Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**Метеорологическая карта**

Студент

Гр.053501

Савчук М.С.

Руководитель

ассистент кафедры информатики

Удовин И.А.

Минск 2021

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой Информатики

––––––––––––––––––––––––

(подпись)

Волорова Н.А., 2021 г.

ЗАДАНИЕ

по курсовому проекту

Студенту   *Савчуку Максиму Сергеевичу*

1. Тема работы «Метеорологическая карта»

2. Срок сдачи студентом законченной работы «\_\_».\_\_\_\_\_\_\_.2021 г.

3. Исходные данные к работе

3.1. Операционная система Windows.

3.2. Язык программирования C++. Фреймворк Qt.

4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке).

4.1. Введение.

4.2. Постановка задачи.

4.3. Структура проекта.

4.4. Реализация проекта.

4.5. Общий вид карты.

4.6. Тестирование

4.7. Заключение.

4.8. Список используемой литературы.

5. Консультант по курсовой работе Удовин И.А.

6. Дата выдачи задания «\_\_» \_\_\_\_\_\_\_\_\_ 2021г.

7. Календарный график работы над проектом на весь период

проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

7.1. разделы 1-3 к 28.02.2021г. – 10 % готовности работы;

7.2. раздел 4 к 15.03.2021г. – 30 % готовности работы;

7.2. раздел 5 к 15.04.2021г. – 60 % готовности работы;

7.2. разделы 6-7 к 10.05.2021г. – 80 % готовности работы;

Заключение, Приложения к 20.05.2021г. – 90 % готовности работы;

Оформление пояснительной записки и графического материала к 24.05.2021г. – 100 % готовности работы.

Руководитель курсовой работы *Удовин И. А.*

Задание принял к исполнению *Савчук М.С.* (дата и подпись студента)

**ОГЛАВЛЕНИЕ**

**ВВЕДЕНИЕ**............................................................................................................5 **1.** **АНАЛИЗ ПРЕДМЕТНОЙ** **ОБЛАСТИ**.......................................................7

**2.** **РАЗРАБОТКА** **ПРОГРАММНОГО** **СРЕДСТВА**....................................12

**3.** **ТЕСТИРОВАНИЕ**.........................................................................................21 **ЗАКЛЮЧЕНИЕ**...................................................................................................23 **СПИСОК** **ИСТОЧНИКОВ**................................................................................24

**ВВЕДЕНИЕ**

В любой современной стране погода и климат давно перестали быть чисто метеорологическими категориями. Они все более становятся терминами экономического, социального и иных профилей, т.к. от них зависит множество самых разнообразных видов человеческой деятельности.

Погода на сегодняшний день без преувеличения стала важнейшим фактором экономической безопасности территории любой страны или региона. Также не требуют особых доказательств все преимущества от гармоничного сосуществования общества и окружающей природной среды.

Природа двояко влияет на хозяйственную деятельность человека. С одной стороны, стихийные экологические и гидрометеорологические бедствия наносят значительный урон экономике. Однако с другой - достоверная информация о погоде и климате формирует целый комплекс дополнительных ресурсов экономической эффективности. Например, это значит, что предприятия могут минимизировать потери и ущербы от неблагоприятных погодных условий и существенно увеличить выгоду за счет рационального использования благоприятных факторов природной среды. Особенно заметный экономический эффект дает использование метеорологической информации в авиации, энергетике, строительстве, рыболовстве и судоходстве, сельском хозяйстве и ряде других отраслей и направлений человеческой деятельности.

Одним из направлений в решении этой задачи является создание погодных «информеров». Использование функционального онлайн-сервиса прогноза погоды на сегодня является самым простым и широко востребованным способом узнать текущую погоду в определенном месте или ознакомиться с прогнозом на предстоящие дни.

Данная программа представляет собой специализированное приложение, с помощью которого можно получать информацию о погодных явлениях во всех точках мира. Данные представляются в виде цветных погодных карт, окраска которых соответствует значениям погоды в данной точке.

Текущие и прогнозные погодные карты отображают:

* влажность;
* атмосферное давление;
* температура.

Интерфейс также оснащен стандартными значками обозначения погодных явлений, знакомыми каждому человеку.

**1.** **АНАЛИЗ** **ПРЕДМЕТНОЙ** **ОБЛАСТИ**

**1.1** **Постановка задачи**

Задачей курсовой работы является написание программы, которая наглядно предоставляет информацию о погоде в мире в виде «карты погоды». Программа должна строить:

* Карты температуры
* Карты давления
* Карты влажности

На ближайшую неделю с помощью билинейной интерполяции. Информация запрашивается со стороннего метеорологического сайта с помощью ключа API.

