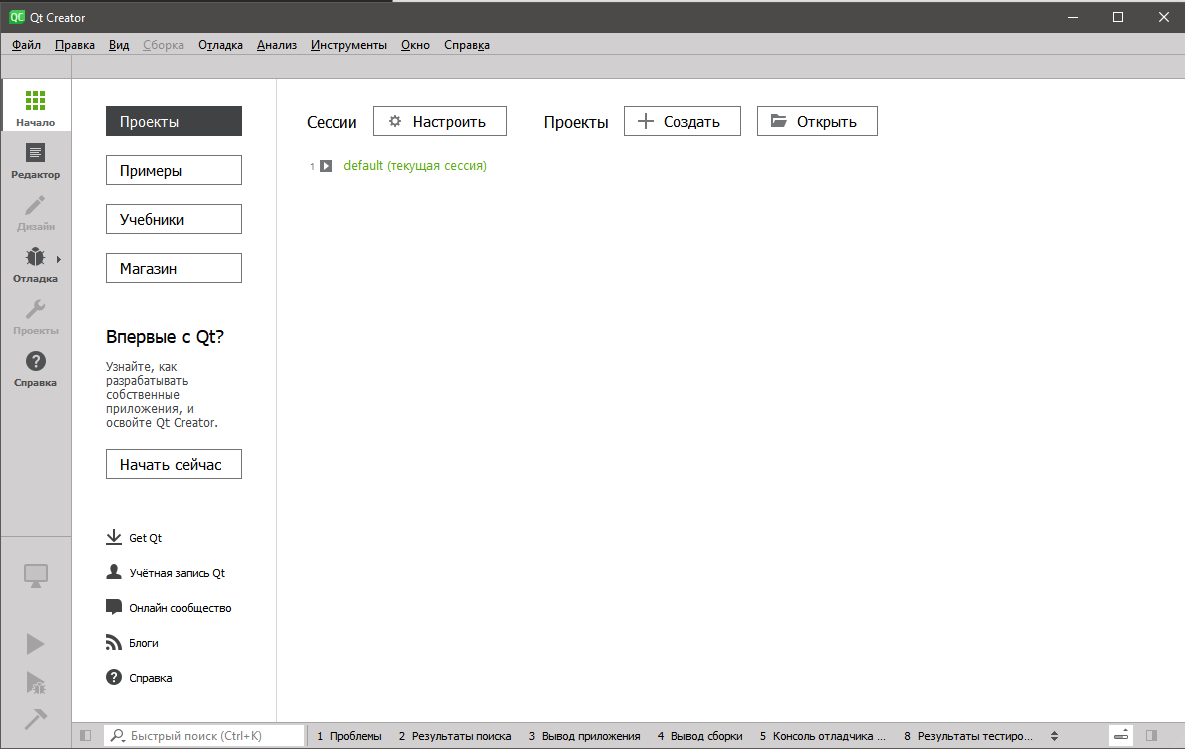
Вводный урок

1. Редактор кода

Qt Creator после первого запуска будет выглядеть следующим образом:



Страница, на которой оказались, – домашняя страница приложения, здесь будут показаны недавние проекты, которые уже редактировали. Также в этой вкладке можно создать или открыть уже существующий проект.

В левой части на сером фоне находятся кнопки “Редактор”, “Дизайн”, “Отладка”, “Параметры” и “Справка”. Данная часть интерфейса отвечает за переключение между основными вкладками при работе с проектами. О первой вкладке было рассказано ранее, она открывается по умолчанию при запуске IDE. Следующая страница – текстовый редактор, который позволит изменять файлы форматов .cpp, .h и некоторых других. Вкладка “Дизайн” позволит редактировать формы (окна), которые будут использоваться в приложении. “Отладка” будет использована для работы с ошибками в приложении или проблемами на этапе компиляции. Следующая страница позволит работать с настройками сборки проекта и управлять проектами в целом.

Чуть ниже расположены инструменты для работы с конкретным проектом:

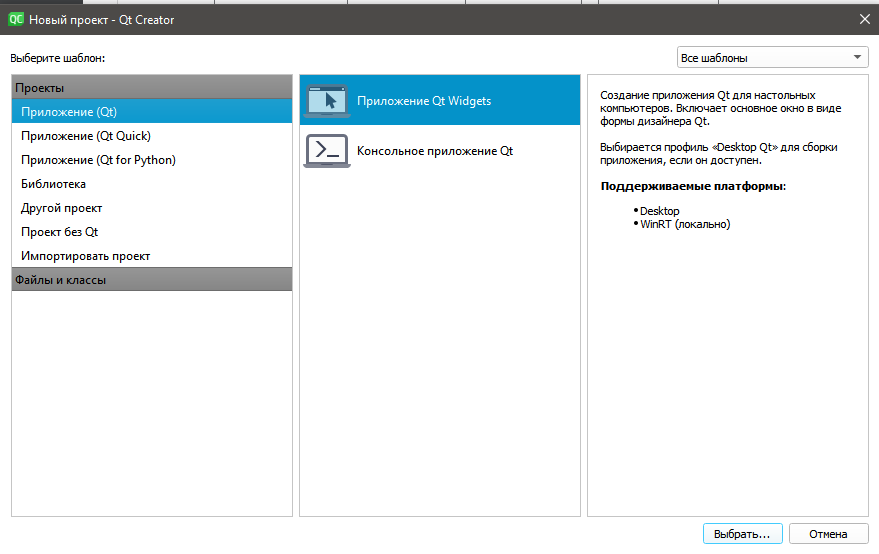
* “Экран” – выбор режима работы с проектом;
* “Треугольник” – запуск проекта в режиме отладки;
* “Треугольник с отладкой” – запустить отладчик проекта;
* “Молоток” – собрать проект.

