**Основные структуры и процедуры**

В программе реализованы две структуры данных, одна из них описывает город и все его свойства, а другая – карту городов. Структура описывающая город содержит в себе следующие методы:

1. Буфер. Предназначен для хранения текущего значения расстояния от заданного города. В процессе вычисления оптимального маршрута корректируется, если появилось значение дистанции от начального города меньшее, чем хранящееся в буфере.
2. Указатель на структуру. В дальнейшем будет использоваться для указывания сам на себя, чтобы корректно заполнить карту городов.
3. Указатель на указатель на город. В дальнейшем будет использоваться, как указатель на массив указателей, указывающих на города соседи – с которыми есть прямой путь.
4. Существует целочисленный массив, который будет использоваться для хранения расстояния между данным городом и соседним. Связь расстояния с конкретным городом, хранящимся в массиве указателей на соседние города, осуществляется через индекс в данных массивах. То есть важно на этапе формирования городов и карты городов правильно заполнить массив соседей, и правильно заполнить массив расстояний между соседями.
5. Также в структуре есть переменная, хранящая в себе символ, данный символ есть имя города, которое будет дано ему в процессе заполнения карты городов и соседей конкретного города.
6. Существует и целочисленная переменная, хранящая в себе число соседей. Структура города максимально рассчитана на 20 городов соседей. Это значение достаточно легко менять в коде программы, инициализировав новым существующую там константу.

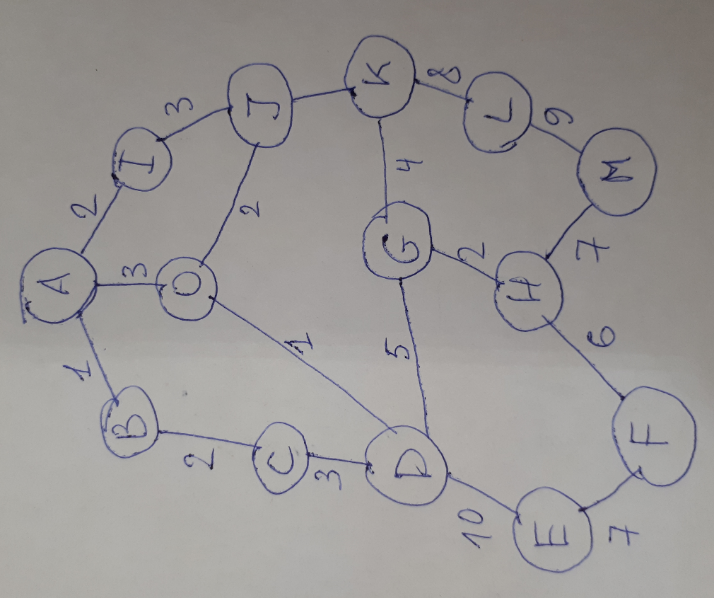
Вторая структура –это карта городов. Содержит в себе массив городов, представленный вектором.

На первом этапе работы программы происходит считывание с файла набора пар городов и расстояний между ними, создание городов и «нанесение их на карту».

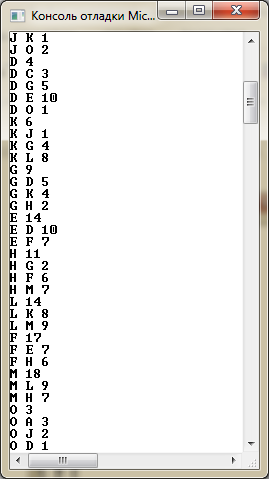
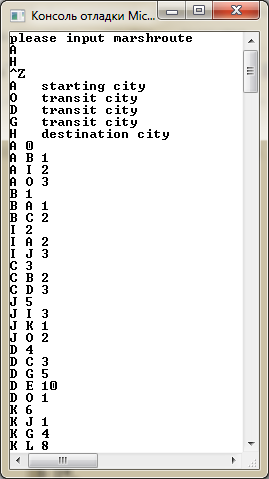
Данный процесс происходит следующим образом: считываются из текстового файла три значения– два символа: названия городов, и целое число: расстояние между ними. Далее в структуре карты городов проверяется существуют ли уже в массиве городов города с указанными именами, если какого-либо города нет, то он создаются с помощью создания указателя на конструктор структуры города с параметром символа-имени, который инициализирует все его методы. И далее в массив указателей на города-соседи добавляется указатель на соседа, то же самое производится с массивом указателей на города соседи самого города-соседа. И в их массивы хранящие расстояния между ними добавляется расстояние между ними. Таким образом производится обработка паты городов и уточнение всех их данных обусловленных их соседством-расстояния и указателя на соседа. Далее город, которого нет в карте вносится в карту: добавляется в вектор городов с помощью функции pushback(). Если же города есть в карте городов, то производится только корректировка их соседских отношений путём внесения указателя на новый город в массив указателей на соседей и расстояния-в массив расстояний между соседями каждого, находящегося в карте городов, города, чьё имя совпало с вновь считанным именем из потока ввода данных. В нашем случае поток ввода устанавливается с текстовым файлом, и считывание данных производится из него. В представленной программе эту работу выполняет процедура settown, которая получает на вход по ссылке карту городов; два символа обозначающие имена городов, и целочисленное значение.

