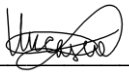
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информатики и прикладной математики
Кафедра прикладной математики и экономико-математических методов

КУРСОВАЯ РАБОТА
по дисциплине:
«Системы компьютерной математики»

Тема: «Реализация алгоритма для создания температурных карт в системе Wolfram Mathematica»

Направление: 01.03.02 Прикладная математика и информатика

Студент: Иксанов Марат Васильевич
Группа: ПМ-2001 Подпись 

Проверил: Фридман Григорий Морицович
Должность: д.т.н., профессор

Оценка: ОТЛИЧНО Дата: 01.06.2021

Подпись: 

Санкт-Петербург
2021 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Тепловые карты – один из способов графического отображения данных, в котором каждому индивидуальному значению соответствует свой цвет, который может быть выбран пользователем произвольно.

Цель данной курсовой работы – научиться создавать температурные карты, основанные на метеоданных, полученных от метеостанций различных стран мира при помощи системы Wolfram Mathematica [1].

Основные аспекты данных карт – это формирование базы данных, основанной на данных метеостанций, их преобразование для дальнейшей работы, создание корректной покрываемой оболочки для выбранной пользователем страны, визуализация данных, взятых из сформированной базы.

1. ПОЛУЧЕНИЕ ВХОДНЫХ ДАННЫХ И ИНФОРМАЦИИ О МЕТЕОСТАНЦИЯХ СТРАНЫ

При создании температурных карт пользователь в самом начале задает ряд параметров:

- название страны, которое будет храниться в переменной $country$;

- дата начала и конца выбранного промежутка временного промежутка, которые хранятся в переменных $dateStart$ и $dateEnd$ соответственно. Данные переменные хранятся в виде списков, где первый элемент списка – год, второй элемент списка – месяц, третий элемент списка – день;

- тип временного промежутка (например, день, месяц, год), по которому будет преобразовываться база данных, хранимый в переменной k.

Создается функция $getStations$, которая принимает на вход переменную $country$. Эта функция обрабатывает данные, хранимые в $country$: находит все расположенные на территории выбранной пользователем страны метеостанции и их координаты.

Следующим шагом в переменные $stations$ и $stationsToCoordinates$ присваиваются найденные функцией $getStations$ станции и список ассоциаций, где ключи – метеостанции и значения – их координаты для каждой переменной соответственно.

На рисунке 1 изображен результат применения описанных выше действий.

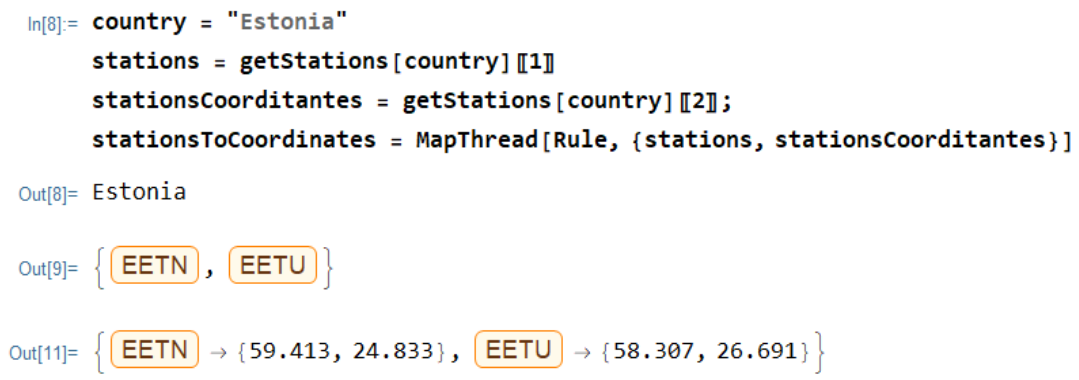


Рисунок 1 – Применение функции $getStations$

2. СОЗДАНИЕ ПЛОТНОЙ СЕТКИ ТОЧЕК

2.1. Первичная выборка основных точек

Для образования первичной сетки необходимо создать список точек, которые будут лежать в ней. На первом шаге создается список крайних точек выбранной страны. На следующем шаге выбирается минимумы и максимумы среди всего списка для координат каждой из осей. Создаются два списка точек $gridFirst$ и $gridSecond$. Каждый список строится с помощью функции $Table$, где на каждой итерации создается список – пара чисел – точка. Крайние точки, от и до которых ходит итератор выбираются из максимумов и минимумов, найденных раньше. Шаг между каждой точкой определяется переменной rad, которая может изменяться в зависимости от желаемой плотности сетки (по умолчанию ее значение равно 0.5). $gridSecond$ является смещенной сеткой $gridFirst$ на параметр $rad/2$.