В нижней части окна будут появляться консольные выводы разного содержания, которое будет зависеть от выбранной вкладки.

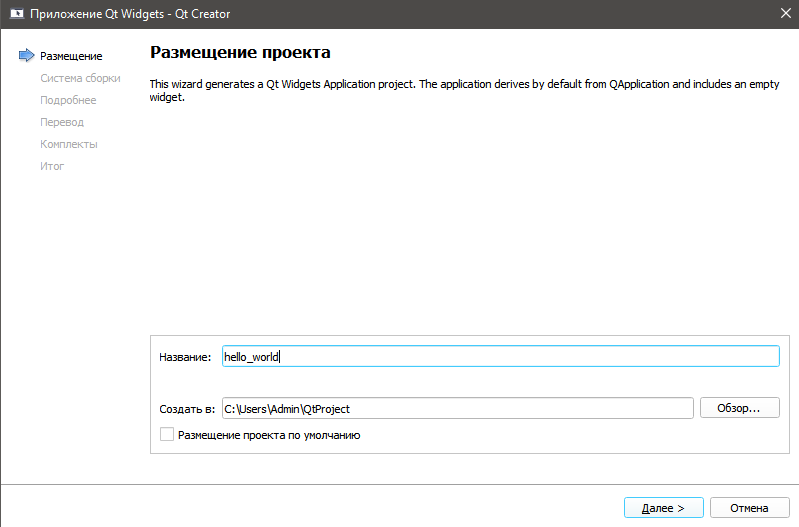
1. Создание проекта

Для того чтобы создать новый проект, нажмите на соответствующую кнопку на домашней странице.

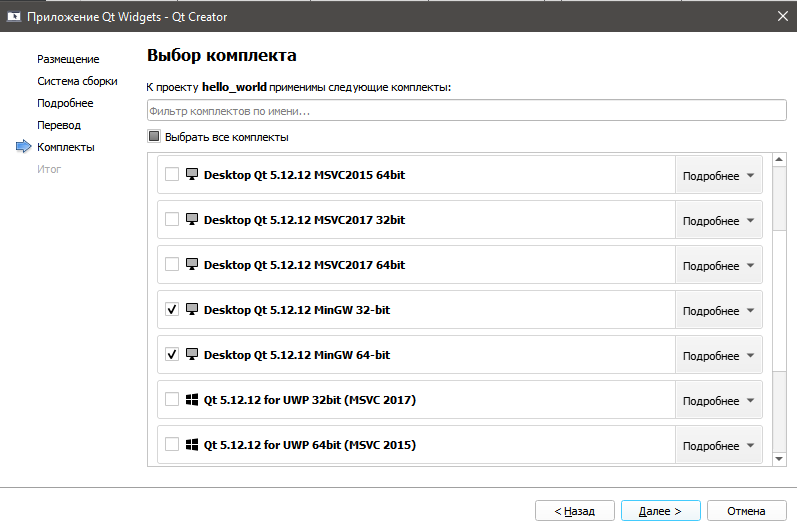
Далее выберите Приложение (Qt) > Приложение Qt Widgets > ”Выбрать”



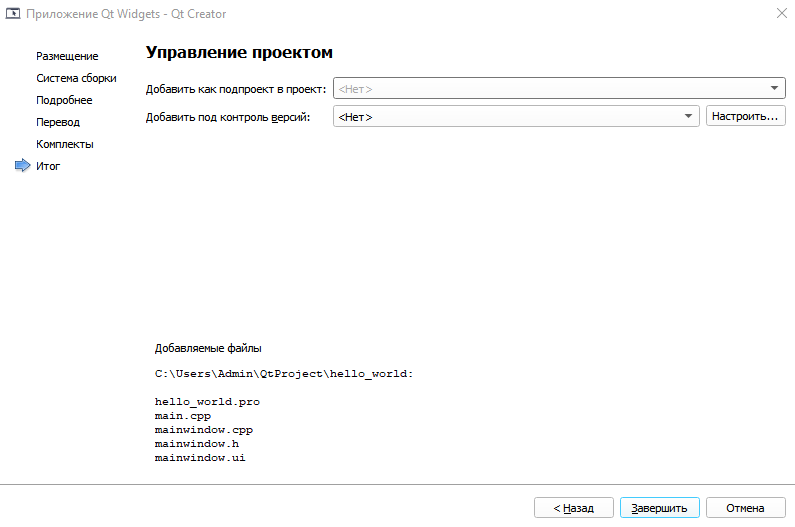
Укажите название проекта и директорию для его расположения, нажмите ”Далее”.



