|  |  |
| --- | --- |
| Программа для определения кратчайшего пути из одной точки в другую для заданной схемы сообщения маршрута. | |
| Внутренняя спецификация | |
| Студент | Аде-Гива М. Д |
| Преподаватель | доц. Сычев О.А. |
| Сдано |  |
| Лабораторная работа №2 |  |

# Общие сведения

Наименование программы – «Программа для вычисления кратчайшего пути между двумя точками маршрутных сообщений» (далее «Программа»).

Для функционирования программы необходима ОС Windows 7 и более поздней версии.

Программа должна быть написана на языках С/С++ с использованием библиотеки Qt 5.12.

1. Описание логической структуры программы
   1. Алгоритм программы

Из командной строки программа получает путь к двум текстововым файлам, первый содержит название точек маршрутных сообщений и матрицу смежности между этими точками, а второй содержит точки от и до которых ищем кратчайший путь. После прочтения содержимого файлов производится следующее:

1. Проверка содержимого файлов на наличие ошибок.
2. Если ошибок не обнаружено, то производится создание модели маршрутных сообщений в виде направленного взвешенного графа.
3. Из начальной точки производится просчёт расстояний до всех других точек по алгоритму Дейкстры.
4. Извлекается расстояние до нужной нам точки.
5. По обратному алгоритму производится расчёт кратчайшего маршрута до нужной нам точки, на основании рассчитанных расстояний.
6. Полученное расстояние и путь записывается в выходной файл Результат работы программы или сообщение об ошибке во входных

данных записывается в текстовый файл, путь к которому передан в виде аргумента командной строки.

* 1. Декомпозиция программы

Основные типы и структуры данных программы представлены в приложении А.

Выделенные подпрограммы (функции) представлены в приложении Б. Иерархия вызовов подпрограмм представлена в приложении В. Диаграмма потоков данных программы представлена в приложении Г.

Приложение А Основные типы и структуры данных программы

## В приложении перечисляются основные структуры данных, используемых в программе.

*Класс Graph содержит следующие приватные поля*

/// <summary>

/// Последовательный список имён всех точек графа

/// </summary> QStringList pointNames;

/// <summary>

/// Взвешенная матрица смежности

/// </summary>

QList<QList<int>> distancesMatrix;

/// <summary>

/// Индекс точки, откуда производился расчёт по алгоритму Дейкстры

/// </summary>

int originPointIndex;

/// <summary>

/// Имя точки, откуда производился расчёт по алгоритму Дейкстры

/// </summary>

QString originPointName;

/// <summary>

/// Список меток, рассчитанных алгоритмом Дейкстры для конкретной точки

/// </summary> QList<int> pointLabels;

# Приложение Б

Выделенные подпрограммы

## В приложении приводятся заголовки и псевдокод функций программы.

/// <summary>

/// Инициализирует граф

/// </summary>

/// <param name="lines">На первой строке список точек из файла, на других матрица расстояний</param>

Graph(QStringList lines);

Псевдокод функции Graph:

Записываем список точек из первой линии lines, разделяя её по символу ‘;’ Каждую последующую линию:

Разделяем линии на числа по символу ‘;’

Записываем эти числа как следующая строка матрицы смежности

/// <summary>

/// Возвращает дистанцию от точки до точки в графе

/// </summary>

/// <param name="pointName">Название точки</param>

/// <returns>Расстояние от указанной точки до расчётной. Если точки не соедененны, возвращает -1</returns>

int getDistanceTo(QString fromPointName, QString toPointName);

Псевдокод функции getDistanceTo:

Если метки были рассчитаны не для fromPointName, то заново рассчитать метки Получаем индекс toPointName

По индексу возвращаем значение метки

/// <summary>

/// Возвращает полный кратчайший путь от точки до точки в графе

/// </summary>

/// <param name="pointName"></param>

/// <returns>Список имён точек, которые представляют кратчайший путь. Если точки не соедененны, возвращает пустой список</returns>

QStringList getMinPathTo(QString fromPointName, QString toPointName);

Псевдокод функции getMinPathTo:

Если метки были рассчитаны не для fromPointName, то заново рассчитать метки Получаем индекс toPointName, называем его текущим

Пока текущий индекс не равен индексу fromPointName: Добавляем в список пути имя текущей точки Получаем список расстояний до текущей точки Перебираем метки каждый точки:

Если метка текущей точки минус расстояние до текущей точки равна метки перебираемой точки, то текущая точки теперь равна перебираемой точке

Возвращаем получившийся кратчайший путь

/// <summary>

/// Расчитывает расстояние до всех точек графа из определенной точки

/// </summary>

/// <param name="pointName">Точка, до куда расчитывать расстояние</param> void calculateLabelsFromPoint(QString pointName);

Псевдокод функции calculateLabelsFromPoint:

Находим индекс точки pointName

Очищаем и заполняем список меток, где метка нашей точки равна 0, а других максимально возможному числу

Пока не посетим все точки по одному разу: Перебираем все метки точек:

Если мы эту точку не посещали, то добавляем её индекс в список индексов, а её метку в список меток

Ищем индекс точки с минимальной меткой из списка меток Получаем расстояния до всех остальных точек из найденной точки

Если расстояние не равно 0 (есть связь) и расстояние до точки из текущей меньше, чем её метка, то меняем её метку на расстояние до неё

