МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им.

Н. Э. Баумана

КАФЕДРА ПРОЕКТИРОВАНИЕ И ТЕХНОЛОГИЯ ПРОИЗВОДСТВА

ЭЛЕКТРОННОЙ АППАРАТУРЫ

Отчет о выполнении практического задания №4

«»

по курсу «Функциональная логика и теория алгоритмов»

Выполнил: студент каф. ИУ4-21Б

Степанова Диана Станиславовна

Проверил:

Москва 2024

**Цель работы**

Целью данного кода является создание графа, добавление и удаление вершин, а также сортировка вершин по их степени.

**1. Исходные данные**

**Данные, подающиеся на вход (вводимые с клавиатуры):**

- Пользователь вводит количество ребер графа.

- Выбирается вершина для удаления.

**Данные, которые необходимо вывести:**

- Исходный граф в виде матрицы смежности.

- Граф после удаления выбранной вершины.

- Отсортированный массив вершин по возрастанию степени.

1. **Выполнение**

Данный код на языке программирования C реализует базовые операции с графом, используя матрицу смежности для представления графа. Описание каждой функции и основной части программы:

1. **Graph** - структура данных, представляющая граф. Она содержит матрицу смежности и количество вершин в графе.

2. **initGraph(Graph\* graph, int numVertices)** - функция для инициализации графа. Устанавливает количество вершин и заполняет матрицу смежности нулями.

3. **addEdge(Graph\* graph, int src, int dest)** - функция для добавления ребра между вершинами src и dest в графе. Устанавливает соответствующие значения в матрице смежности.

4. **removeVertex(Graph\* graph, int vertex)** - функция для удаления заданной вершины из графа. Сдвигает все вершины после удаляемой вершины на одну позицию влево и вверх, а затем уменьшает количество вершин в графе.

5. **getDegree(Graph\* graph, int vertex)** - функция для получения степени заданной вершины в графе. Подсчитывает количество инцидентных ей ребер.

6. **sortVerticesByDegreeAsc(Graph\* graph, int\* sortedVertices)** - функция для сортировки вершин по возрастанию степени. Использует функцию getDegree для сравнения степеней вершин.

7. **main()** - основная часть программы, где запрашивается количество ребер, инициализируется граф, добавляются ребра, выводится исходный граф, предлагается удалить вершину, удаляется вершина, выводится измененный граф и затем выводятся отсортированные вершины по возрастанию степени.

1. **Плюсы и минусы:**

**Плюсы:**

- Код является понятным и легко читаемым.

- Реализованы основные операции с графом: добавление, удаление вершин, получение степени вершины.

- Программа работает корректно для небольших графов.

**Минусы:**

- Нет проверки на дублирование ребер при добавлении.

- Удаление вершины не учитывает возможные связи с другими вершинами.

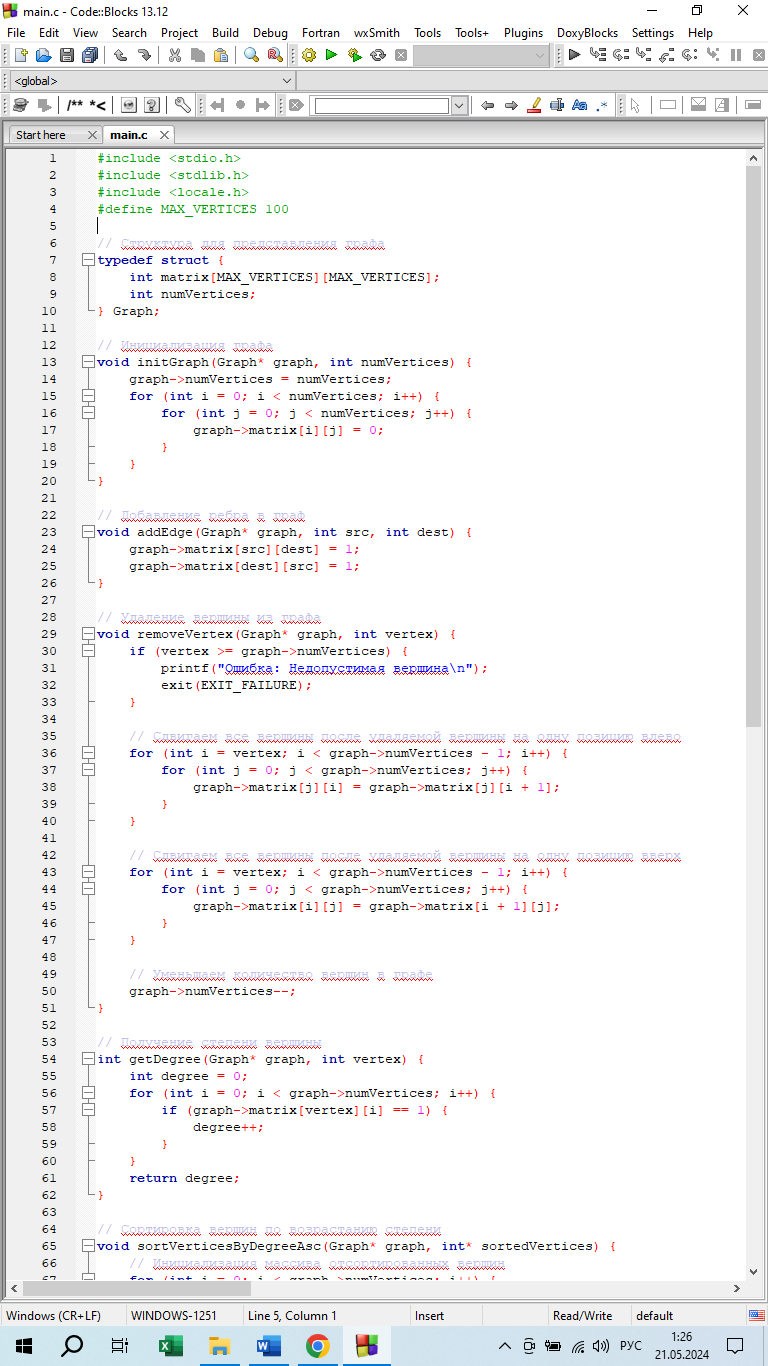
- Алгоритм сортировки вершин по степени может быть неэффективным для больших графов из-за квадратичной сложности.