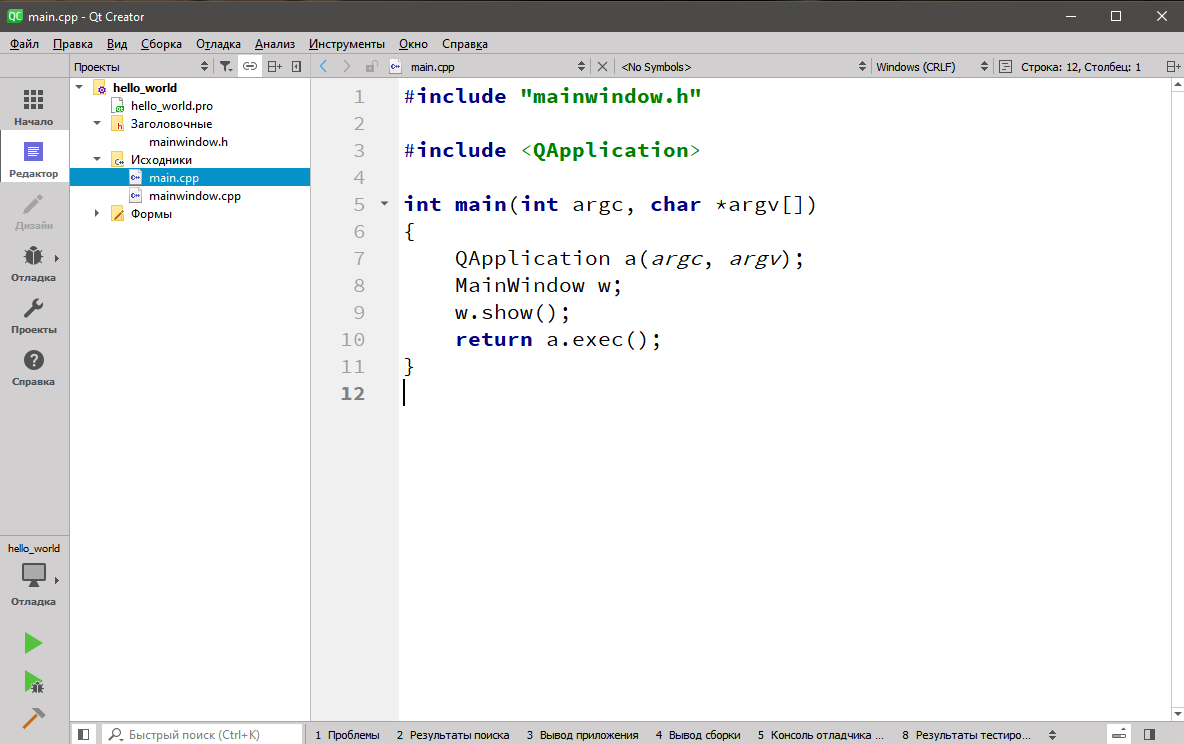
Во вкладках ”Система сборки”, “ Подробнее”, “ Перевод” оставим все по умолчанию и нажмем ”Далее”. На вкладке “Комплекты” выберем два основных компонента и продолжим создание проекта.



На последнем этапе можно указать родительский проект или подключить репозиторий системы контроля версий.



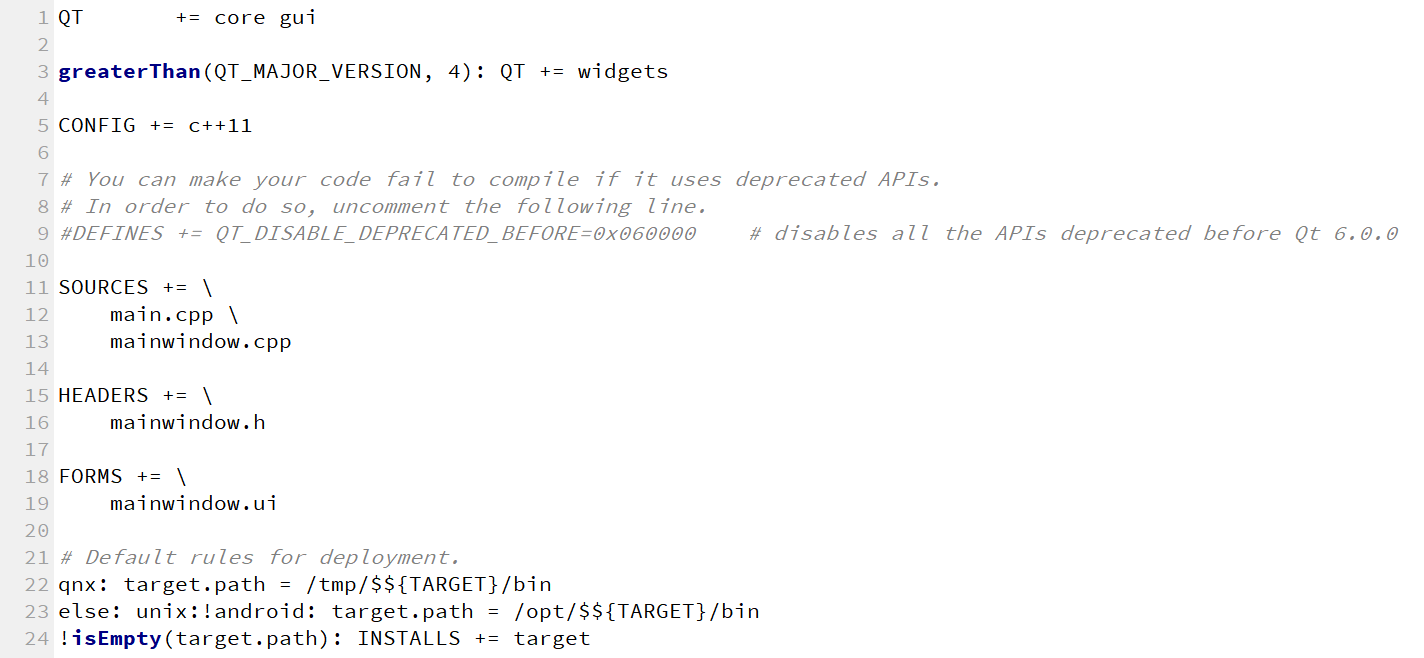
После завершения создания нового проекта появится следующее окно:



Автоматически открылось окно редактора. В левой части видно древо проекта. На данный момент в древе можно наблюдать только 4 файла. На самом деле их больше, но взаимодействовать с оставшимися пока не придется.

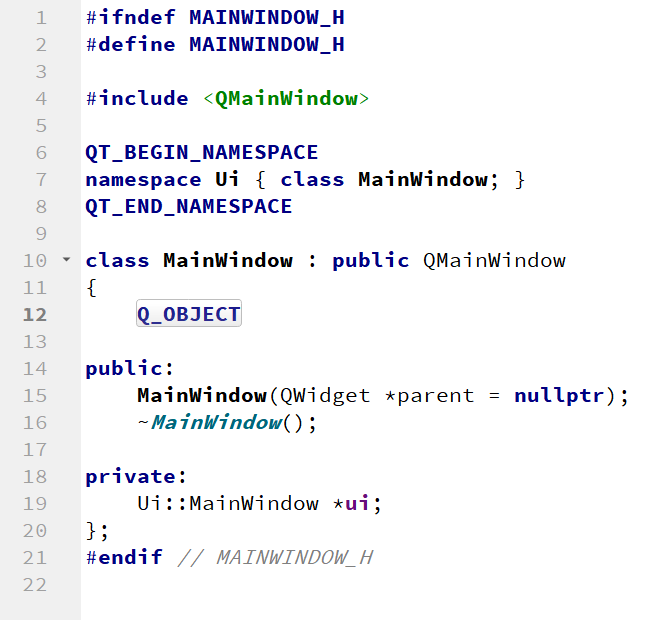
Пройдемся по содержимому каждого из файлов.

hello\_world.pro – файл конфигурации проекта.



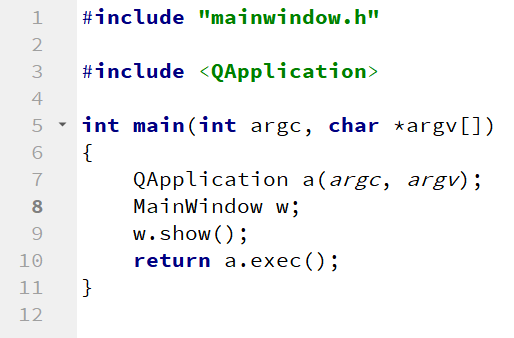
В нем указаны версия языка, названия файлов ресурсов, заголовочных файлов и форм. Нижняя часть отвечает за деплой проекта.

mainwindow.h – заголовочный файл, описывающий главный класс приложения.



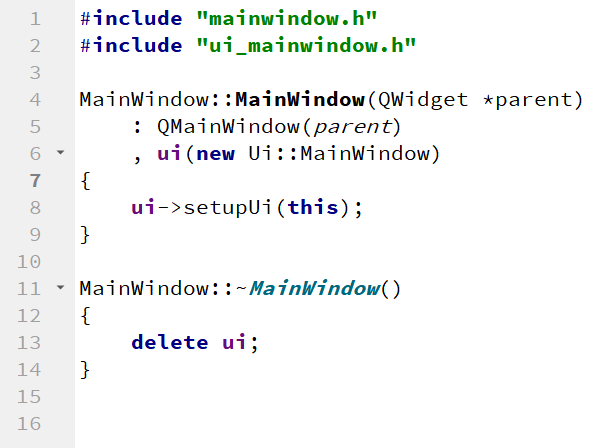
Выделим 19 строку. Указатель ui будет хранить в себе пространство объектов, находящихся на форме (панелей, кнопок, текстовых полей и т.д.). С помощью него в дальнейшем будем получать доступ к элементам интерфейса.

main.cpp – файл, в котором находится функция main, запускающая приложение.



