**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 «Программное обеспечение ЭВМ и информационные технологии»

# Отчет по лабораторной работе №7 по дисциплине «Анализ Алгоритмов»

**Тема** Графовые модели

**Студент** Смирнов И.В.

**Группа** ИУ7-52Б

**Преподаватель** Волкова Л. Л., Строганов Д.В.

2024 г.

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

В данной лабораторной работе исследуется применение графовых моделей для анализа программного кода, реализующего конвейерную многопоточную обработку рецептов с веб-сайта. Использование таких моделей позволяет наглядно отразить информационные и управляющие зависимости, упростить оптимизацию и понимание структуры программы.

**Цель лабораторной работы** — получение навыка построения и применения графовых моделей для анализа кода. Для достижения поставленной цели необходимо решить следующие задачи:

— построить информационный граф, информационную историю, граф управления и операционную историю для выбранного фрагмента кода;

— проанализировать построенные графовые модели;

— сделать вывод о применимости графовых моделей к задаче анализа программного кода.

# 1 Фрагмент кода

Фрагмент кода на 15 осмысленных / значимых строк кода (не пробелы, не комментарии, не фигурные скобуи и т.п.) представлен в листинге 1.1.

Листинг 1.1 – Фрагмент кода

```
1  string inputFilePath =
       Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
       "inputfile.txt");
2  using (var writer = new StreamWriter(inputFilePath, false)) {
3      for (int i = 2; i <= pageLimit + 1; i++) {
4          string pageUrl = $"{baseUrl}?page={i}";
5          var doc = LoadHtmlDocument(pageUrl);
6          if (doc == null) {
7          logger.LogCollector($"Cant load page {pageUrl}");
8          continue; }
9          var recipeLinks = ExtractRecipeLinks(doc);
10         Console.WriteLine($"{pageUrl} - {recipeLinks.Count}");
11         logger.LogCollector($"Page {i} loaded, links amount:
               {recipeLinks.Count}");
12         if (recipeLinks.Count == 0)
13            logger.LogCollector($"Page {i} has no links.");
14         else {
15             foreach (var link in recipeLinks)
16                 writer.WriteLine(link);
17             logger.LogCollector($"Page {i} loaded, links amount:
                   {recipeLinks.Count}"); }
18     }
19 }
```

# 2 Информационный граф

Информационный граф по приведенному листингу кода представлен на рисунке 2.1.

