_SPACE:
self.paused = not self.paused
elif event.key == pygame.K_r:
self.reset()                 elif event.key ==
pygame.K_ESCAPE:
self.running = False

    def reset(self):
        self.currentArc = 0

    def update(self):
        if not self.paused and self.currentArc < len(self.sequence.sequence) - 1:
self.currentArc += 1

    def draw(self):
        bgColor =
tuple(self.config["backgroundColor"])
self.screen.fill(bgColor)
self.drawNumberLine()

        for i in range(self.currentArc):
startVal = self.sequence.sequence[i]
endVal = self.sequence.sequence[i + 1]
totalArcs = len(self.sequence.sequence) - 1
```

```
            self.visualizer.drawArc( self.screen, startVal, endVal, i, totalArcs,
                                     self.scale, self.offsetX, self.offsetY)

        # Отображаем текущее число последовательности
        font = pygame.font.Font( None, 30)
        currentValue = self.sequence.sequence[ self.currentArc]
        textSurface = font.render( f"Текущее число: {currentValue }", True, (255, 255, 255)
        self.screen.blit(textSurface, ( 10, 40))

        self.drawInstructions()

        pygame.display.flip()

    def drawInstructions (self):
        font = pygame.font.Font( None, 24)
        instructions = [
            "SPACE: Pause/Resume" ,
            "R: Reset" ,
            "ESC: Exit" ,
            f"Arc: {self.currentArc }/{len(self.sequence.sequence) - 1}"
        ]

        y = 10
        for text in instructions:
            surface = font.render(text, True, (200, 200, 200))
            self.screen.blit(surface, ( 10, y))
            y += 25

    def run(self):
        while self.running:
            self.handleEvents()
            self.update()
            self.draw()
            self.clock.tick( self.config[ "animationSpeed" ])

        pygame.quit()


visualization = RecamanVisualization()
visualization.run()
```

## ОТВЕТЫ НА 5 ВОПРОСОВ

**1. Почему иногда идти назад?**

Алгоритм пытается вычесть текущее число шага, если это возможно и число не было использовано ранее. Это обеспечивает более сложный и интересный паттерн, а не простой прогресс вперёд.

**2. Как полукруги показывают числа?**

Каждая полуокружность соединяет пару соседних чисел в последовательности, визуализируя переходы от одного члена к другому на числовой прямой.

**3. Почему паттерн вверх-вниз?**

Чтобы дуги не перекрывались и визуализация была более читаемой, дуги чередуются вертикально: одна дуга вверх, следующая вниз.

**4. Зачем автоматически подстраивать размер?**

Максимальные значения последовательности могут расти быстро и непредсказуемо, масштабирование позволяет уместить всю визуализацию в окно.

**5. Появятся ли все числа в конце?**

Это не доказано. Известны примеры, что некоторые числа могут не появиться даже при очень большом числе шагов.