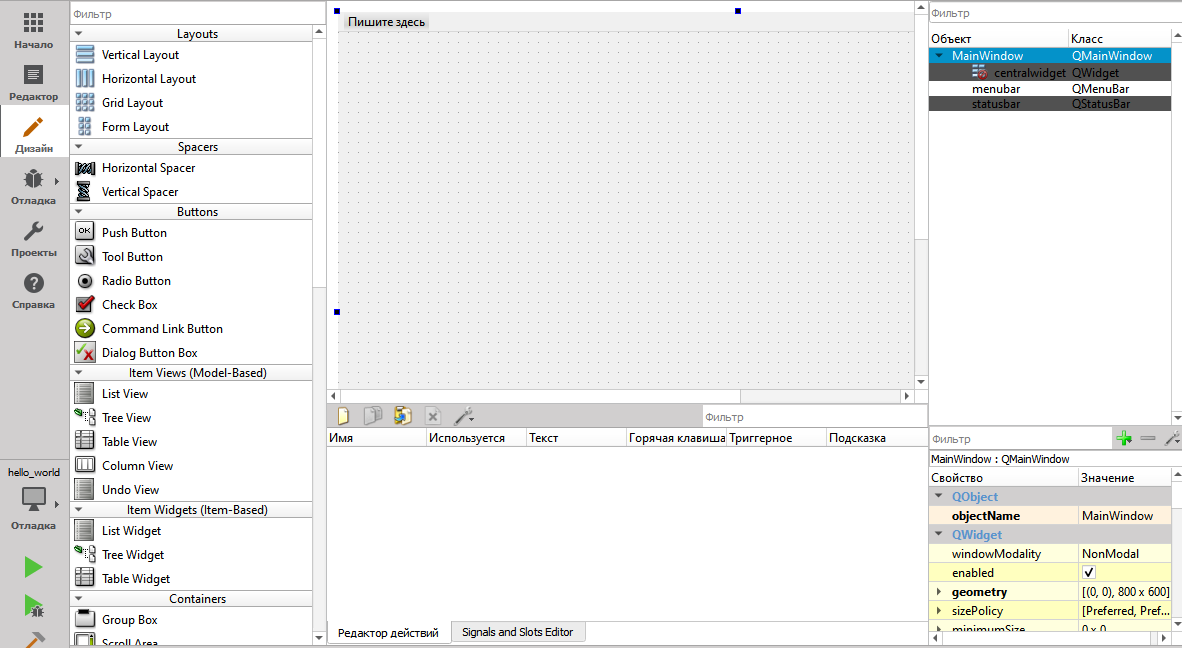
В понятие запуска входит как запуск второстепенных процессов (подключение к БД, первоначальные инициализации состояний), так и создание интерфейса, который будет наблюдать пользователь. Следует обратить внимание, что приложение может состоять из нескольких самостоятельных окон, поэтому не выгодно инициализировать данные как состояние конкретного окна (например, класса MainWindow).

mainwindow.cpp – файл с реализациями методов, которые описаны в mainwindow.h.



Изучив код, можно убедиться, что основная часть состояния приложения будет храниться в переменной ui.

mainwindow.ui – файл в формате xml, который хранит разметку внешней части приложения (frontend).



Этот файл можно открыть как в режиме формы, так и в текстовом, но документация гласит, что редактирование файла дописыванием xml кода крайне не рекомендуется.

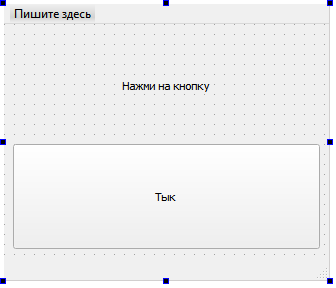
Окно редактирования формы состоит из нескольких компонентов:

* Список компонентов располагается в левой части окна. Элементы списка сгруппированы по принципу их использования.
* Форма, которая будет итоговым интерфейсом приложения, расположена по центру окна. Для добавления элемента на форму его достаточно перетащить.
* Древо объектов интерфейса находится в правом верхнем углу. В этой части удобно следить за отношениями элементов друг к другу (родительский – дочерний).
* Блок со свойствами выбранного элемента расположен в нижнем правом углу. Настройку расположения, размера шрифта, текстового содержимого можно выполнить, изменяя поля соответствующих полей. Следует отметить, что первое в списке свойство objectName является уникальным идентификатором. С помощью этого значения можно получить доступ к элементу в коде.

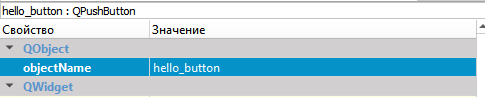
1. Hello World

На примере простого приложения разберем один из вариантов взаимодействия программной части и части с элементами интерфейса. Напишем программу, которая по нажатию кнопки будет менять исходный текст на “Hello World!”.

