Министерство образования и науки РФ

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра Информационные технологии и автоматизированные системы

Лабораторная работа № 2

Тема: «Введение в теорию графов. Алгоритмы Дейкстры и Флойда»

Выполнил: студент группы РИС-22-2б

Мизёв В.А. ф

Проверил: доцент кафедры ИТАС

Полякова О.А.

г. Пермь – 2024

1. **Цель работы**

Получить практические навыки работы с графами

1. **Постановка задачи**

Реализовать следующие алгоритмы:

- обход в ширину (bfs);

- обход в глубину (dfs);

- алгоритм Дейкстры.

Требования к заданию:

- разработать графический интерфейс – инструмент по решению разработчика;

- визуализировать граф с использованием графической библиотеки;

- необходимые инструменты для работы с графами (редактирование графа: создание, добавление и удаление вершины и рёбер, редактирование весов рёбер и редактирование матрицы смежности).

1. **Анализ задачи**

Для визуализации графа будет использоваться SFML – это мультимедийная библиотека, которая предоставляет возможность работать с графикой, звуками и сетью в своей обёртке.

Для графического интерфейса будет использоваться TGUI – это кроссплатформенная современная библиотека графического интерфейса на C++. Данная библиотека предоставляет использование разного backend’a, в числе которых есть SFML, то есть для данной работы TGUI будет использоваться в связке с SFML.

Основная идея в реализации визуализации графов состоит из написания следующих классов: Graph, Vertex и Edge, где Graph выступает в роле менеджера графа (методы с загрузкой, сохранением графа в файл, добавление вершин), Vertex в свою очередь описывает вершину, главные методы которой это привязка вершин друг к другу (добавление ребра) и Edge – класс для визуализации ребра, веса и направления.

Также будет использоваться связующий класс Core, данный класс будет работать с SFML, где будут обрабатываться все нажатия и отрисовываться примитивы.

Основные методы класса Edge продемонстрированы на рисунке 1.

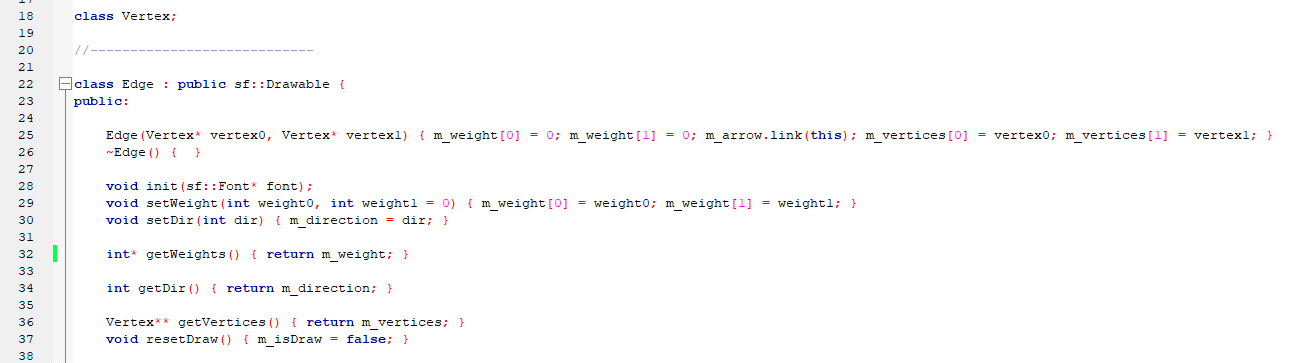


Рисунок 1 - Класс Edge

Как видно из 1 рисунка, конструктор класса принимает два аргумента, это вершины, которые будут связаны ребром.

Также для инициализации шрифта используется метод init().

Метод setWeight() - устанавливает вес ребру. Само ребро может иметь несколько направлений, для этой реализации используется массив из двух элементов, поэтому метод с установкой веса принимает 2 аргумента.

Метод setDir() – устанавливает направление.

Основные методы класса Vertex продемонстрированы на рисунке 2.