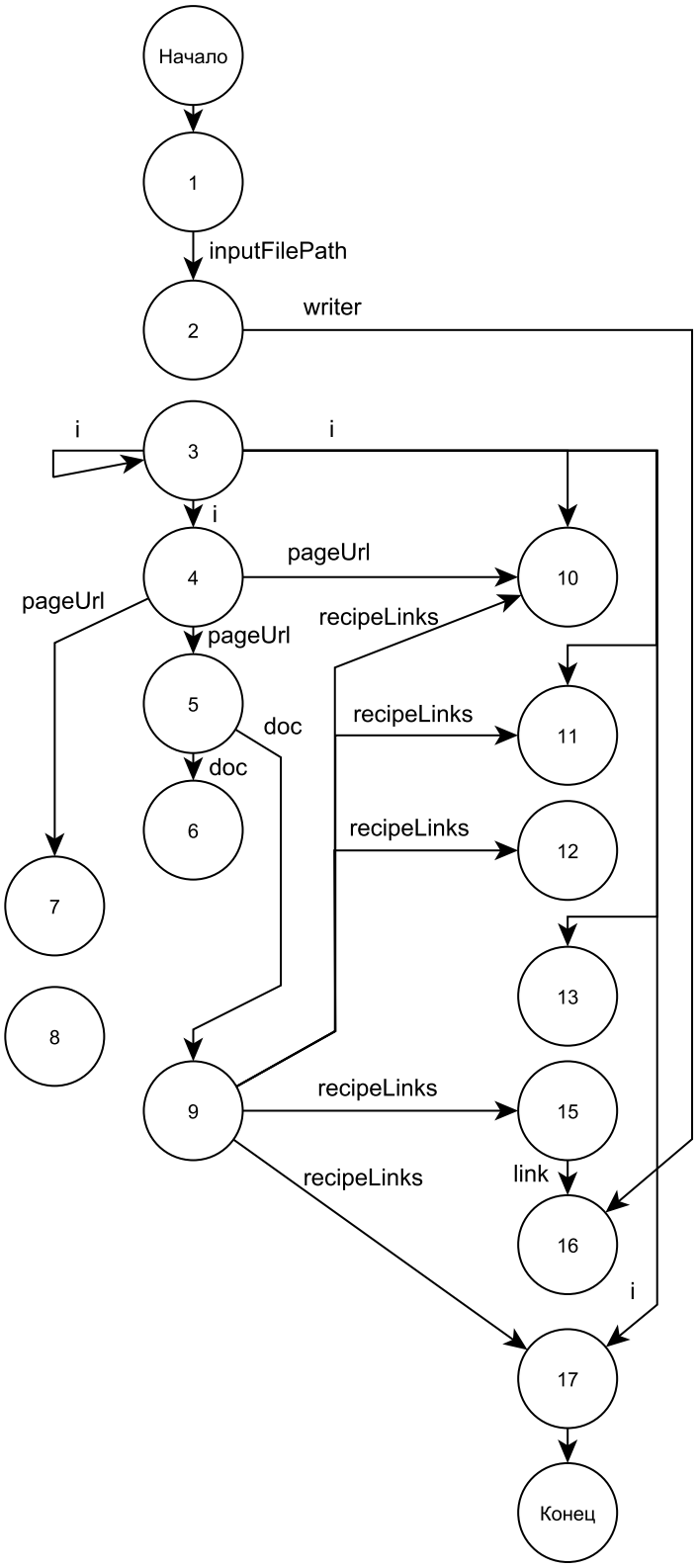


Рисунок 2.1 – Информационный граф

# 3 Информационная история

Информационная история по приведенному листингу кода представлена на рисунке 3.1.