Конечным шагом является объединение полученных списков и создание первичной сетки $grid$.

На рисунках 2, 3, 4 визуализированы $gridSecond$, $gridFirst$ и $grid$ для $country = "Estonia"$

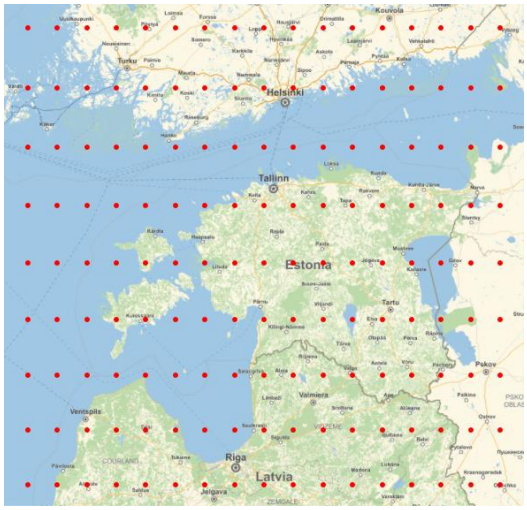


Рисунок 2 – Список точек $gridSecond$

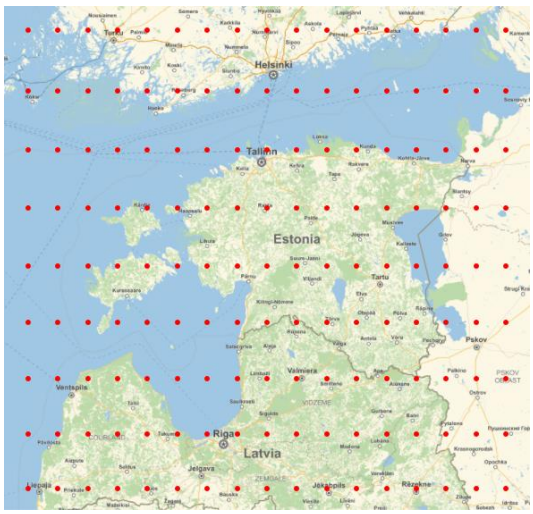


Рисунок 3 – Список точек $gridFirst$

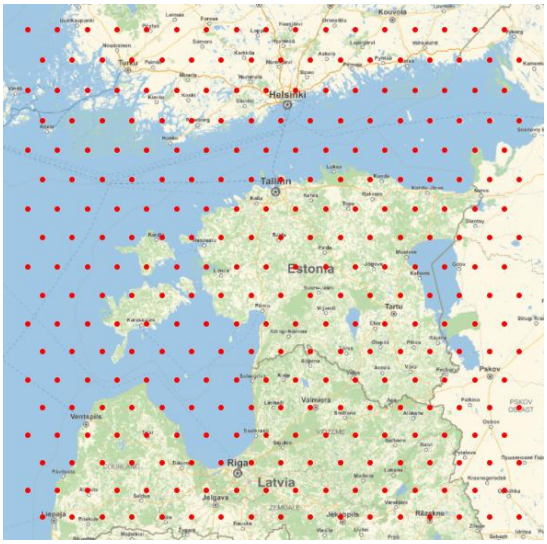


Рисунок 4 – Итоговая первичная сетка $grid$

2.2. Создание оболочки

В первичной сетке для выбранной выше $country = "Estonia"$ большое количество точек из $grid$ находятся вне границ выбранной страны. Для решения этой проблемы создается минимально выпуклая оболочка [2] страны с помощью функции $ConvexHullMesh$ по координатам крайних точек, найденных в пункте 3.1. Полученная оболочка записывается в переменную ob. Данная функция создает минимальную выпуклую оболочку, в которой заключены все заданные точки.

Поскольку крайние точки выбираются по некоторому количеству крайних точек выбранной страны, для более корректной дальнейшей работы стоит немного увеличить в размерах полученную оболочку. С помощью функции

$RegionResize$ оболочка ob способна увеличиться. Для корректного увеличения выбирается параметр l , который представляет собой значение максимального расстояния между крайними координатами на каждой из осей. l хранит в себе минимальную длину стороны квадрата, в котором заключена оболочка ob.