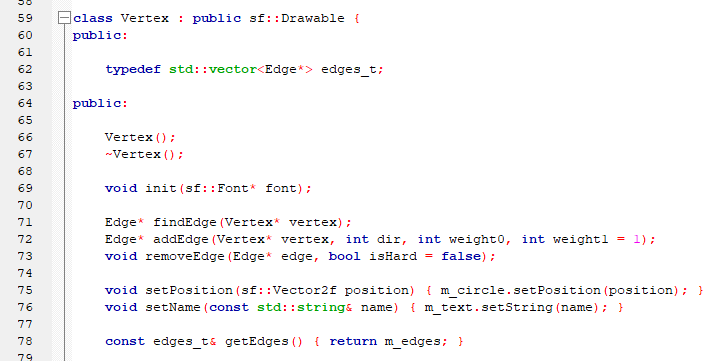


Рисунок 2 - Класс Vertex

Также как и у класса Edge, для класса Vertex предусмотрен метод с инициализацией шрифта, который используется для отображения имени вершины.

Метод findEdge() ищет ребро, которое связано с указанной вершиной и вершиной, где используется вызов данного метода.

Метод addEdge() добавляет ребро, в качестве аргументов он принимает вершину, направление и 2 веса.

Метод removeEdge() – удаляет ребро, причём есть несколько удалений, мягкое и жёсткое, мягкое – удаляет связь ребра с данной вершиной и связанной вершиной, жёсткое же удаляет все связи указанного ребра и удаляет его из памяти.

Основные методы класса Graph продемонстрированы на рисунке 3.

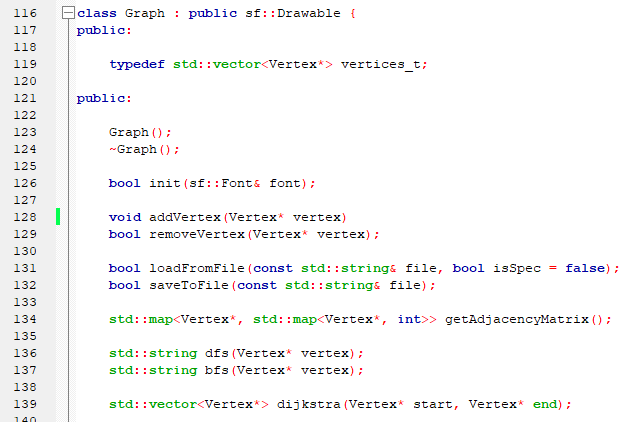


Рисунок 3 - Класс Graph

Метод addVertex() добавляет вершину в граф.

Метод removeVertex() удаляет вершину из графа.

Метод loadFromFile() загружает граф из файла, также предусмотрена загрузка из памяти, для этого необходимо взвести флаг isSpec в true.

Метод saveToFile() сохраняет граф в файл.

getAdjacencyMatrix() – возвращает матрицу смежности графа.

Методы dfs() и bfs() принимают в качестве аргумента стартовую позицию для обхода в глубину и ширину соответственно.

Метод dijkstra(), в качестве аргумента принимает стартовую вершину и завершающую вершину, для поиска пути, если путь не найден, то в векторе будет лежать одна вершина.

1. **Результаты работы программы**

Основное окно программы продемонстрировано на рисунке 4.