**1.2** **Что такое API?**

API (application programming interface) — описание способов (набор [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [процедур](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [функций](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [структур](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) или [констант](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D0%B0%D0%BD%D1%82%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))), которыми одна компьютерная программа может взаимодействовать с другой программой. Обычно входит в описание какого-либо [интернет-протокола](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) (например, [RFC](https://ru.wikipedia.org/wiki/RFC)), программного [каркаса (фреймворка)](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) или стандарта вызовов функций [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0). Часто реализуется отдельной [программной библиотекой](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0) или сервисом [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0). Используется программистами при написании всевозможных [приложений](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5).

Если программу (модуль, библиотеку) рассматривать как [чёрный ящик](https://ru.wikipedia.org/wiki/%D0%A7%D1%91%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D1%89%D0%B8%D0%BA), то API — это множество «ручек», которые доступны пользователю данного ящика и которые он может вертеть и дёргать.

Программные компоненты взаимодействуют друг с другом посредством API. При этом обычно компоненты образуют иерархию — высокоуровневые компоненты используют API низкоуровневых компонентов, а те, в свою очередь, используют API ещё более низкоуровневых компонентов.

По такому принципу построены [протоколы передачи данных](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) по [Интернету](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). Стандартный стек протоколов ([сетевая модель OSI](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI)) содержит 7 уровней (от физического уровня передачи бит до уровня протоколов приложений, подобных протоколам [HTTP](https://ru.wikipedia.org/wiki/HTTP) и [IMAP](https://ru.wikipedia.org/wiki/IMAP)). Каждый уровень пользуется функциональностью предыдущего («нижележащего») уровня передачи данных и, в свою очередь, предоставляет нужную функциональность следующему («вышележащему») уровню.

Понятие протокола близко по смыслу к понятию API. И то, и другое является абстракцией функциональности, только в первом случае речь идёт о передаче данных, а во втором — о взаимодействии приложений.

API библиотеки функций и классов включает в себя описание сигнатур и семантики функций.

Практически все [операционные системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) ([UNIX](https://ru.wikipedia.org/wiki/UNIX), [Windows](https://ru.wikipedia.org/wiki/Windows" \o "Windows), [OS X](https://ru.wikipedia.org/wiki/OS_X), [Linux](https://ru.wikipedia.org/wiki/Linux" \o "Linux) и т. д.) имеют API, с помощью которого программисты могут создавать приложения для этой операционной системы. Главный API операционных систем — это множество [системных вызовов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2).

Однако, для данного проекта используется только API key - это имя, данное некоторой форме секретного токена, который отправляется вместе с запросами веб-службы (или аналогичными), чтобы идентифицировать источник запроса. Часто ключ API действует как форма аутентификации, которая может привести к контролю доступа. Например, вы можете ограничить доступ к определенным действиям API в зависимости от того, кто выполняет запрос.

**1.3** **Выбор и описание стороннего сервиса для получения данных.**

В данном проекте используется сайт[1], который предоставляет множество видом информации:

* Текущие данные о погоде
* Почасовой прогноз на 4 дня
* Ежедневный прогноз на 1 – 30 дней
* Погода в предыдущие дни

Эту информацию для определенного местоположения можно получить многими способами:

* По названию города
* По ID города
* По географическим координатам
* По почтовому индексу

Для нескольких городов можно получить информацию

* Города в прямоугольной зоне
* Города в кругу

Зарегистрировав аккаунт на данном сервисе, предоставляется возможность отправить 10,000,000 запросов в месяц, при этом не более 3,000 запросов в день.

Другие сервисы являются менее удобными в использовании, а также требуют оплаты тарифного плана, на время использования.

**1.4 Описание принципа создания карты с помощью билинейной интерполяции и сравнение с другими принципами**

Для рисования карты в нужном цветовом спектре, необходимо знать значения температуры/давления/влажности в каждой точке мира. Запрос значений погоды в каждой точке мира требует большого количества времени и попыток запросов. Встает вопрос о минимизации количества запросов и получения погоды в каждой точке мира из имеющихся данных.