После того, как будет заполнена карта городов, можно приступать к вводу маршрута между городами, и вычислению оптимального маршрута и вывода его на экран. Для этого в программе предусмотрен ряд процедур: процедура очистки буферов каждого города, находящегося в карте городов – clearroute, на её вход по ссылке поступает карта городов; процедура вывода на экран готового маршрута – printtrack, на её вход по ссылке поступает карта городов, кроме неё два символа-исходный город и город адресат; также процедура вычисления оптимального расстояния до всех городов от заданного города - marsh, на вход которой поступает ссылка на карту городов и символ-название начального города. Прописана также процедура вывода на экран списка городов с их соседями, с расстоянием до каждого соседа и выводом значения буфера, которое и является минимальным расстоянием от заданного города до города адресата, если уже производилось вычисление маршрута без очистки буферов. Данная процедура применяется для отладки программы и наглядной демонстрации правильности работы.

**Принцип работы программы для следующей карты городов:**



Рассмотрим вариант, когда необходимо добраться из пункта A в пункт H. Программа выдаёт следующий результат:



Чтобы получить данный результат буферу города А присвоено значение 0, в то время как буферы остальных городов содержат число -1, это число показывает, что расстояние до данного города ещё не вычислено.

Далее запускается двойной цикл по количеству всех существующих городов в карте городов, кроме исходного города. В работе двойного цикла для каждого города не являющегося исходным городом запускается ещё один цикл по количеству соседей данного города. В этом цикле проверяется буфер соседа, и если он не равен -1 и сумма буфера соседа и расстояния до соседа меньше записанного расстояния в буфер рассматриваемого города, то значение буфера меняется на вновь вычисленное. Таким образом цикл за циклом происходит заполнение буферов каждого города минимальным расстоянием от него до исходного города. Исходный город выступает своего рода константой и его буфер не меняется. Можно образно представить за аналог расходящиеся по поверхности воды круги от упавшей капли, где точкой падения капли и будет исходный город. И так круг за кругом вычисляется минимальное расстояние по всей карте городов.

Двойной цикл по количеству городов выбран, чтобы реализовать вычисление маршрута для случая, когда города вытянуты в цепочку, хотя если при вводе анализировать среднее количество городов соседей и вывести параметр, то можно сократить количество вычислений путём замены верхнего предела двойного цикла по количеству городов значением вычисленного параметра.

В приведённом примере в дополнительной распечатке городов видно напротив названия города А цифру 0, это и есть значение буфера, которое показывает расстояние до исходного города в данном конкретном вычислении, то есть расстояние из А в А равно 0. Если упростить выведенный результат в дополнительной распечатке и убрать все города, которые не присутствовали в маршруте и мини списки соседей, то получим следующее:

А 0

O 3

D 4

G 9

H 11

Что равносильно тому, что минимальное расстояние от A до О будет 3, от А до D будет 4, от A до G будет 9 и от А до H будет 11. Сверившись с изображённой картой, и с распечаткой городов, где также указаны все расстояния между городами-соседями, убеждаемся в правильности представления в программе карты городов, и в правильности вычисления минимального расстояния от исходного города до любого города на карте.

Процедура вывода маршрута действует следующим образом: сначала в списке городов на карте определяется город адресат по его названию, его название вносится в вектор символов. Далее в цикле проверяется условие равенства разницы значения буфера рассматриваемого города и расстояния до соседа со значением буфером соседа. Если условие верно, это значит, что этот город-сосед находится на оптимальном маршруте. Его название вносится в вектор символов. Далее исполняется такой же цикл, только уже по городам-соседям вновь найденного города. Эти вычисления производятся до тех пор, пока одним из соседей вновь исследуемого города не окажется исходный город, это проверяется по совпадению имён. Далее достаточно вывести на экран в обратном порядке список символов из вектора и предусмотреть соответствующий комментарий.

А чтобы организовать обработку более сложного маршрута, состоящего из более чем одного города адресата, организовывается цикл, выходом из которого служит значение конца ввода Ctrl+Z для Виндовс. В данном цикле берётся пара городов, следующих друг за другом в введённом маршруте и для них проделываются описанные выше действия, выводится маршрут. Далее запускается процедура очистки карты городов, которая присваивает буферам каждого города значение -1. Далее в рассмотрение берётся следующая пара городов: один выступавший городом адресатом в предыдущем вычислении, другой-следующий из списка ввода. В результате отработки программы на экран будет выведен оптимальный маршрут. Задача решена.