Начнем с расположения элементов на форме. Добавим label и pushButton.

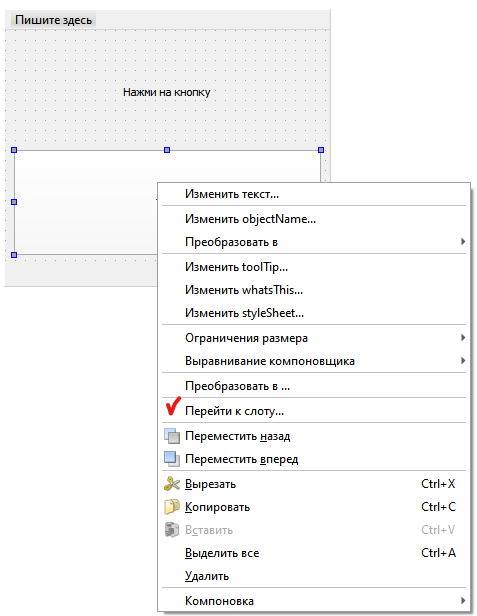




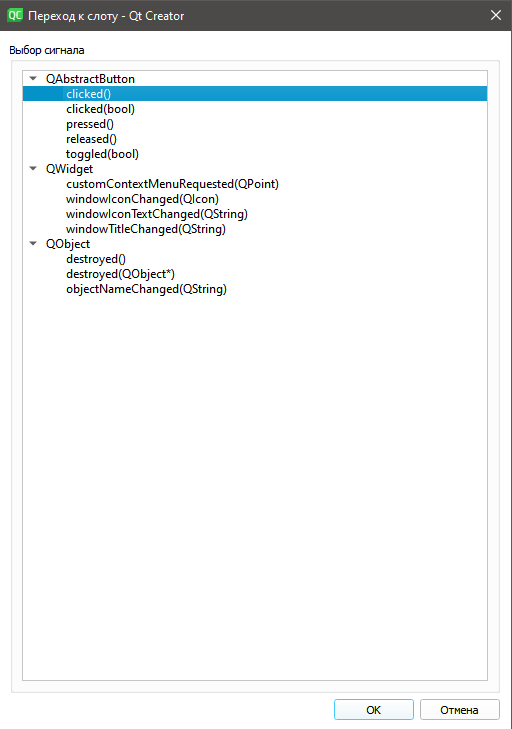


В поле свойства objectName можно оставить значения по умолчанию или повторить идентификаторы из примера. Текст на кнопке можно поменять двойным щелчком или в свойствах элемента.

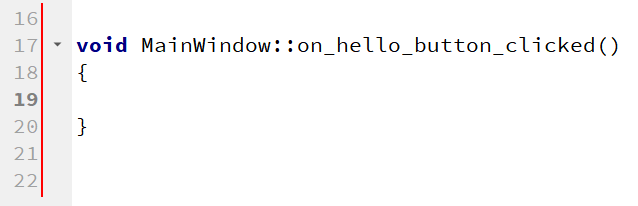
В Qt существует сигнально-слотовая система, которая позволяет добавлять “слушатели” на определенные события. Для того чтобы добавить действие, которое произойдет по нажатию кнопки, в контекстном меню pushButton выбирают пункт “Перейти к слоту”.



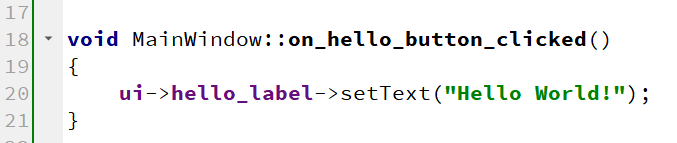
Выберите пункт clicked() и нажмите “Ок”.



Далее IDE перебросит в файл mainwindow.cpp, в котором создастся новый метод.



Его название сразу может подсказать, в какой момент будет вызван этот метод. Внутри него и реализуем смену текста в поле.



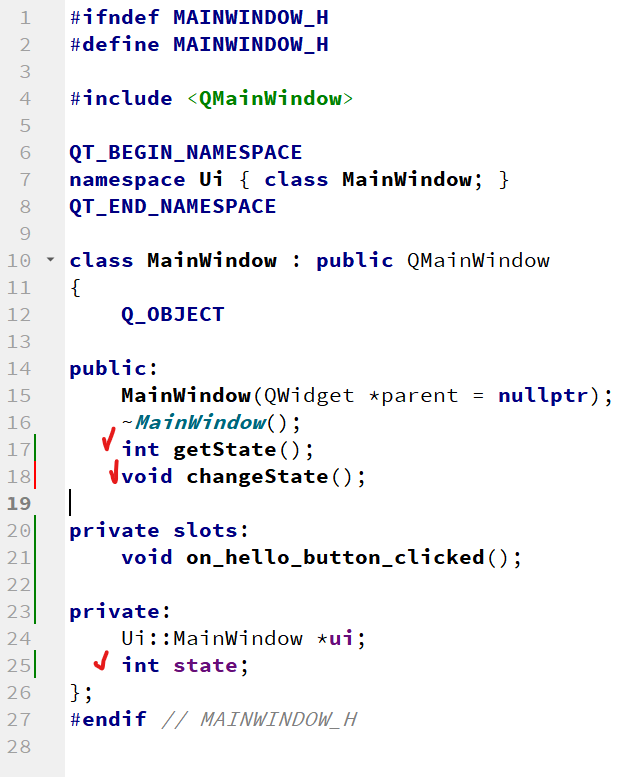
Как уже говорилось ранее, доступ к элементам происходит через указатель ui. Далее получаем элемент по его идентификатору (objectName) и используем интуитивно понятные методы для работы с состоянием элемента.