Одним из способов получения значений является BFS (поиск в ширину), где в очередь добавляются какие-то вершины, от которых начинается отрисовка карты (цветом, соответствующим значению погоды в вершине). В дальнейшем, в очередь добавляются ближайшие точки к вершине. Когда алгоритм встречает уже закрашенную точку, то пропускает её. Однако, у этого метода есть огромные минусы:

* Проблема выбора начальных вершин
* Резкий контраст соседних областей

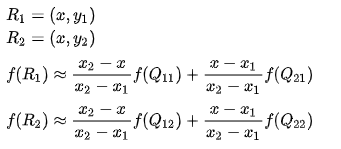
Схожие проблемы и при рисовании карты диаграммами Вороного.

Другой способ подразумевает получение значений на карте с помощью билинейной интерполяции. Билинейная интерполяция — в [вычислительной математике](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) — обобщение [линейной интерполяции](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D0%BE%D0%BB%D1%8F%D1%86%D0%B8%D1%8F) одной [переменной](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D0%B0) для функций двух переменных. Обобщение основано на применении обычной линейной интерполяции сначала в направлении одной из координат, а затем в перпендикулярном направлении.

Итак, для построения карты достаточно знать значения погоды в углах её сетки. Затем для получения значения температуры/давления/влажности в случайной точке карты, необходимо найти 4 ближайшие к ней узла сетки.

Пусть нам нужно интерполировать значение функции f(x,y) в точке P = (x,y). Значения функции в окружающих точку P точках  Q11=(x1,y1), Q12=(x1,y2), Q21=(x2,y1) и Q22=(x2,y2) известны.

Первым шагом линейно интерполируется значение вспомогательных точек  R1 и R2 вдоль [оси абсцисс](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%86%D0%B8%D1%81%D1%81%D0%B0), где



Теперь проводится линейная интерполяция между вспомогательными точками R1 и R2.



Это и есть интерполируемое значение функции f(x,y). Здесь, подбирая под своё значение точки цвет, переходы будут создавать красивый градиент.

**1.5 Получение карты мира**

Даже при получении готового градиента температуры/давления/ влажности, встает вопрос о нанесении на карту границ мира. Для решения этой проблемы используется сайт [2], на котором содержаться файлы разрешения .json, содержащие точки границ выбранных стран, объединений стран, материков и т.д. Полученный файл содержит точки границ карты всего мира.

**1.6 Выбор** **инструментов** **разработки**

**C++** — компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.

**Qt** — фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++.

**2.** **РАЗРАБОТКА** **ПРОГРАММНОГО** **СРЕДСТВА**

**2.1 Настройка проекта**

Файл 8.pro содержит всю необходимую информацию для того, чтобы qmake собрал приложение. Для работы с получением и отправкой запросов прописана следующая строка:

QT += network

Для отображения иконки приложения:

win32:RC\_ICONS += icon.ico

Прописан define для удобного обращения к директории с исходным кодом:

DEFINES += PRJ\_PATH='\\"$${PWD}\\"'

**2.2 Описание класса menu**

Все вычисления происходят в классе menu. Здесь содержаться методы, которые получают информацию погоды в узлах сетки, интерполируют значения для всей карты, накладывают поверх карту мира и полосу градиента.

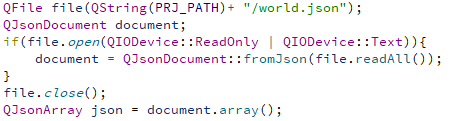


В классе содержится единственная кнопка, по нажатию на которую выполнятся следующие действия по порядку:



Сначала создаются новые экземпляры структур данных, которые используются внутри класса. Затем вызываются следующие методы:

1) Метод get\_coords() парсит координаты границ каждой страны из файла world.json при помощи удобного функционала QT.



В файле содержится js массив, который мы извлекаем из документа. Цикл проходит по каждой стране и, в зависимости от вложенности, сохраняет все точки границы в полигоны. Предварительно координаты преобразовываются в допустимые и удобные для монитора по формулам:

X = lon \* 4 + 680,

Y = lat \* (-4) + 400.

Существует проблема вложенностей, так как некоторые страны состоят из одного замкнутого участка земли, а некоторые имеют не прилегающие территории или острова (из-за этого некоторые страны имеют на один вложенный массив больше). Для хранения границ одной страны используется QList<QPolygon>, где QPolygon – один замкнутый участок земли.

Итак, формируется QVector<QList<QPolygon>>, который содержит границы всех стран мира.

2) Метод get\_icons() получает с сайта [2] стандартные иконки погоды, поддерживающиеся на их сервисе.



