Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий

**Лабораторная работа №2**

Выполнили:

Бабаев Р.С.

Садовая А.Р.

Абдулов И.А.

Проверил:

Мусаев А.А.

Санкт-Петербург,

2022

Оглавление

[Введение 3](#_Toc117212767)

[1 Решение задач 4](#_Toc117212768)

[1.1 Программа для бинарного поиска 4](#_Toc117212769)

[1.2 Программа для угадывания студента 6](#_Toc117212770)

[1.3 Граф для второй программы 9](#_Toc117212771)

[Заключение 12](#_Toc117212772)

[Список литературы 13](#_Toc117212773)

# **Введение**

Данная работа представляет собой отчет о выполненных заданиях:

1. Написать программу для бинарного поиска, результатом которой является количество шагов, необходимых для нахождения требуемого числа;
2. Написать программу, которая по определенным характеристикам будет угадывать студента;
3. Составить граф для второй программы.

# **Решение задач**

## **1.1 Программа для бинарного поиска**

Бинарный поиск — тип поискового алгоритма, который последовательно делит пополам заранее отсортированный массив данных, чтобы обнаружить нужный элемент. Другие его названия — двоичный поиск, метод половинного деления, дихотомия.

Основная последовательность действий алгоритма выглядит так:

1. Сортируем массив данных;
2. Делим его пополам и находим середину;
3. Сравниваем срединный элемент с заданным искомым элементом;
4. Если искомое число больше среднего — продолжаем поиск в правой части массива (если он отсортирован по возрастанию): делим ее пополам, повторяя пункт 3. Если же заданное число меньше — алгоритм продолжит поиск в левой части массива, снова возвращаясь к пункту 3.

Схема действий данного алгоритма представлена на рисунках 1.1. и 1.2.