Помечаем, что посетили текущую точку

/// <summary>

/// Возвращает числовой индекс точки по её имени

/// </summary>

/// <param name="pointName">Имя точки</param>

/// <returns>Индекс точки. -1 если такой точки не существует в графе</returns> int getPointIndex(QString pointName);

Псевдокод функции getPointIndex:

Перебираем имена всех точек, сохраняя индекс:

Если имя совпало, то возвращаем сохраненный индекс точки

/// <summary>

/// Находит индекс минимального значения в списке

/// </summary>

/// <param name="list">Список значений</param>

/// <returns>Индекс минимального значения</returns> int getIndexOfMin(QList<int> list);

Псевдокод функции getIndexOfMin:

Объявляем минимальное значение, равное первому элементу Объявляем минимальный индекс, равный нулю

Перебираем значения из списка:

Если минимальное значение больше, чем текущее: Присваиваем минимальному индексу текущий Присваиваем минимальному значению текущее

Возвращаем минимальный индекс

/// <summary>

/// Возвращает дистанцию до всех точек из указанной

/// </summary>

/// <param name="toPoint">Индекс точки, из которой ищем дистанцию</param>

/// <returns>Список дистанций до всех точек</returns> QList<int> getConnectedPoints(int toPoint);

Псевдокод функции getConnectedPoints:

Перебираем каждую строчку матрицы смежности:

Добавляем в список дистанций элемент toPoint из текущей строчки Возвращаем список дистанций

/// <summary>

/// Считывает файл по строкам

/// </summary>

/// <param name="absolutePath">Абсолютный путь до файла</param>

/// <returns>Считанные строки файла</returns> QStringList readFile(QString absolutePath)

Псевдокод функции readFile:

Открываем файл по указанному пути Пока не дошли до конца файла:

Считываем строку и записываем в лист Закрываем файл для чтения и возвращаем лист строк

/// <summary>

/// Записывает в файл строки (с перезаписью)

/// </summary>

/// <param name="absolutePath">Абсолютный путь до файла</param>

/// <param name="lines">Строки для записи в файл</param> void writeToFile(QString absolutePath, QStringList lines)

Псевдокод функции writeToFile: Открываем файл по указанному пути Для каждой линии lines:

Записываем её в файл Закрываем файл для записи

/// <summary>

/// Преобразует относительный путь в абсолютный

/// </summary>

/// <param name="path">Относительный путь, в начале должна стоять точка</param>

/// <returns>Абсолютный путь</returns> QString getPath(char\* path)

Псевдокод функции getPath:

Если в начале строки точка, то добавляем вместо неё путь до папки, откуда запускаем приложение

Возвращаем путь

/// <summary>

/// Точка входа в программу

/// </summary>

int main(int argc, char \*argv[])

Псевдокод функции main:

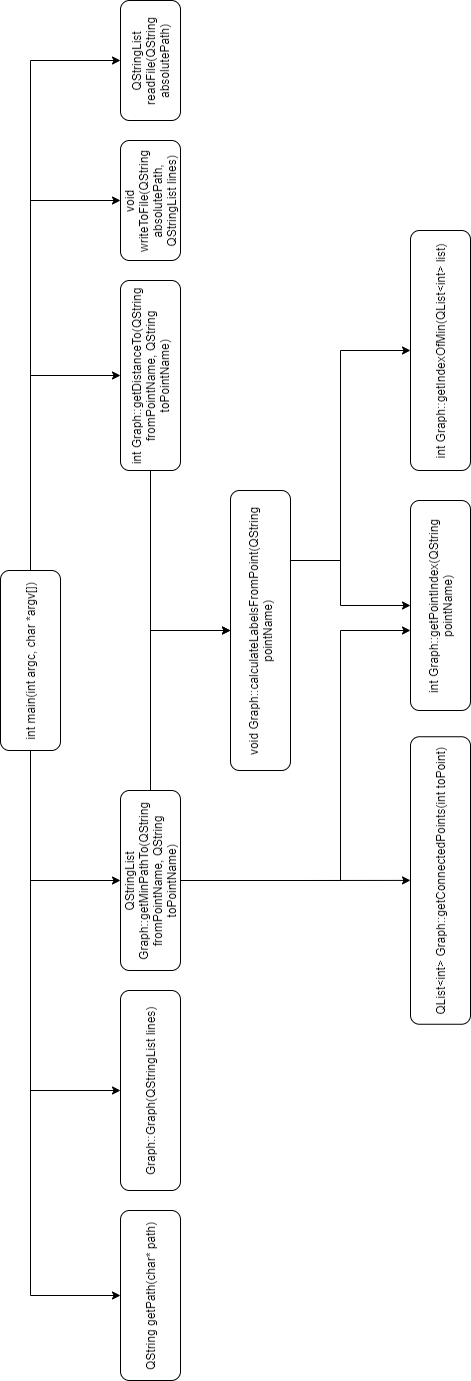
Считываем входные файлы, проверяя их на ошибки Создаем модель графа с помощью считанного файла графа Получаем минимальную дистанцию из точки до точки Получаем кратчайший путь из точки до точки

Если дистанция равна -1, то выводим, что путь отсутствует

Если нет, то записываем в входной файл дистанцию и кратчайший путь

# Приложение В

Иерархия вызовов подпрограмм



Приложение Г

Диаграмма потоков данных программы