Все иконки преобразуются в класс QImage и сжимаются до размера 30x30 пикселей. Затем иконки сохраняются в QMap<QString, QImage>, где для кодовых строчных названий иконок соответствует изображение.

3) Метод get\_weather() получает прогнозируемые погодные данные в узлах сетки карты. Для работы с запросами используются библиотеки QNetworkAccessManager, QNetworkRequest, QNetworkReply.

Создается экземпляр QNetworkAccessManager, куда параметром передается объект menu. Дальше двойной цикл проходит по всем кратным 10 координатам долготы и широты, запрашивая в каждом узле прогноз на 7 дней. Ссылка формируется из базовой ссылки + необходимых координат + ключа API, авторизующего запрос.

Затем цикл проходит по полученному файлу .json и выделяет погоду отдельно для всех дней. Каждый день содержит информацию о температуре днем, давлении, влажности, кодовом названии иконки погоды и многом другом, что не используется в проекте.



Итак, формируется QMap<QPair<int, int>, QVector<QPair<QList<double> ,QImage>>>, который для индексации использует QPair из x и y координаты узла (преобразованных, аналогично методу get\_coords()), и хранит QVector из пары листа параметров и изображения иконки. QVector будет содержать 7 элементов (на 7 дней), QList в свою очередь 3 элемента (температуру, давление и влажность).

Удаляются созданные экземпляры QNetworkAccessManager, QNetworkReply.

Вид полученного прогноза на 1 день файлом .json имеет вид:

{"dt":1621900800,

"sunrise":0,

"sunset":0,

"temp":{

"day":221.33,

"min":217.39,

"max":221.9,

"night":220.94,

"eve":221.17,

"morn":218.5},

"feels\_like":{

"day":203.91,

"night":203.24,

"eve":204.42,

"morn":202.18},

"pressure":1041,

"humidity":94,

"weather":[{"id":800,

"main":"Clear",

"description":"sky is clear",

"icon":"01d"}],

"speed":4.98,

"deg":248,

"gust":4.62,

"clouds":6,

"pop":0}

4) Метод get\_pictures() создает по 4 изображения на каждый из 7 дней.

Изначально создается 3 пустых изображения, размером с разрешение экрана и 3 инструмента QPainter для рисования на изображениях. Затем для каждого пикселя изображения, находятся ближайшие 4 вершины, вычисляется примерное значение температуры в данной точке, вызывается метод

long double **map**(long double x, long double in\_min, long double in\_max, long double out\_min, long double out\_max) {

return (x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + out\_min;

}

Этот метод переводит значение из старого диапазона (in\_min - in\_max) в новый удобный (out\_min - out\_max). Так получается значения параметров погоды в пределах от 0 до 255. Далее каждый пиксель окрашивается в подходящий цвет RGB. Для разных типов карт – разные градиенты выходят.

Поверх трёх карт накладываются границы стран, методом draw\_boundaries(QPainter& painter). Этот метод проходит, по ранее созданному вектору из списков полигонов, рисуя их на изображении.

Затем создается копия QImage температурной карты, на которую поверх, с помощью QPainter, вставляются в узлы сетки иконки соответствующей там погоды.

5) На карты рисуются палитры, с помощью методов draw\_palitra\_temp(QPainter& painter), draw\_palitra\_press(QPainter& painter), draw\_palitra\_humid(QPainter& painter). Первый используется для карты температуры и карты температуры с иконками.

Второй и третий соответственно для давления и влажности.

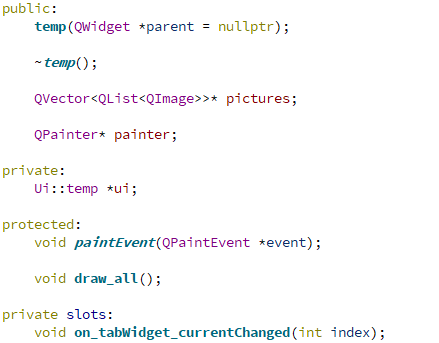
Все палитры рисуются с помощью 255 вертикальных однопиксельных полос, идущих друг за другом. Цвета полосам, аналогично пикселям после интерполирования, задаются соответствующим набором RGB.

Все конечные изображения помещаются в вектор из списка картинок.

6) Создается экземпляр окна класса temp, которому передается указатель на QVector<QList<QImage>> pictures. К окну применяются флаги, которые заставят его открыться на полный экран. Происходит вызов нового окна и закрытие старого.

