**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №8**

**по дисциплине «Объектно-Ориентированное Программирование»**

|  |  |  |
| --- | --- | --- |
| Студенты группы № 2372 |  | Полуянов В.Н., Юрин А.А. |
| Преподаватель |  | Егоров С.С. |

Санкт-Петербург

2023

**Задание**

Разработать GUI приложение, выполняющее функцию визуализации графа состояний.

Граф состояний описывается матрицей, число строк которой равно числу вершин, а число столбцов - числу событий. Элементом i-ой строки и j-го столбца является номер строки (т.е. соответствующая ей вершина графа), которая становится активной при возникновении j-го события, если при этом вершина i была активна.

Основной функцией объекта класса "Интерфейсное окно" является выбор файла, который содержит данные о графе состояний. При чтении файла необходимо проверить корректность данных и в случае обнаружения ошибки необходимо сформировать соответствующее сообщение пользователю.

Номер активной вершины также задается в интерфейсе.

При корректности данных создается объект класса "Граф состояний", устанавливаются (если необходимо) связи между новым объектом и существующими, после чего граф отображается в соответствующем окне (объект класса "Окно представления графа").

Активная вершина помечается цветом. При смене значения номера активной вершины должны происходить изменения в отображении.

В интерфейсе должна быть предусмотрена возможность инициирования любого из возможных событий. При их возникновении должен происходить переход в новую активную вершину, согласно графу, смена значения в интерфейсном окне и его перерисовка.

При выборе в интерфейсе другого графа (другого файла) старый должен заменяться на новый, номер активной вершины принимать исходное (корректное) значение и граф перерисовываться.

Реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленным целям.

**Спецификации классов**

**Class Vertex:**

Атрибуты:

-name – Имя вершины

-neighbors – Вектор указателей на связанные вершины

Методы:

+Vertex(char) – Конструктор вершины с параметром «название»

+addNeighbor(Vertex\*) – Установить связь с новой вершиной

+getName() – Возвращает имя узла

+getNeighbors() - Возвращает вектор указателей на связанные узлы

**Class Graph:**

Атрибуты:

-vertices – Вектор указателей на вершины

Методы:

+Graph() – Конструктор по умолчанию

+setVertices() – Установить вектор вершин

+empty() – Возвращает true если граф пустой

+size() – Возвращает текущий размер графа

+getVertices() – Возвращает список вершин

**Class TParser:**

Атрибуты:

-\*graph - Указатель на граф

Методы:

-getPath() – Получить у пользователя путь к файлу

-parseFile() – Считать данные с файла

-showError() – Вывести окно с ошибкой

+TParser() – Конструктор

**Class TSample:**

Атрибуты:

#\*graph - Указатель на граф

Методы:

+TSample() – Конструктор

+changeSource() – Выбрать новый файл

+empty() – Возвращает true если граф пустой

+draw() – Рисует граф

**Class TSampleSwitcher : public TSample:**

Атрибуты:

-activeIndex – Индекс текущей активной вершины

Методы:

+TSampleSwitcher() – Конструктор

+changeSourse() – Выбрать новый файл

+updateActive() – Изменить активную вершину

+draw() – Рисует граф состояний

**Class TInterface:**

Атрибуты:

-\*btnUpdFile – Кнопка «выбрать новый файл»

-\*sample – Указатель на рисунок

Методы:

+TInterface() – Конструктор

+~TInterface() – Деструктор

#paintEvent() – Событие отрисовки графа

#mousePressEvent() – Событие нажатия кнопки мыши

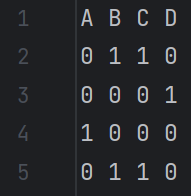
+updFile() – Выбрать новый файл

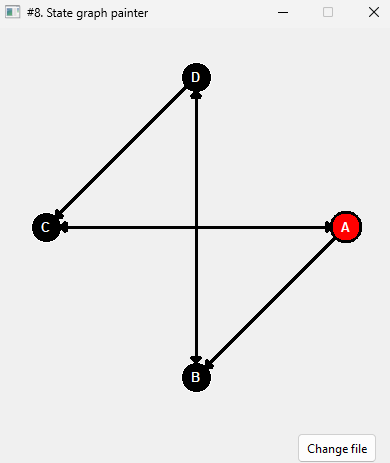
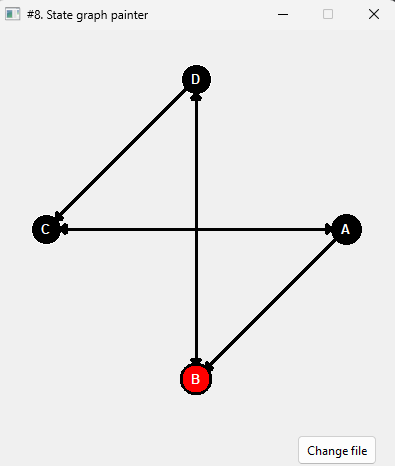
**Диаграмма классов, дополненная атрибутами и методами**

|  |
| --- |
|  |
|  |
|  |
|  |

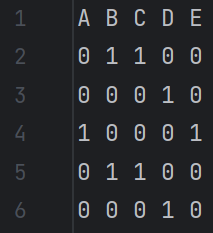
**Скриншоты работы программы на контрольных примерах. Два примера**

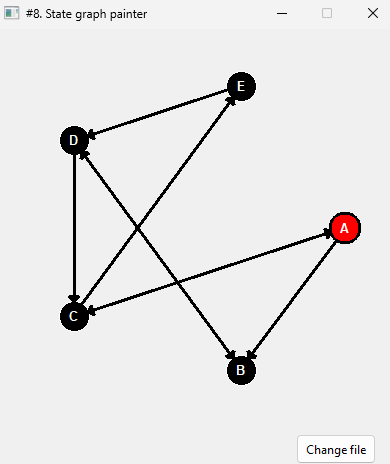
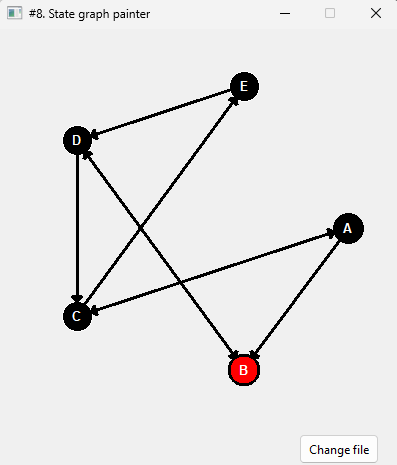
1. *Первый пример*



1. *Второй пример*



**Вывод**

В ходе работы было разработано GUI приложение для визуализации ориентированного графа состояний. Пользователь выбирает файл с данными о графе, при чтении файла проверяется корректность данных. Если данные корректны, создаётся объект класса Граф, затем граф передаётся окну представления и визуализируется. При нажатии левой кнопки мыши активной вершиной становится последний по очереди добавления сосед текущей активной вершины, при нажатии правой – ближайший сосед (первый по очереди добавления).