В функции $RegionResize$ параметров увеличения выбирается значение $l * 1.4$, с целью незначительно увеличить размеры оболочки, чтобы в ней хранилось минимальное количество точек, которые лежат за границами страны. Значение 1.4 для параметра выбрано эмпирически. При меньших значениях у некоторых стран полученная оболочка не покрывает всю территорию выбранной страны. При работе с конкретными странами, данный параметр может изменяться.

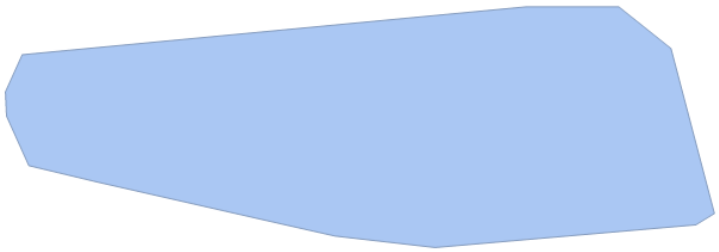


Рисунок 5 – Пример оболочки ob, повернутой на 90 градусов

2.3. Создание итоговой плотной сетки точек

В результате действий из пункта 3.1 и 3.2 выбираются точки из $grid$, которые принадлежат оболочке. В конечном счете получается преобразованный список $grid$, который является итоговой сеткой для дальнейшей работы.

На рисунке 6 визуализирована модификационная сетка точек grid. Сравнение рисунков 4 и 6 показывает разницу в количестве точек вне границ страны, участвующих в сетке. Важно отметить, что в случае с $country = "Estonia"$ для оптимальной работы выбран параметр для $RegionResize$ $l \times 0.5$ и $rad = 0.35$ для генерации $grid$

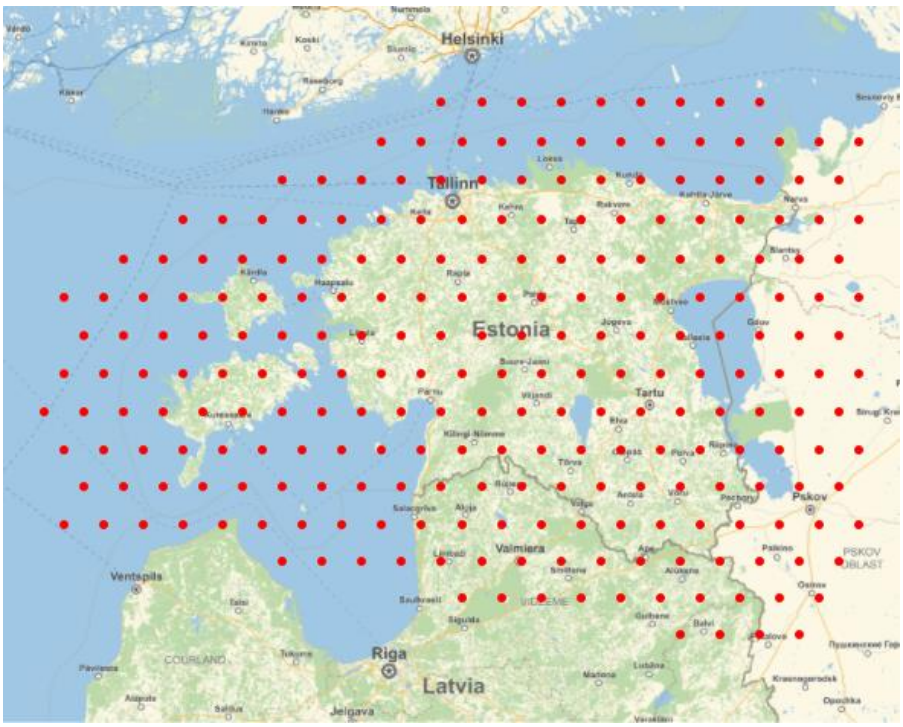


Рисунок 6 – Сетка $grid$ с $rad = 0.35$ и параметром $l \times 0.5$

3. ФОРМИРОВАНИЕ БАЗЫ ДАННЫХ

База данных формируется следующим образом: ранее был получен список стран для выбранной страны в пункте 2. Система Wolfram Knowledge [1] позволяет получить все доступные данные с каждой из станций. Однако существует проблема, что информации о некоторых станциях в конкретный временной промежуток вовсе не существует, поэтому полученная информация со станций подвергается дополнительной обработке, чтобы исключить станции, которые не несут какой-либо информации и лишь нарушают структуру базы данных.