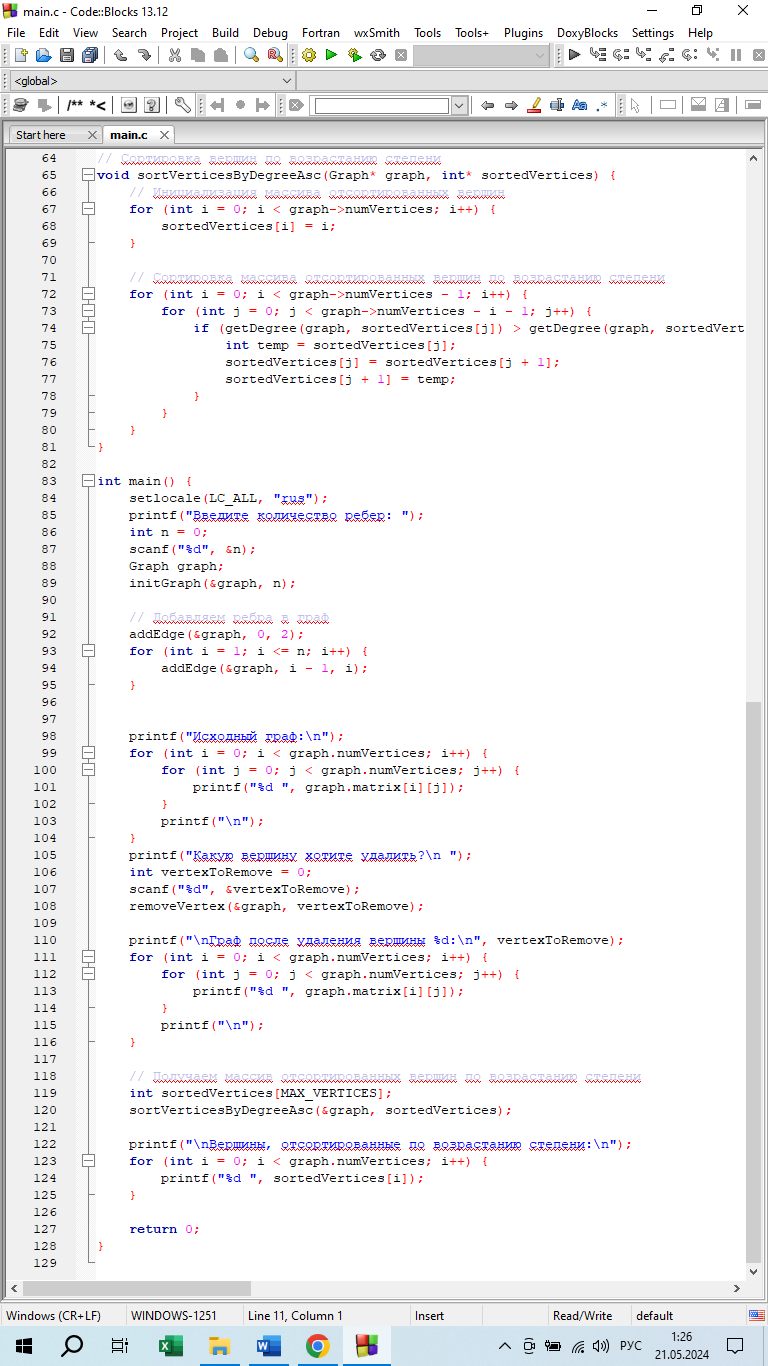
1. **Сложность:**

- Временная сложность добавления и удаления вершины составляет O(V^2), где V - количество вершин.

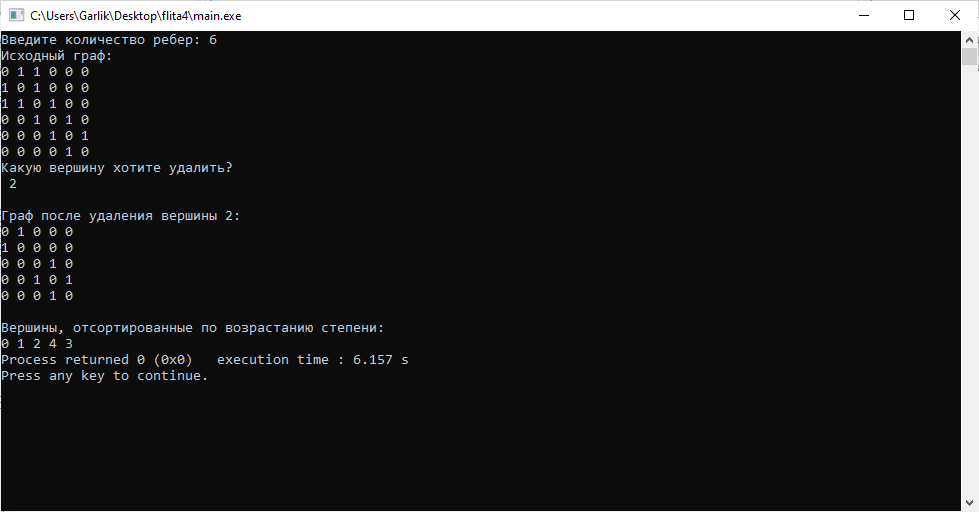
- Временная сложность сортировки вершин по степени - O(V log V).

1. **График эффективности:**
2. **Блок-схема:**
3. **Код:**

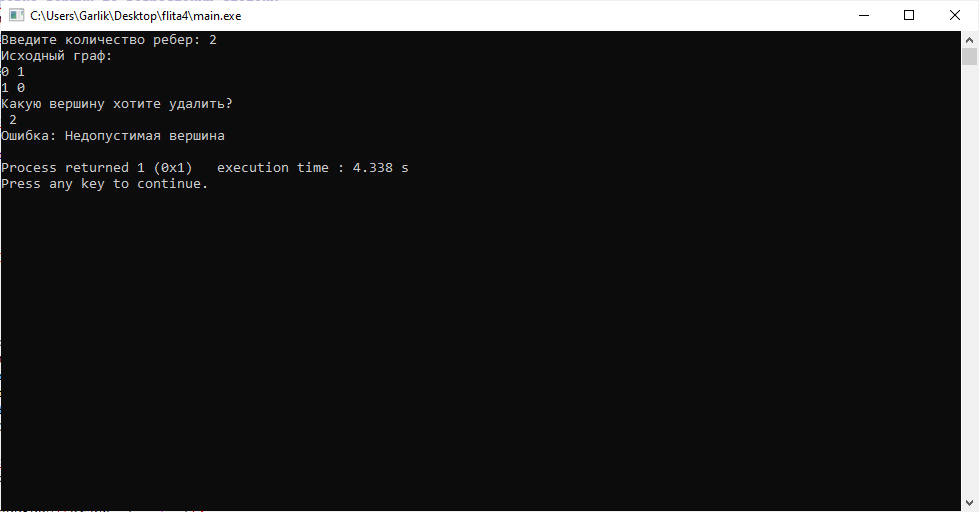




1. **Результаты работы.** **Удачная попытка:**



**Неудачная попытка:**



1. **Вывод**

Программа успешно реализует алгоритмы для работы с графом, включая добавление ребер, удаление вершины и сортировку вершин по степени. Однако, для полноценного использования программы необходимо добавить обработку исключений и проверку входных данных.