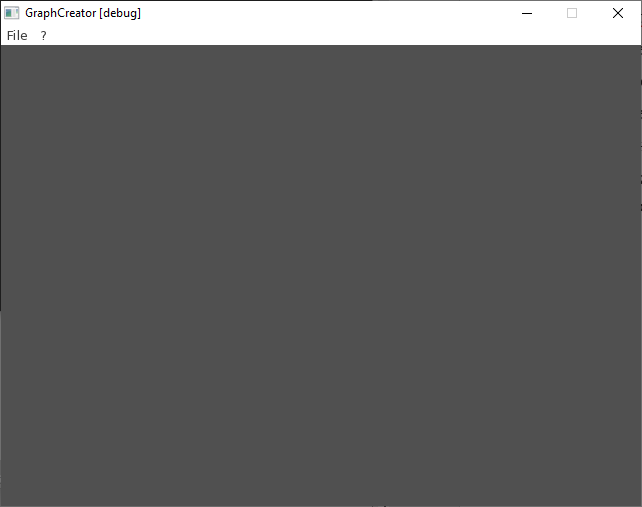


Рисунок 4 - Окно программы

Для создания вершины необходимо вызвать всплывающее меню правой кнопкой мыши. Данное действие продемонстрированно на рисунке 5.

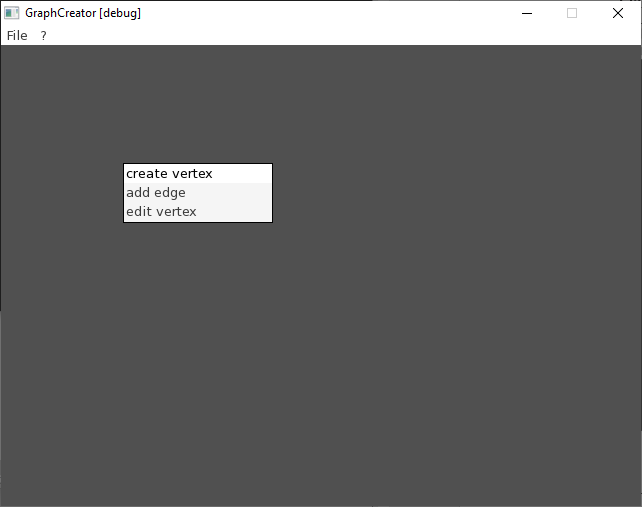


Рисунок 5 - Всплывающее меню

Также есть возможность фокусироваться на одну из вершин и перемещать её, зажимая левую кнопку мыши. Если данная вершина в фокусе, то её контур сменит цвет на красный. Данное действие продемонстрировано на рисунке 6.

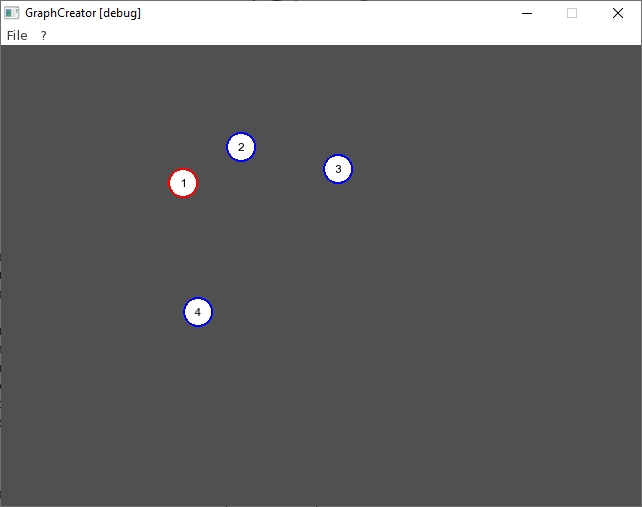


Рисунок 6 - Перемещение вершин

Для соединения вершин рёбрами необходимо вызвать вплывающее меню и выбрать пункт «add edge», после выбора данного пункта отобразиться окно для ввода, которое продемонстрировано на рисунке 7.