Принцип работы метеостанций: станции фиксируют определенную информацию несколько раз в течение какого-либо промежутка времени и подаются в общую базу данных для этой станции. И если существуют данные, которые были зафиксированы в выбранный временной промежуток, то это те данные, который пойдут в базу данных для дальнейшей работы

Информация, которую пользователь получает для каждой станции представляет собой список списков, где каждый элемент – пара чисел: 1) количество секунд от начала эпохи, 2) метеоданные (в нашем случае температура воздуха в Цельсиях), полученная в конкретную секунду станцией.

Таким образом, по примеру, описанному выше, формируется общая база данных, где каждый элемент – это пары чисел, полученные для каждой станции в выбранный временной промежуток.

4. РАБОТА С БАЗОЙ ДАННЫХ

С полученной базой данных в пункте 4 очень затруднительно работать, поскольку данные не разделены на какие-либо временные промежутки, например, на дни, месяцы или года, и также системе не удается получить полную информацию, поэтому в некоторых парах в базе данных отсутствует корректная информация в какой-то момент времени. Поэтому для дальнейшей работы необходимо ввести несколько функций

4.1. Функция конвертации данных

Данная функция необходима для преобразования количества секунд с начала эпохи в привычный для пользователя вид, а также группирование элементов базы по выбранному пользователем типу, записанному в переменную k. Типами могут быть дни, месяца, года и если необходимо, секунды, часы, минуты.

Функция конвертации данных принимает на вход базу данных и k. Пусть база данных, полученная на вход, хранится в переменной $base$. Первым шагом, происходит преобразование каждой пары чисел к виду $\{\text{дата}, \text{данные}\}$. Дата представляет собой список:

$$\{\text{год}, \text{месяц}, \text{день}, \text{часы}, \text{минуты}, \text{секунды}\}.$$

Вторым шагом, для каждой станции в $base$ все существующие пары разделяются на списки, которым соответствуют правилу: у первого элемента каждой пары выбирается элемент, который определяется с помощью k и находятся все пары, где есть такой же равный элемент. На данном шаге информация по всем станциям в $base$ становится упорядоченной, однако для дальнейшей работы удобнее преобразовать каждую полученную группу пар в общую пару со средним по всем вторым элементам в каждой паре значением во втором элементе и первым элементом в виде первого элемента первой, участвующей в группе, пары.

Как говорилось ранее, существуют случаи, когда данных за конкретный день у какой-либо станции не было найдено, поэтому для дальнейшей работы с

базой, вводится дополнительное ее преобразование: если количество пар для каждой станции оказалось меньше, чем длина выбранного уже сгруппированного временного промежутках, то для данной станции количество пар добавляется до длины, отобранной ранее. Существует два способа выбирать данные для дополнения: любой существующей случайной парой или парой, второй элемент которого – среднее значение вторых элементов уже существующих до данного преобразования пар.

4.2. Функция нахождения данных для любой точки сетки по информации, полученной от ближайшей метеостанции

Для реализации дальнейших подзадач потребуется функция, способная сопоставлять любой точке сетки данные, полученные с ближайшей метеостанции. Пусть такая функция храниться в переменной $nearestStation$.

Данная функция принимает на вход базу данных и точку – пару чисел. Первым шагом, для заданной точки находится станция, координаты которой максимально приближены к координатам точки. Следующим шагом, рассматривается та часть базы данных, которая связана с найденной станцией: выбирается первая пара, хранящая в себе дату и метеоданные на момент данного времени. Если же метеоданные были не найдены и вторым элементом данной пары является не число, то создается копия базы и формируется новая база, в которой целый блок, связанный с выбранной станцией больше не рассматривается при поиске ближайшей к точке станции. И описанные выше шаги повторяются, пока в качестве метеоданных не окажутся корректные данные.

5. ВИЗУАЛИЗАЦИЯ

Построение температурной карты представляет собой «наложение» заранее подготовленной найденной ранее оболочки на участок карты, в котором располагается выбранная страна

5.1. Окрашивание оболочки

Для дальнейшего использования уже существующей оболочки необходимо в соответствии с информацией из базы данных каждому уникальному значению в каждой точке сетки дать определенный цвет. И в соответствии со значениями во всех точках из $grid$ окрасить оболочку.

Пусть в переменной $temperature$ будет хранится нужная для окрашивания оболочки информация. $temperature$ представляет собой список списков, где каждый список состоит из трёх элементов: координат точки сетки по обоим осям и результат применения функции $nearestStation$.

Также необходимо получить дополнительную информацию о размерах шаблона цвета, накладываемого на оболочку. Такая информация будет хранится в переменной $range$. Значение данной переменной – список списков, где каждый список – минимальная и максимальная координата по каждой оси координат.