Запускаем приложение, проверяем его работу.

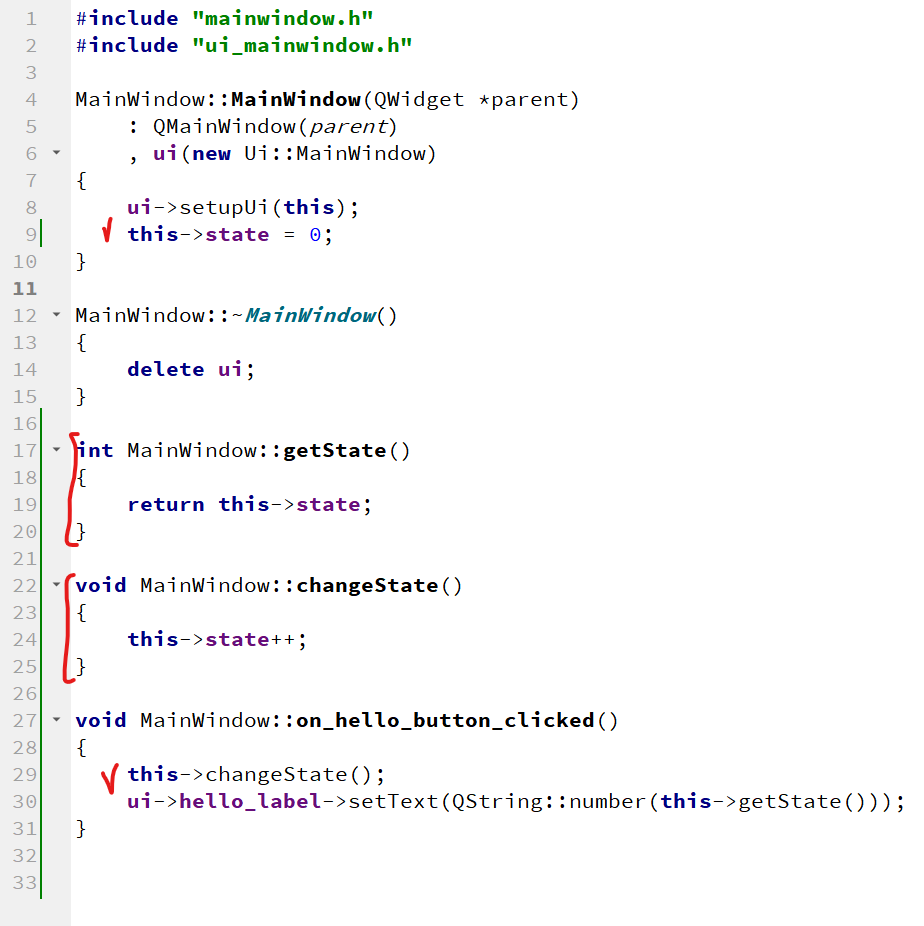
1. Кликер

Усовершенствуем эту программу. Добавим приложению состояние, которое будет меняться после каждого нажатия на кнопку. Простым примером будет кликер, инкрементирующий значение в текстовом поле. Первоначально значение будет равно 0.

В этот раз для наглядности объяснения будем хранить состояние внутри экземпляра класса MainWindow. Для этого объявим поле state и методы getState и changeState в файле mainwindow.h.



Следующим шагом реализуем их в файле minwindow.cpp.

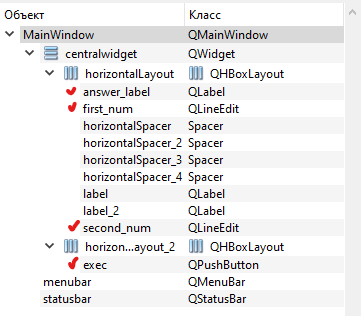
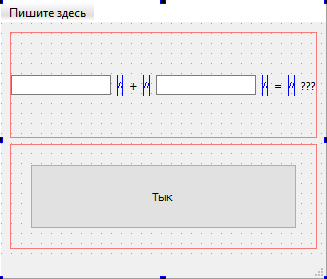


Осталось только подправить файл интерфейса таким образом, чтобы в текстовом поле стоял 0 и приложение готово.