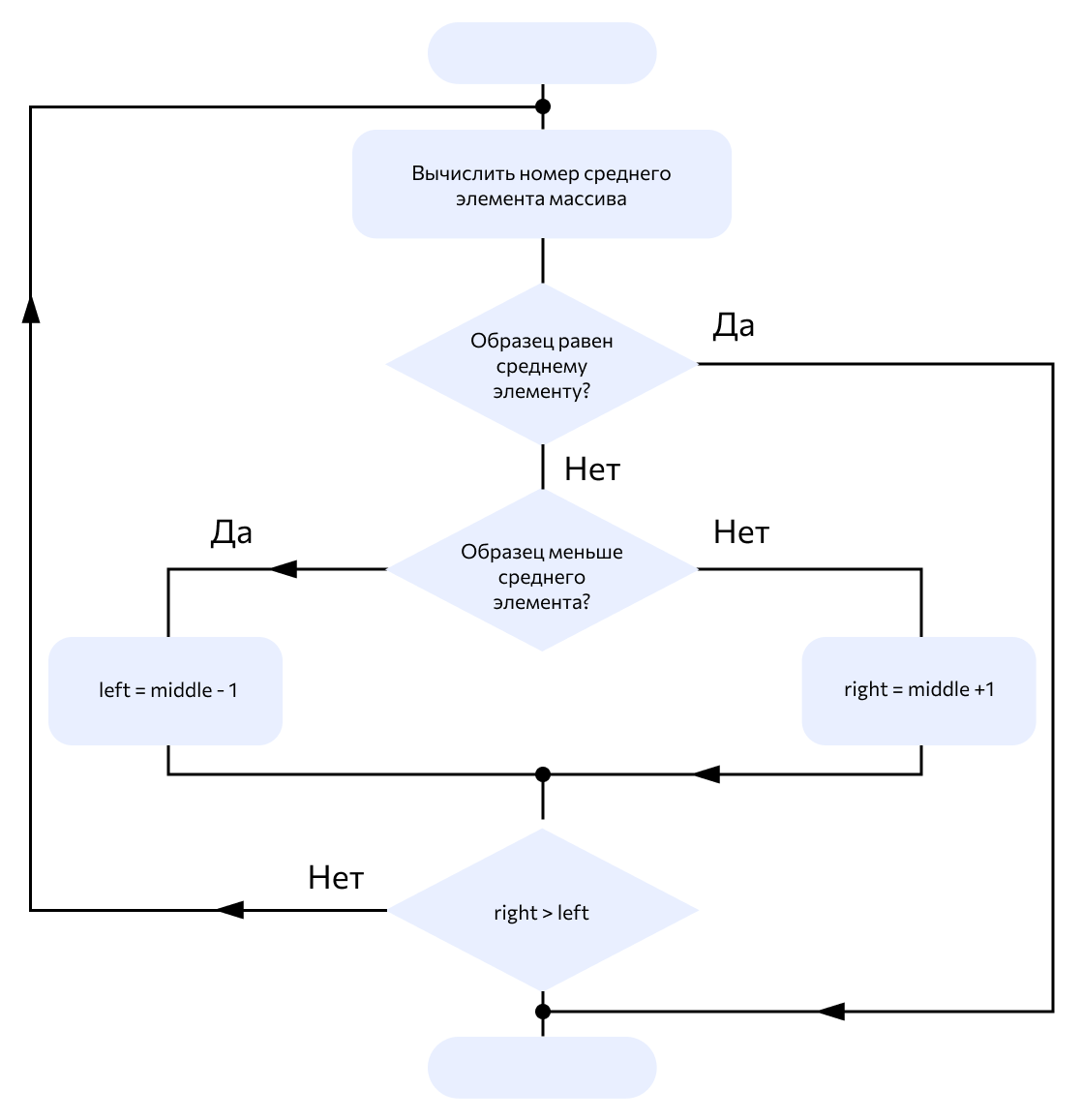


Рисунок 1.1 – Блок-схема бинарного поиска

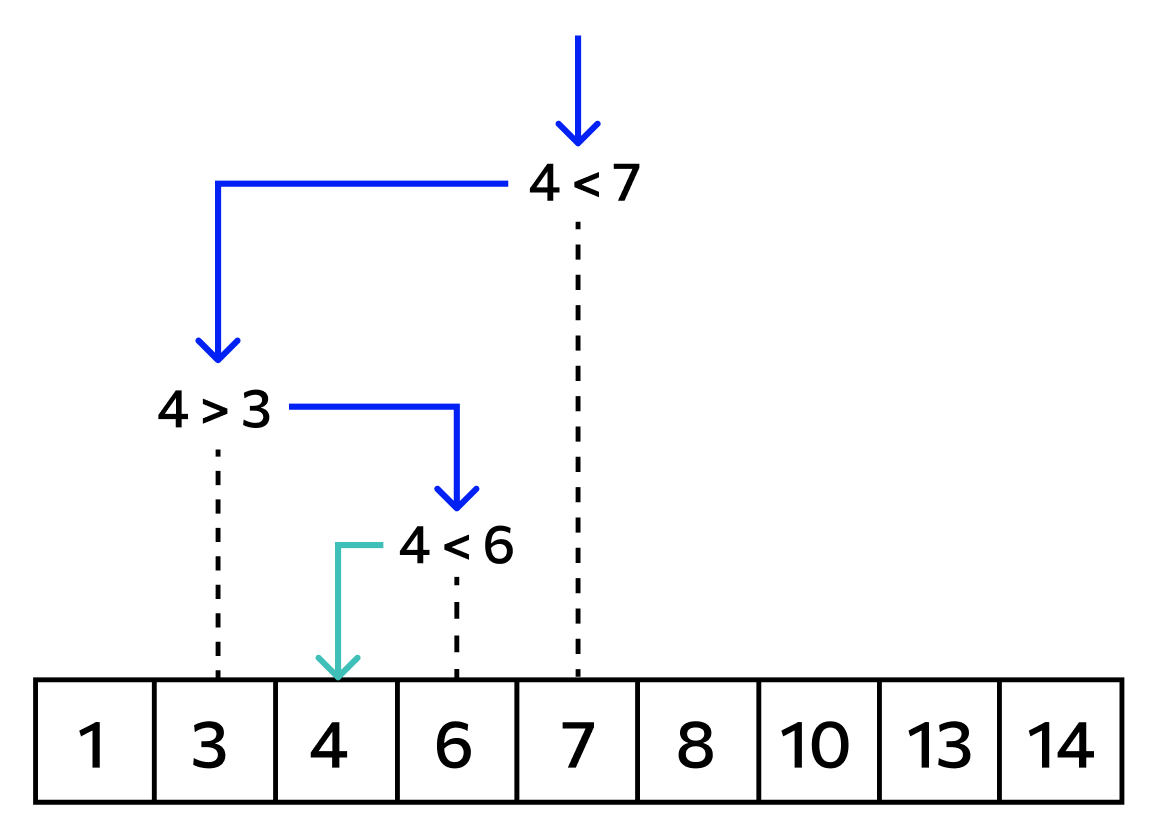


Рисунок 1.2 – Поиск позиции заданного элемента в списке

Существуют два способа реализации бинарного поиска: итерационный (с помощью цикла) и рекурсивный (функция, которая вызывает саму себя). В приведенной программе был реализован рекурсивный метод (см. рисунок 1.3). Функция binary\_search(array, element, start, end), где array – исходный список, element – искомый элемент, start и end – границы поиска, сначала проверяет, не стала ли левая граница правее правой: такой случай означает, что искомого элемента нет в данном списке. Затем, если с границами всё в порядке, программа добавляет шаг (STEP) и индекс среднего элемента в списке. Если искомый элемент равен среднему, то функция выводит количество шагов, необходимых для нахождения данного элемента, иначе – вызывает саму себя с измененными границами.

На рисунке 1.4 представлен пример работы данной программы.