При помощи функции $ListDensityPlot$ создается уже окрашенная оболочка. Главные аргументы, поступающие вход – $temperature$, хранящая в себе информацию о точках сетки, значение параметра для опции $PlotRange$ – информация, хранимая в $range$. Она необходима для дальнейшего корректного наложении оболочки на карту страны для того, чтобы размеры оболочки совпали или были немного больше территории страны, чтобы полностью ее покрыть. Еще одна важная опция $ListDensityPlot$ является MaxPlotPoints, значение параметра которой напрямую зависит от качества смешения цветов оболочки. Чем больше значения параметра, тем плавне и сглаженнее выглядит наложенный цвет.

Пусть результат применения функции $ListDensityPlot$ хранится в переменной $plot$

5.2. Выбор цвета

В пункте 6.1 было описано, как покрыть оболочку каким-либо цветом. Стоит уточнить, как и какой набор цветов можно использовать для окрашивания оболочки. Функция $ListDensityPlot$ обладает опцией $ColorFunction$. Выбор значения ее параметра напрямую будет влиять на конечный вид окрашенной оболочки. В системе Wolfram Knowledge существует множество готовых цветовых гамм, например, «TemperatureMap», «NeonColors», «Rainbow». Однако для более привычного пользователю виду необходимо создать собственную цветовую гамму, но на дальнейшую работу выбор цвета никак не повлияет.

Пусть в переменной $scheme$ хранится цветовая гамма, которая используется при окрашивании оболочки в переменной $plot$. В дальнейшем цветовая гамма по умолчанию будет собственной, а не заготовленной системой. На рисунке 7 изображена созданная заготовленная цветовая гамма.

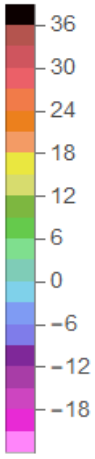


Рисунок 7 – Созданная цветовая гамма

На рисунках 8 и 9 представлена одна и та же оболочка, на которую наложены разные цветовые гаммы.

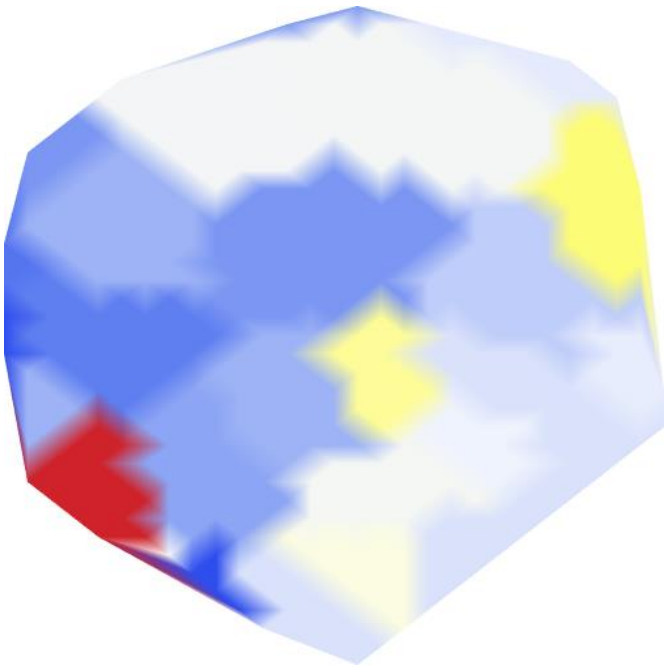


Рисунок 8 – Оболочка с параметром $ColorFunction$ → «TemperatureMap»

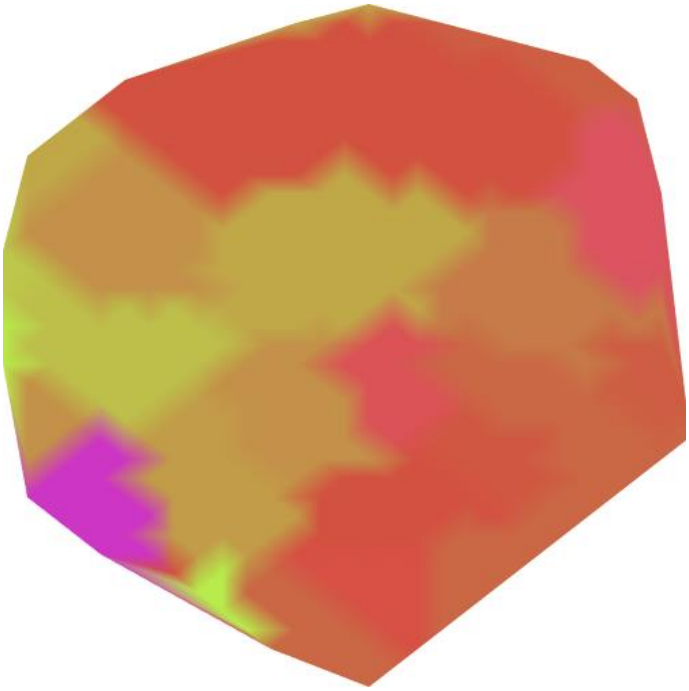


Рисунок 9 – Оболочка с параметром $ColorFunction$ → «NeonColors»