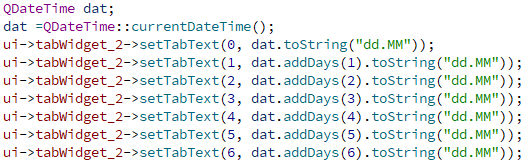
**2.3** **Описание класса temp**

В классе temp происходит только готовых изображений на экран.

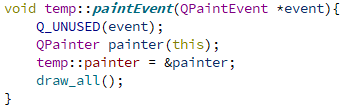


Окно класса temp имеет два контейнера QTabWidget. На верхнем контейнере отображены режимы карты (Temperature, Pressure, Humidity). На нижнем – даты ближайших 7-ми дней. Только во время выбранного режима температурной карты появляется элемент QCheckBox, при нажатии на который, появляется температурная карта с иконками погоды в узлах сетки.

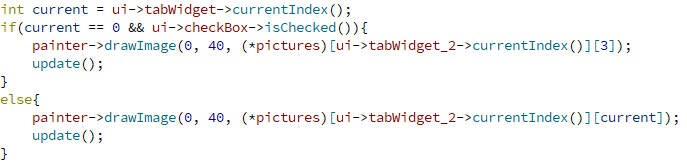
Конструктор окна переименовывает названия вкладок нижнего контейнера в ближайшие даты, используя форму класса QDateTime:



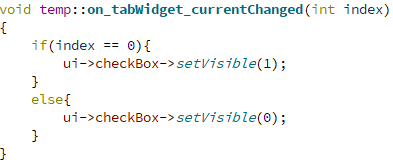
При открытие окна, а так же при любом его изменении, вызывается метод paintEvent(QPaintEvent \*event). Он создает экземпляр QPainter, передавая параметром ссылку на объект окна. Вызывается метод draw\_all().



Этот метод проверяет текущие выбранные режимы QTabWidget и рисует необходимые рисунки на фоне приложения.



Так же реализован метод, который вызывается при смене режима QTabWidget, он проверяет, должен ли быть виден QCheckBox (выводит его, если необходимо).



**2.5 Как выглядит непосредственное взаимодействие с программой**

При запуске программы, появляется такое окно, на котором находиться единственная кнопка. На фоне рисуется QImage, также расположенная в папке с кодом. При нажатии на кнопку программа начинает считывать данные и строить карты. Процесс может быть отслежен с помощью элемента QProgressBar.



Рисунок 1.

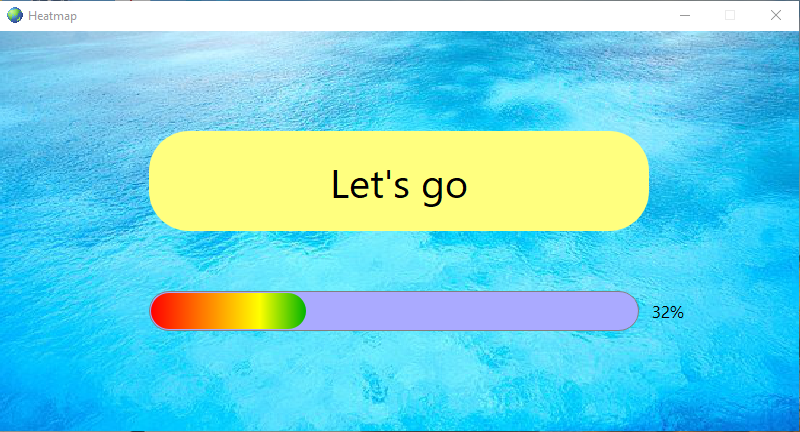


Рисунок 2.