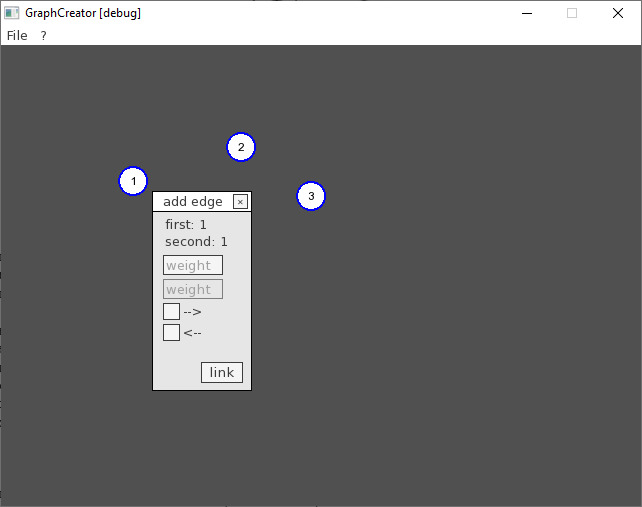


Рисунок 7 - Добавление связей между вершинами

Для добавления соответствующего ребра необходимо выбрать нужные вершины по очереди, выбранные вершины будут отображаться в поле first и second, где first первая вершина от куда будет направление, second вершина в которую будет путь, также для ориентированного графа предусмотрены стрелки, для их выбора необходимо выбрать нужный «CheckBox», относительно него включится дополнительный пункт с введением веса ребра, вес рёбер вводится в поля «EditBox», как только будут заполнены все поля необходимо нажать кнопку «link», который свяжет два ребра по введённым параметрам.

На рисунке 8 продемонстрированы все типы путей, которые возможно создать для ориентированного графа.

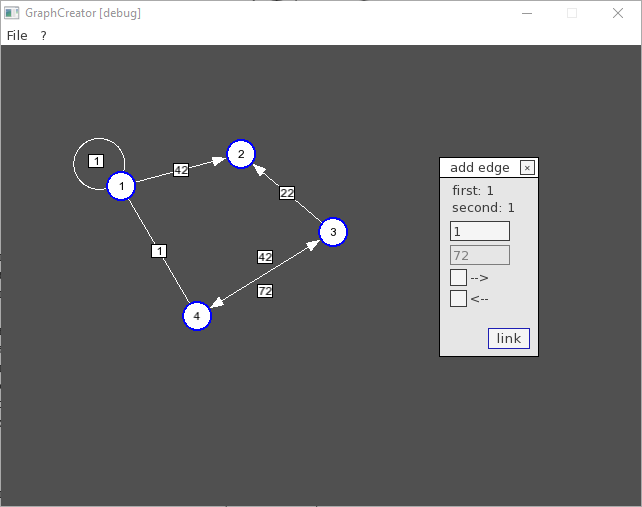


Рисунок 8 - Создание связей между вершинами

Для удаления связей между вершинами в контекстном меню есть пункт «edit vertex», который вызывает окно, в нём возможно удалить все связанные рёбра с данной вершинной, также для редактирования всех связей предусмотрена работа с матрицей смежности.

На рисунке 9 продемонстрировано окно редактирования вершины.

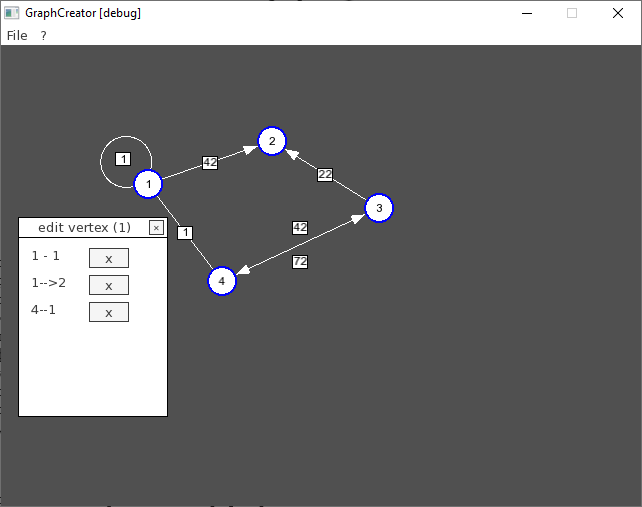


Рисунок 9 - Окно редактирования вершины

Для вызова матрицы смежности, необходимо навестись мышью на меню бар, вызвать контекстное меню File, и из него выбрать Matrix.

На рисунке 10 продемонстрировано окно с матрицей смежности.