На рисунках 10,11 изображена наглядная разница между отсутствием опции $MaxPlotPoints$ и с ее значением параметра, равному 25

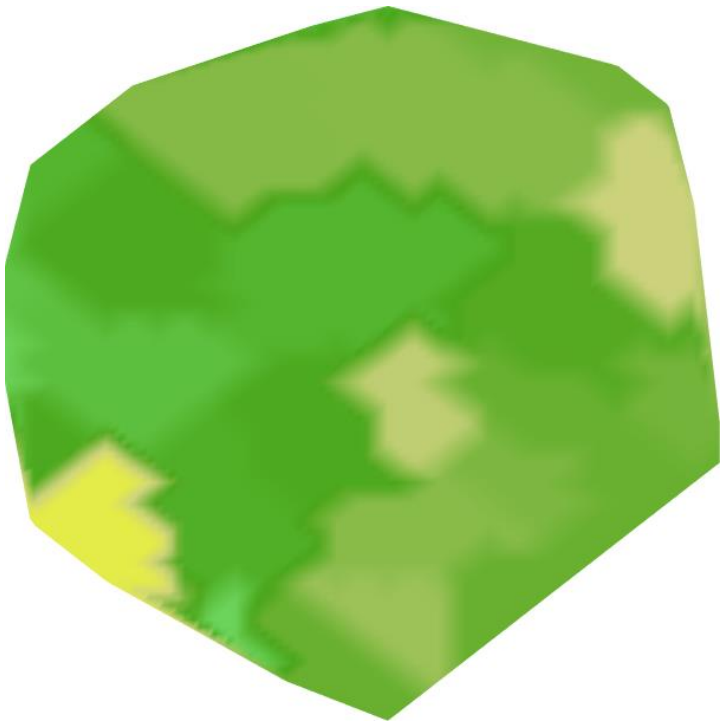


Рисунок 10 – Отсутствие опции $MaxPlotPoints$

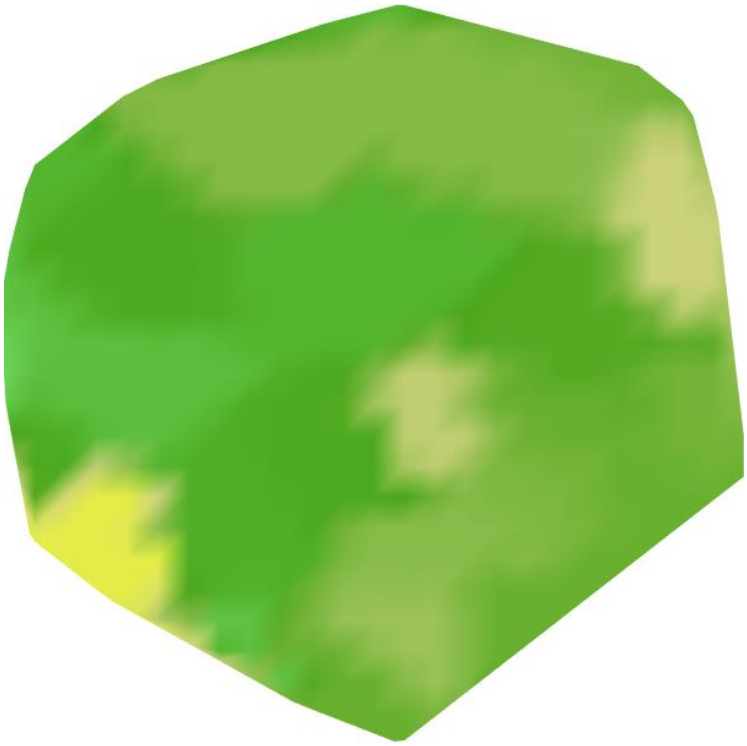


Рисунок 11 – Значение параметра опции $MaxPlotPoints$ равна 25

5.3. Построение карты

Финальным шагом визуализации является построение температурной карты. Основная функция для работы является $GeoGraphics$, которая строит

часть карты по выбранной стране и накладывает на всю территорию страны полученную в пункте 6.2 оболочку.