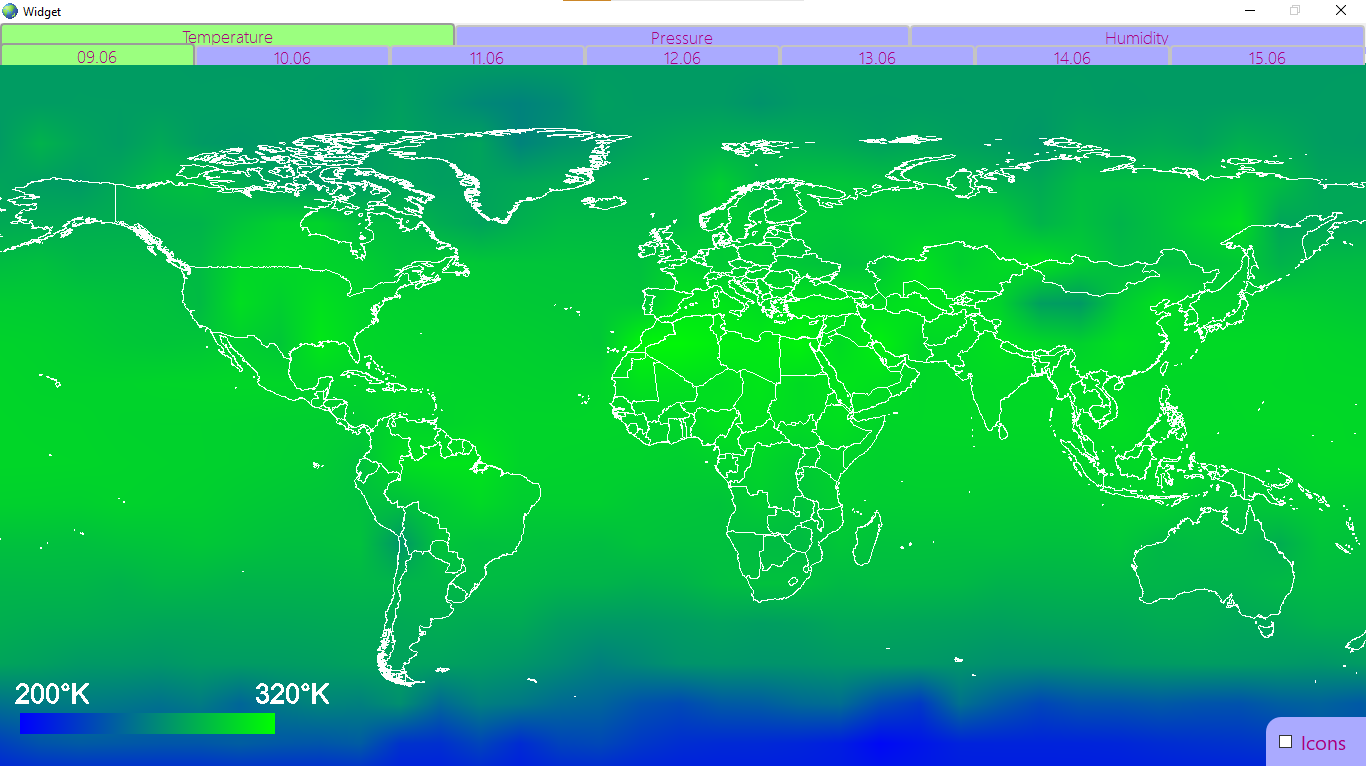


Рисунок 3.

**3.** **ТЕСТИРОВАНИЕ**

**3.1.** **Тест** **1**

Попытка отключения компьютер от сети, а затем переподключение.

Ожидаемый результат: приостановление получения данных

Фактический результат: остановка движения QProgressBar после отключения сети, и возобновление при подключении.



Рисунок 4.



Рисунок 5.

**3.2.** **Тест** **2**

Повторное нажатие на кнопку для начала загрузки данных

Ожидаемый результат: продолжение загрузки

Фактический результат: загрузочная полоса продолжила двигаться, завершая первое нажатие



Рисунок 6.



Рисунок 7.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы:

- Был выполнен анализ предметной области, рассмотрены основные функции, необходимые для успешного написания программы.

- Было рассмотрен принцип и пример работы с key API.

- Был рассмотрен алгоритм билинейной интерполяции и его сравнение с другими алгоритмами построения градиента карты.

- Были закреплены навыки работы C++ и Qt.

- Были исследованы основные принципы работы со средой Qt Creator, изучена работа с основными его компонентами.

- Была проведена работа по парсингу файлов типа json в понятные для С++ данные.

- Было разработано специализированное приложение, с помощью которого можно получать информацию о некоторых состояниях погоды во всех точках мира.

В дальнейшем этот проект может быть доработан. Программа имеет много путей для развития, хоть она уже может использоваться в узкой специализации.

**СПИСОК** **ИСТОЧНИКОВ**

1)<https://openweathermap.org/> [1]

2)https://public.opendatasoft.com/explore/dataset/world-administrative-boundaries/export/ [2]

3)<https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%BB%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D0%BE%D0%BB%D1%8F%D1%86%D0%B8%D1%8F>

4)<https://ru.wikibooks.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2/%D0%91%D0%B8%D0%BB%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D0%BE%D0%BB%D1%8F%D1%86%D0%B8%D1%8F>

5) <https://doc.qt.io/qt.html>