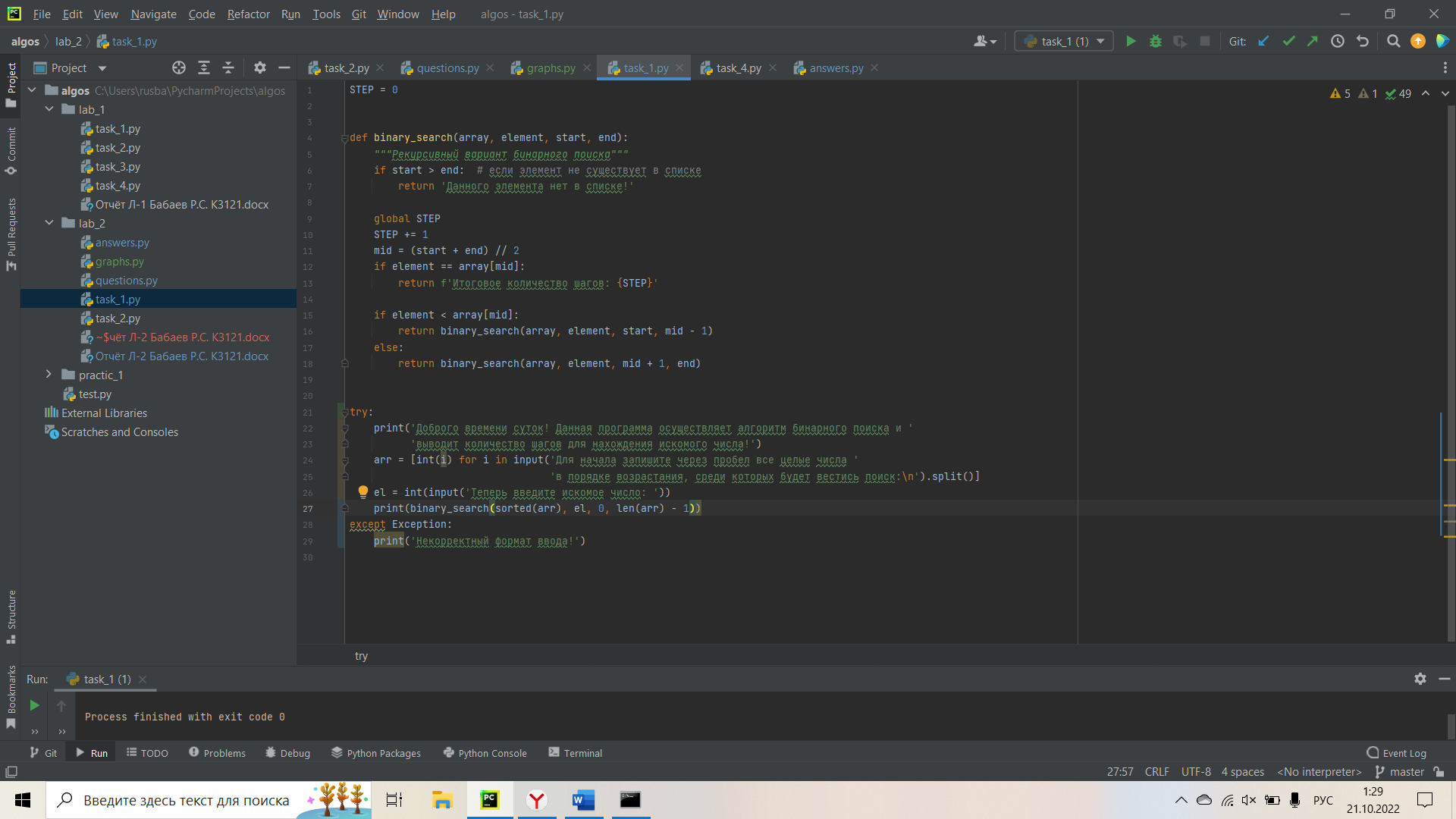


Рисунок 1.3 – Код программы для бинарного поиска

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.4 – Пример работы программы для бинарного поиска

## **1.2 Программа для угадывания студента**

Перед написанием программы был проведен опрос среди студентов группы К3121, на основе которого были созданы список словарей студентов, где ключами выступают темы вопросов, а значениями – их ответы вида «да/нет» (см. рисунок 1.5), а также словарь вопросов с идентичной структурой (см. рисунок 1.6).

Изображение выглядит как текст, монитор, снимок экрана, электроника

Автоматически созданное описание

Рисунок 1.5 – Пример словаря с информацией о студенте

Изображение выглядит как текст, снимок экрана, электроника, монитор

Автоматически созданное описание

Рисунок 1.6 – Словарь с вопросами

Алгоритм угадывания студента достаточно прост: программа задает случайный вопрос из списка и в зависимости от ответа вычеркивает неподходящих студентов до тех пор, пока не останется единственный студент (в таком случае она выведет его имя) или вовсе не останется ни одного студента (в данной ситуации программа сообщит, что студент не найден). Код программы представлен на рисунке 1.7, а примеры ее работы – на рисунках 1.8 и 1.9.