После предыдущего шага для удобства пользования следует добавить шкалу зависимости температуры по Цельсию и цветов. Например, как показано на рисунке 7

Один из вариантов температурной карты для страны Германия изображен на рисунке 12

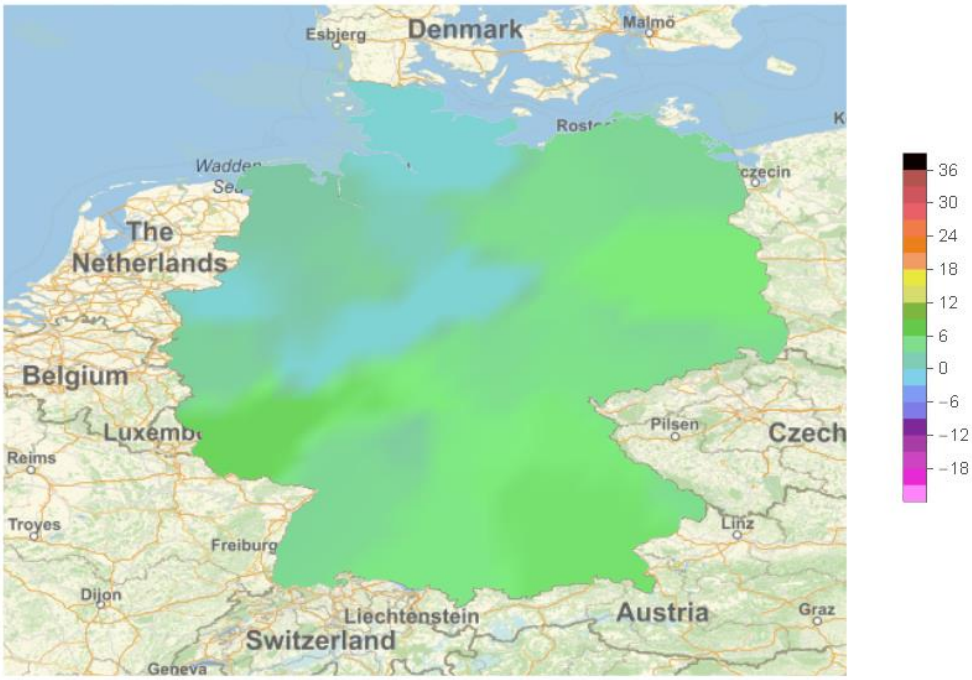


Рисунок 12 – Температурная карта Германии

6. СПОСОБЫ ПРЕДСТАВЛЕНИЯ

В предыдущих пунктах было описано, что функция $NearestStation$ сопоставляет точкам данные за один промежуток времени, на которые делилась информация в базе. Соответственно, построение температурных карт для каждого из промежутка происходит по отдельности. Именно поэтому заключительной частью является представление этих карт в структурированном виде, чтобы было удобнее в дальнейшем пользователю работать с ними.

6.1. Создание карт для каждого «дня» заданного промежутка

Назовем временный промежуток, по которому данные преобразовывались в группы «днем». В разделе 6 были описаны методы создания температурных карт для произвольного «дня». Следующая задача будет заключаться в представлении в одном списке карт всех «дней». Для этого достаточно с помощью функции $Table$ и преобразованием копии базы данных строить карту для каждого «дня». Что касается реализации, она остается такой же. Разница лишь в том, что перед созданием новой карты, копия базы текущей базы преобразуется так, что текущий «день» в базе для каждой станции становится первым элементом, поскольку функция $NearestStation$ работает с первым элементом каждой из станций в базе. Пусть в переменной $maps$ хранится список всех построенных температурных карт. Именно с $maps$ будет происходить дальнейшая работа

6.2. Анимированное представление

Одним из способов представления полученных карт будет анимация. С помощью функции $ListAnimate$ задается количество секунд, по истечению которых один «слайд» переключается на следующий. Данный способ очень удобен, если требуется наглядно сравнить, как менялась погода в заданном промежутке.

6.3. Представление с помощью вкладок

Еще одним способом представления температурных карт являются вкладки. С помощью функции $TabView$ создается цельная структура, в которой располагаются вкладки по «дням» и основное окно, в котором отображается созданная карта для этого «дня». Такой метод очень удобен, когда не требуется наглядное представление, как в пункте 7.2, однако для работы с картами пользователю необходима общая картина и возможность сравнивать любые «дни» между собой, например. На рисунке 13 изображен пример работы с вкладками.