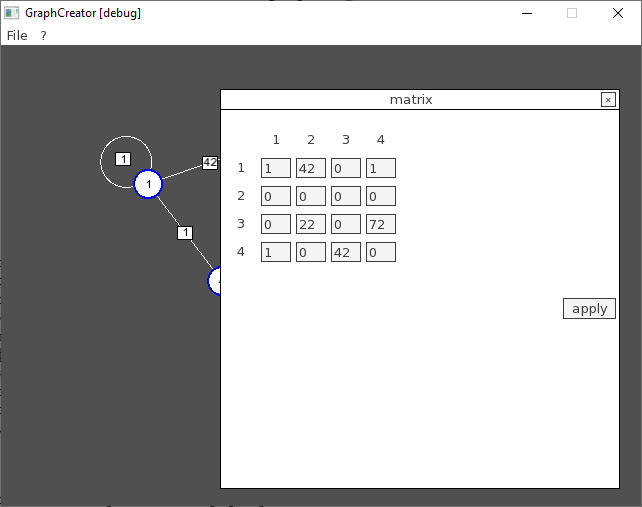


Рисунок 10 - Матрица смежности

Алгоритм поиска в глубину вызывается на F1, при этом берётся стартовая вершина, которая была выделена пользователем.

Алгоритм поиска в ширину вызывается на F2, аналогично с алгоритмом поиска в глубину берётся стартовая вершина.

Алгоритм Дейкстры, ждёт на вход две вершины, для вызова поиска пути необходимо нажать на F3.

Все три алгоритма показывают последовательность вершин слева внизу в формате «1 2 3», за исключением алгоритма Дейкстры, если данный алгоритм не нашёл пути, то выводится «path not found!», если же путь найден, то он выводится в формате «1->2->3», означающий, что из вершины 2 в вершину 3 есть вот такой наикротчайший путь.

1. **Тестирование алгоритмов. Задание вариант 1.**

Дан граф, приведённый на рисунке 11. Необходимо выполнить обход графа в глубину (выполнение начать с 1 вершины).

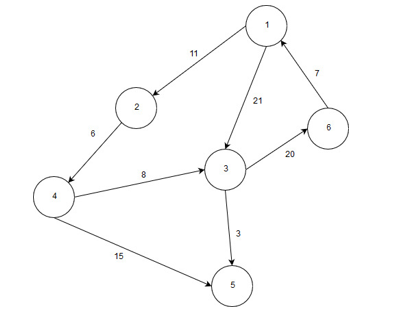


Рисунок 11 - Задание 1

Выполнение задания приведено на рисунке 12.

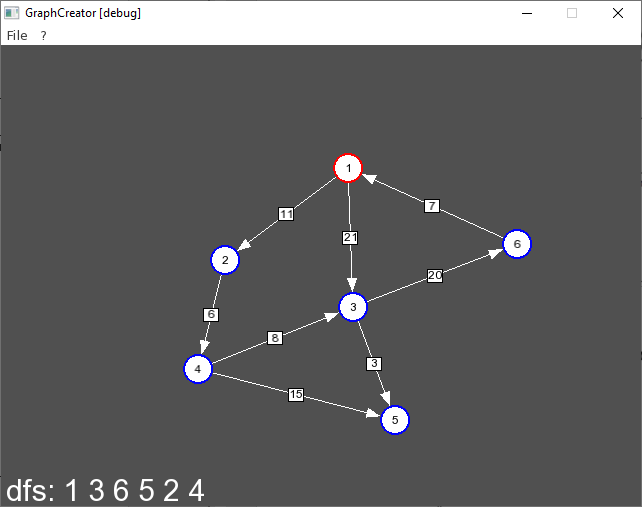


Рисунок 12 - Результат выполнения программы

**Заключение**

В результате работы был реализован построитель графов, который может искать наикротчайшие пути из одной вершины в другую, обходить граф в глубину и ширину, а также имеет функционал удаления, добавления и редактирования вершин и рёбер. Цель получить практический навык работы с графами достигнута.