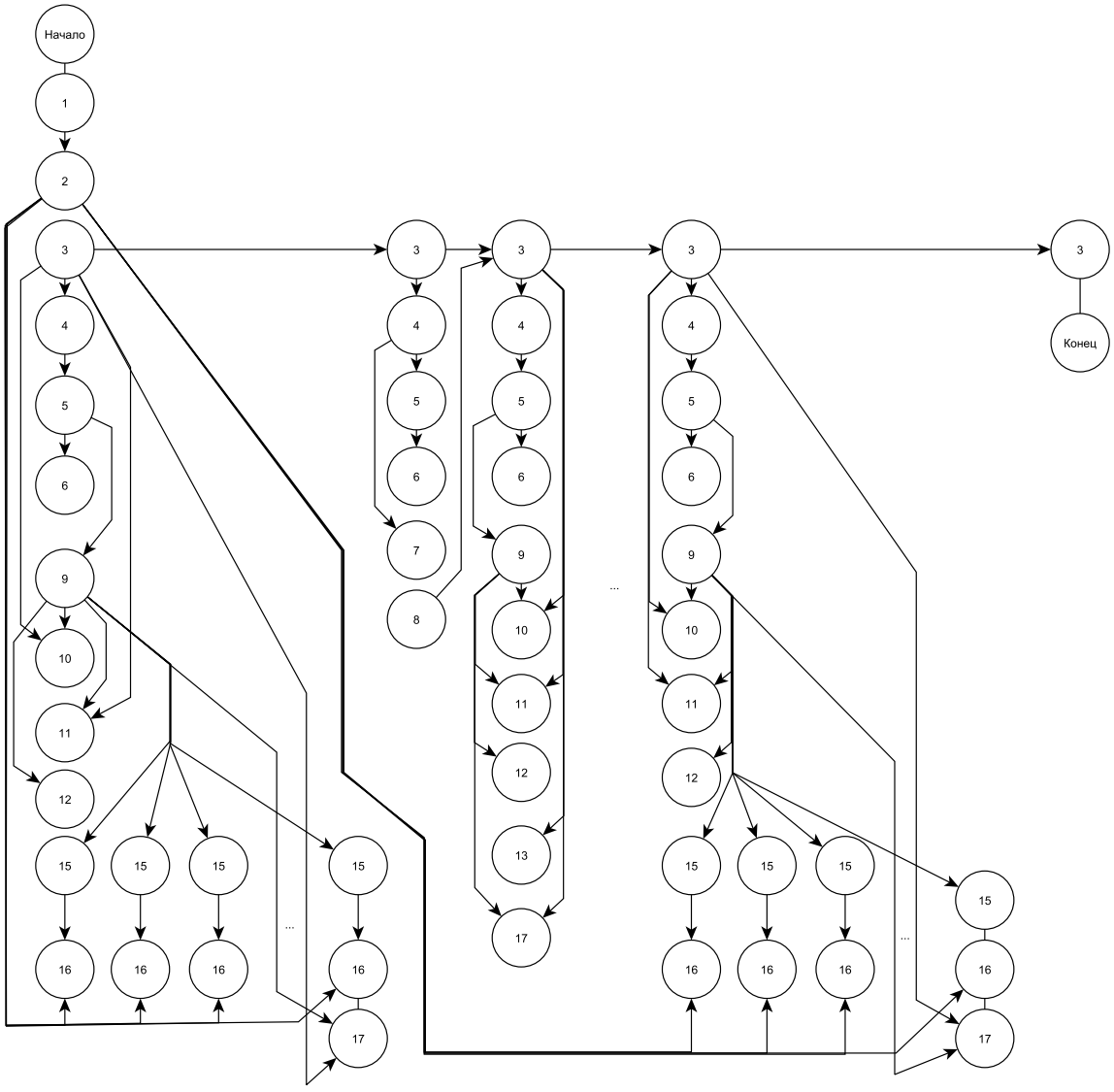


Рисунок 3.1 – Информационная история

# 4 Граф управления

Граф управления по приведенному листингу кода представлен на рисунке 4.1.

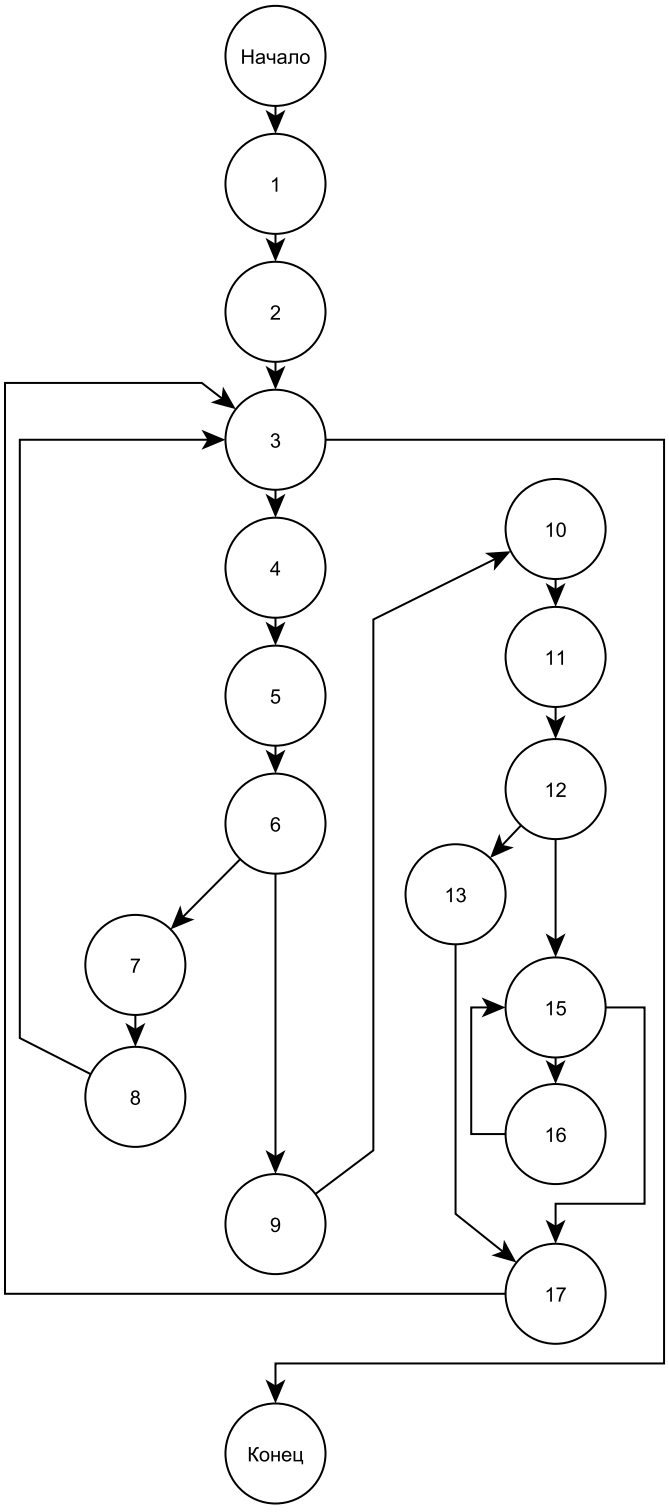


Рисунок 4.1 – Граф управления

# 5 Операционная история

Операционная история [1] по приведенному листингу кода представлена на рисунке 5.1.

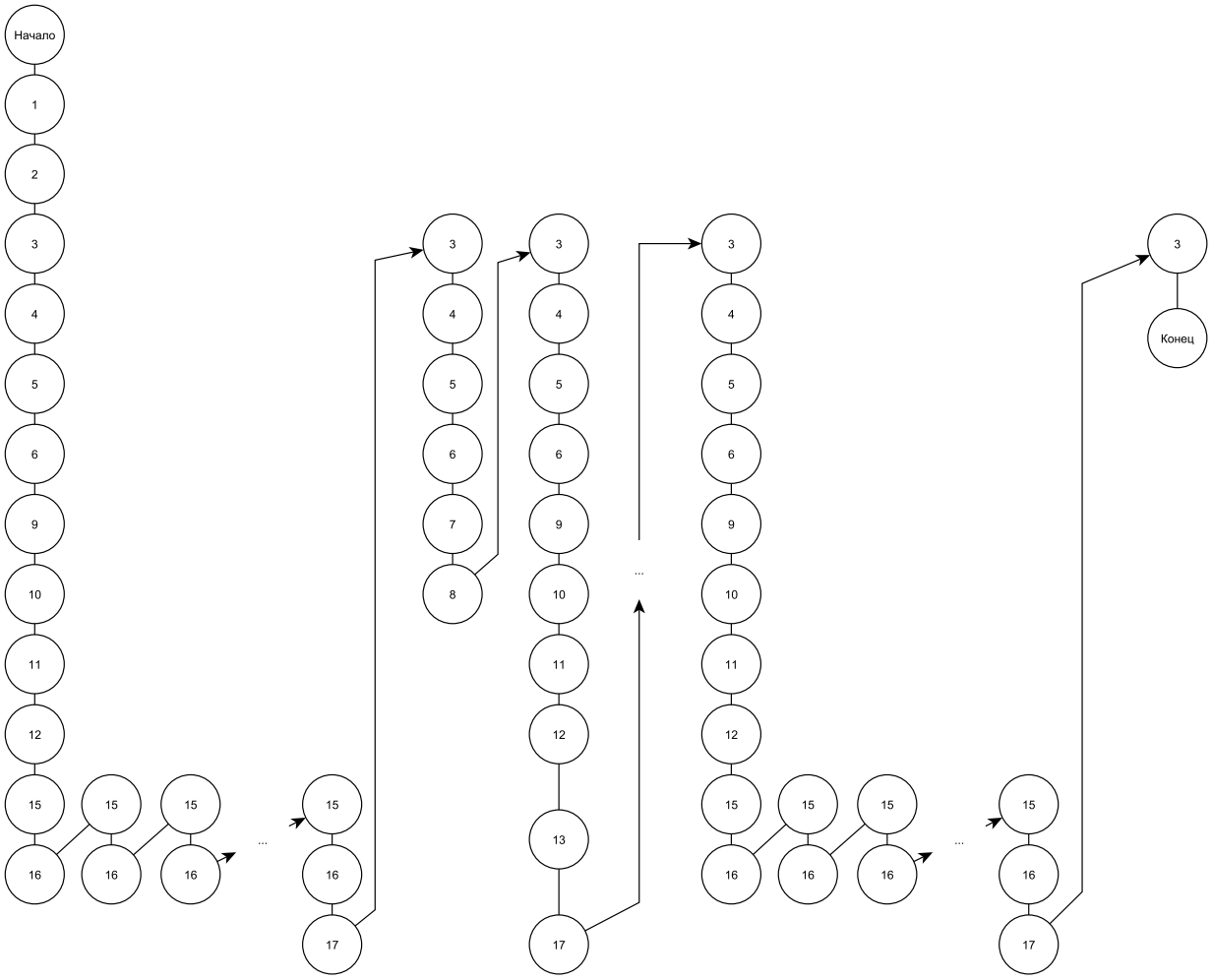


Рисунок 5.1 – Операционная история

## ВЫВОД

Графовые модели позволяют структурированно и наглядно представить логику и поток данных в программе. Информационный граф и информационная история дают понимание, как данные перемещаются и трансформируются, что упрощает отладку и оптимизацию. Граф управления проясняет логику ветвлений и циклов, помогая анализировать корректность и полноту тестирования, а также оценивать потенциальные точки улучшения. Операционная история позво-

ляет рассмотреть пошаговое выполнение кода, полезно для точной диагностики ошибок и оптимизации.

Таким образом, графовые модели упрощают анализ кода, делают структуру и поведение программы более понятными для программистов.

# ЗАКЛЮЧЕНИЕ

Цель лабораторной работы достигнута — получен навык построения и применения графовых моделей для анализа кода.

В ходе достижения поставленной цели были решены следующие задачи:

— построены информационный граф, информационная история, граф управления и операционная история для выбранного фрагмента кода;

— проанализированы построенные графовые модели;

— сделан вывод о применимости графовых моделей к задаче анализа программного кода.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методика построения операционных графовых моделей программ [Электронный ресурс]. URL: `https://izv.etu.ru/assets/files/2_p016-021.pdf` (дата обращения: 15.12.2024)