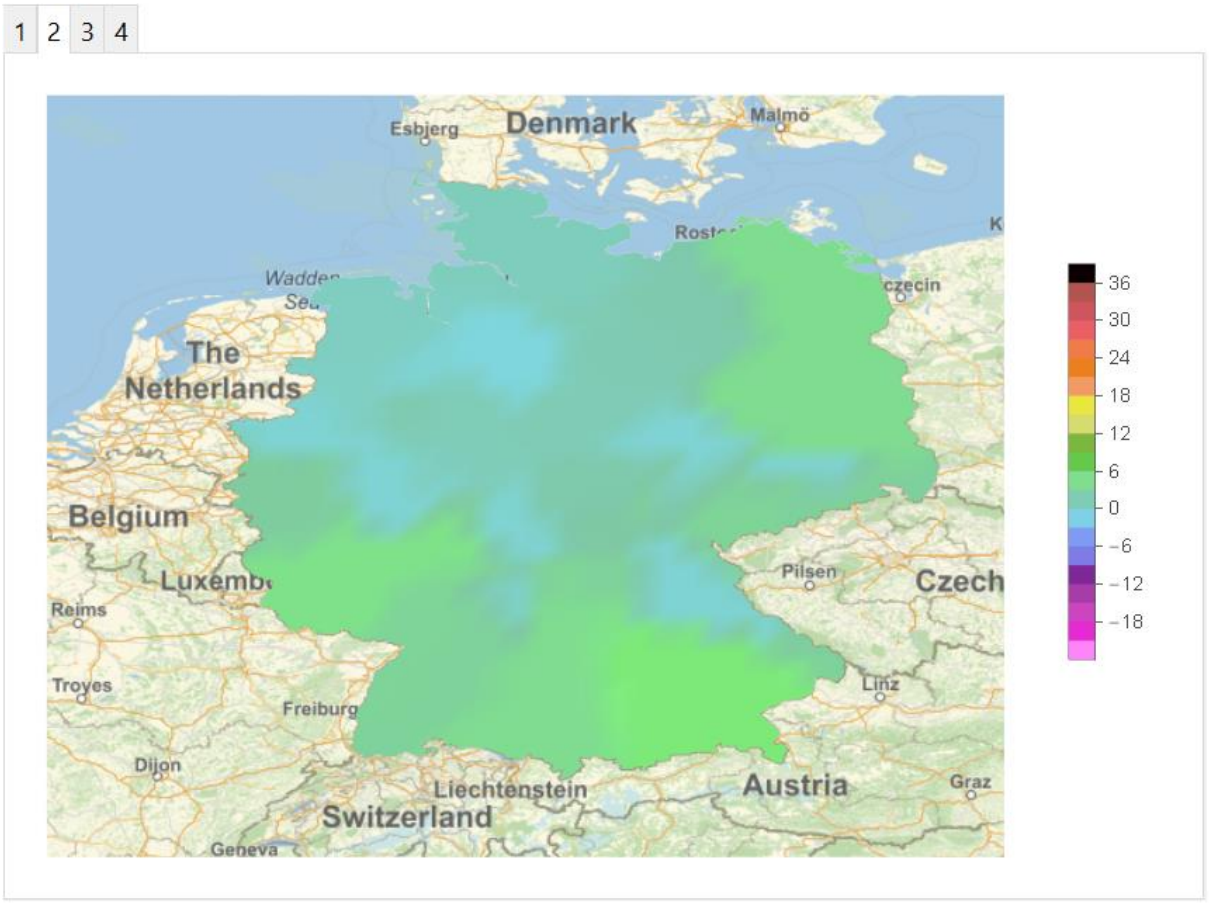


Рисунок 13 – Представление карт с помощью вкладок

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы был разработан и реализован алгоритм построения температурных карт стран мира. Были рассмотрены способы работы с Wolfram Knowledge, формирования на основе ее работы баз данных, их обработки.

Результатом проделанной работы является набор функций, которые способны создавать и представлять в различных формах температурные карты выбранной пользователем страны.

При желании в дальнейшем рассмотрении можно выбирать другие метеоданные, например, скорость ветра, влажность, давление, помимо уже рассмотренной температуры воздуха.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Wolfram Documentation [Электронный ресурс] / Documentation center. URL: https://reference.wolfram.com/language/ (дата обращения – 17.05.21);
2. Алгоритмы: построение и анализ, 3-е изд. / Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн: Пер. с англ. – М. : ООО «И. Д. Вильямс», 2013. – 1328 с.