Изображение выглядит как текст, снимок экрана, электроника, монитор

Автоматически созданное описание

Рисунок 1.7 – Пример работы программы для угадывания студента

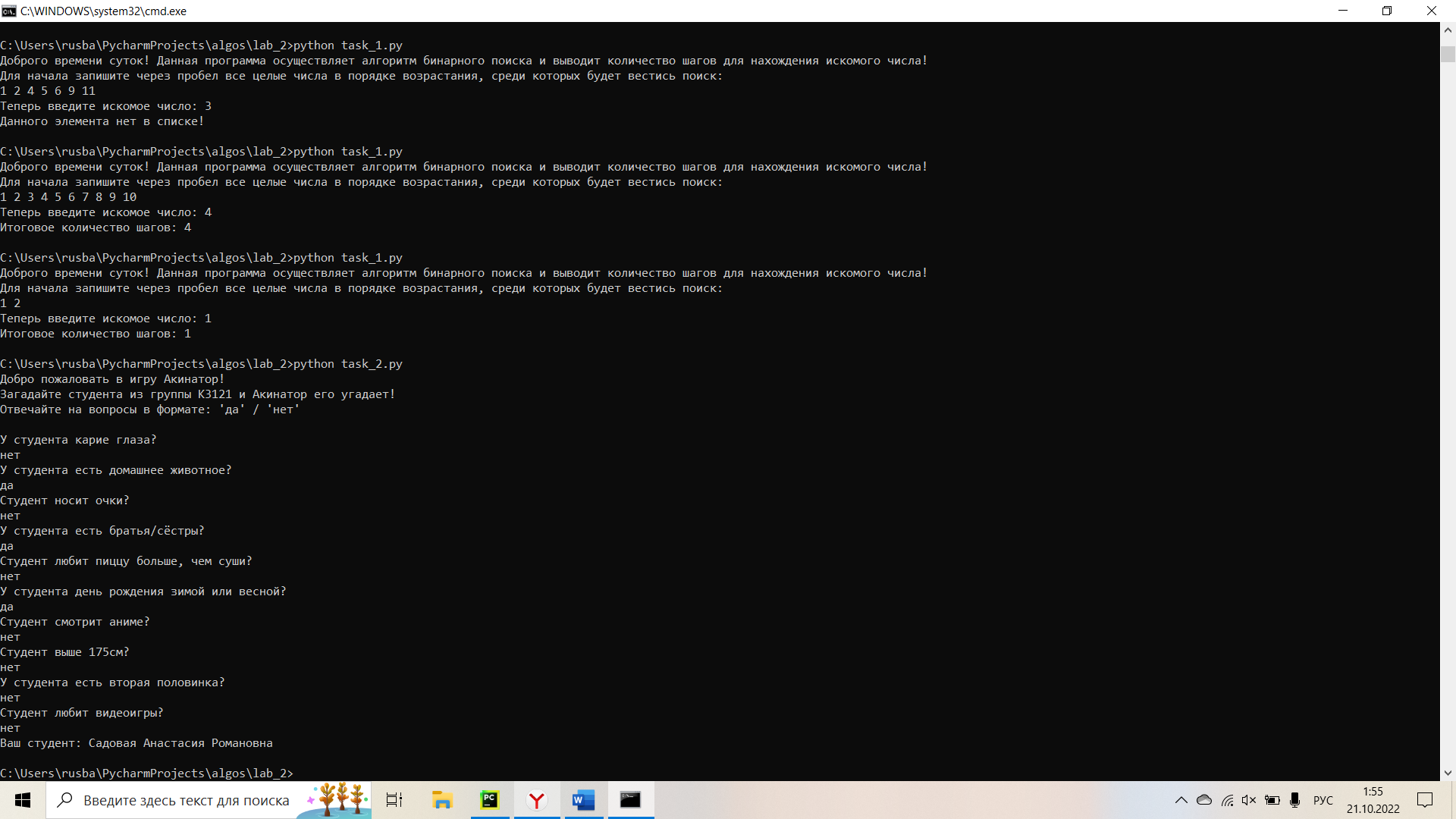


Рисунок 1.8 – Пример работы программы №2 (1)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.9 – Пример работы программы №2 (2)

## **1.3 Граф для второй программы**

Для создания графа была использована библиотека networkx, которая предоставляет довольно удобные инструменты для генерации больших графов. Программа всё так же использует сформированные словари с вопросами и информацией о студентах и с помощью функции add\_edge(f\_item, s\_item, graph) связывает студентов (f\_item) с соответствующими характеристиками (s\_item). Таким образом она проходится по всем выбранным студентам и рисует граф.

С кодом программы можно ознакомиться на рисунке 1.10, а со сформированными ею графами – на рисунках 1.11–1.15.

Изображение выглядит как текст, снимок экрана, монитор, внутренний

Автоматически созданное описание

Рисунок 1.10 – Код программы для генерации графов

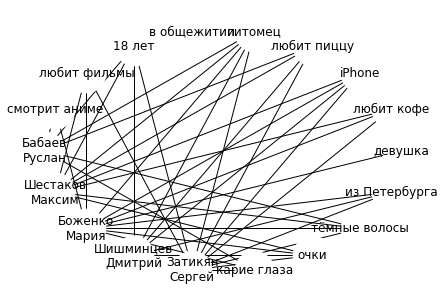


Рисунок 1.11 – Граф №1



Рисунок 1.12 – Граф №2



Рисунок 1.13 – Граф №3



Рисунок 1.14 – Граф №4

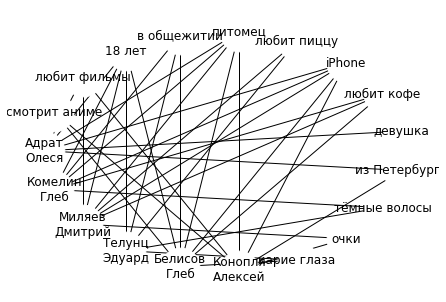


Рисунок 1.15 – Граф №5

# **Заключение**

Таким образом, для каждой задачи была написана программа на языке программирования Python. Был изучен алгоритм бинарного поиска, реализована функция, осуществляющая рекурсивный двоичный поиск элемента из списка (целые числа). Также написана программа, угадывающая студента группы К3121 по ответам пользователя в формате «да/нет». Кроме того, составлены графы для данной программы с помощью библиотеки networkx. Для каждого решения приведены примеры с тестами.

Все программы можно найти на Github [2].

# **Список литературы**

1. SkillFactory.blog [Электронный ресурс] – <https://blog.skillfactory.ru/glossary/binarnyj-poisk/> (Дата обращения – 20.10.2022);
2. Github [Электронный ресурс] – <https://github.com/HeraldOfWar/algos/lab_2> (Дата обращения – 20.10.2022).