1. Сумматор

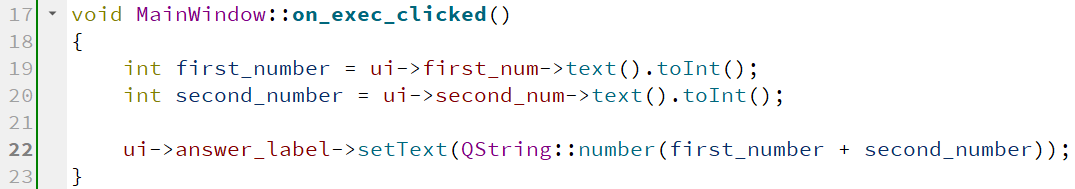
Теперь следует разобраться с тем, как получать данные из формы. Для примера реализуем программу-сумматор, состоящую из 2 полей для ввода, текстового поля для ответа, кнопки и некоторых других элементов (не влияющих на логику программы).

Интерфейс программы будет выглядеть следующим образом:



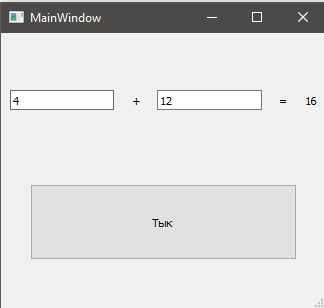
Следует обратить внимание на идентификаторы. С их помощью можно манипулировать данными в форме.

Создадим слушатель для кнопки (как в пункте 3) и добавим логику приложения.



Следует заметить, что основная логика повторяется. Для получения данных из формы, также как и для изменения, используется переменная ui.

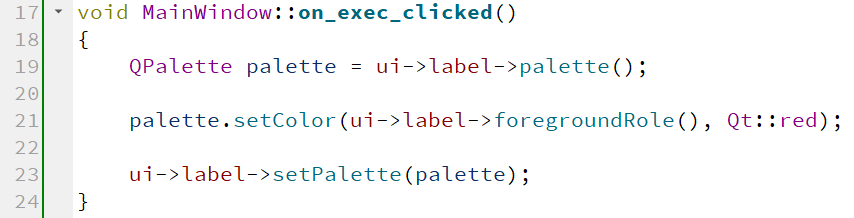
Работа приложения будет выглядеть следующим образом:



1. Стилизация

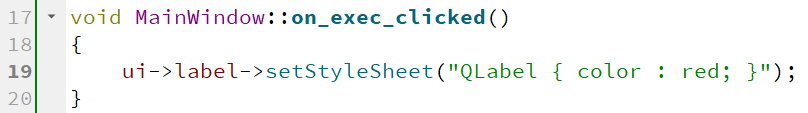
Qt позволяет стилизовать формы как программно, так и средствами css/html. Разберем несколько примеров:

В случае, если интерфейс нужно менять параллельно работе программы, удобнее воспользоваться доступом из кода. Для смены цвета текста в элементе label следует воспользоваться объектом QPallete.



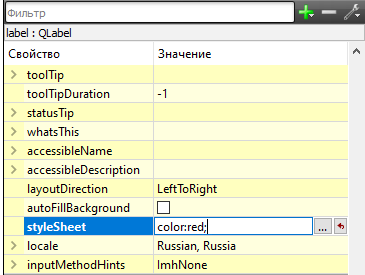
Представим, что у каждого элемента есть набор стилей, который описывает то, каким ему следует быть. В 19 строке мы получаем этот набор и в 21 строке меняем соответствующие значения. Далее возвращаем набор обратно элементу в строке 23.

К такому же результату приведет и альтернативный вариант с использованием css стилей.

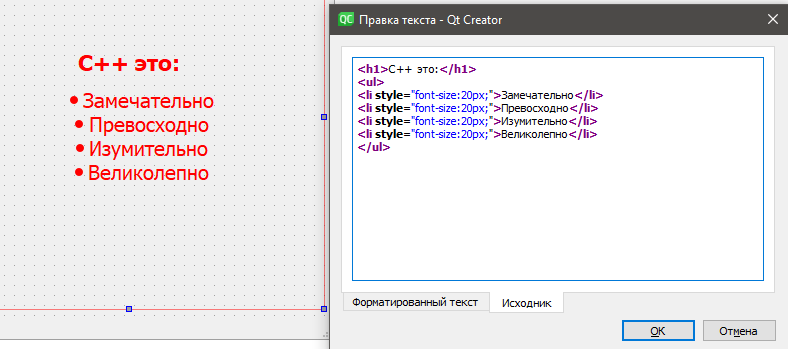


Следует заметить, что таким образом невозможно без создания собственных структур и методов хранения получить доступ к установленному цвету в будущем.

Если дизайн элемента заранее известен и не планирует изменяться после компиляции проекта, то удобнее воспользоваться свойством stylesheet.



Еще один способ стилизации вашего приложения – использование html тегов в текстовых полях. Например, нужно разработать справку к приложению, состоящую из заголовка и списка. В таком случае можно воспользоваться тегами h и li.



Важное замечание – текстовый редактор для этого открывать не обязательно, можно использовать теги и через двойной щелчок мыши по label.

Всеми перечисленными способами можно влиять на любые компоненты формы. Например, таким образом можно стилизовать и кнопку:

