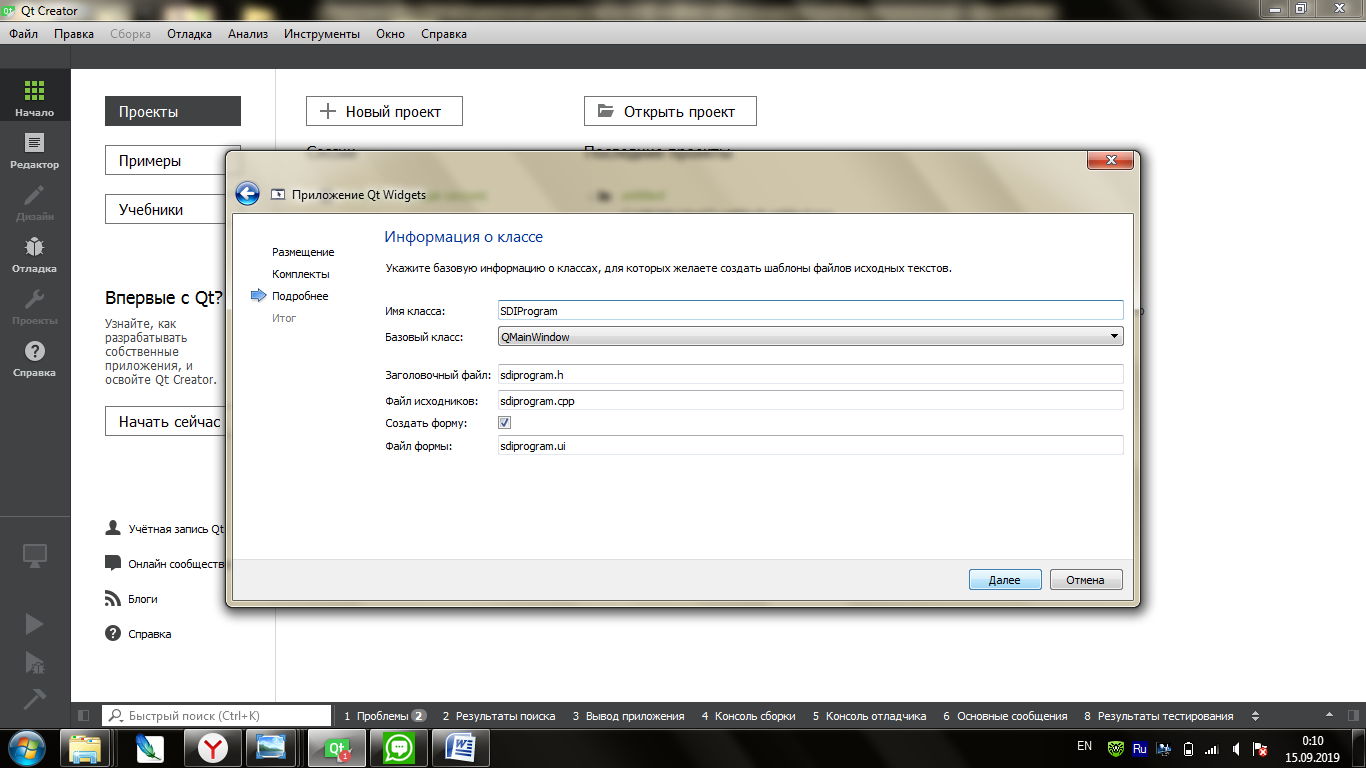
**Цель:** Создание SDI- приложения (Single Document Interface, однодокументный интерфейс). Простой текстовый редактор. Окно заставки.

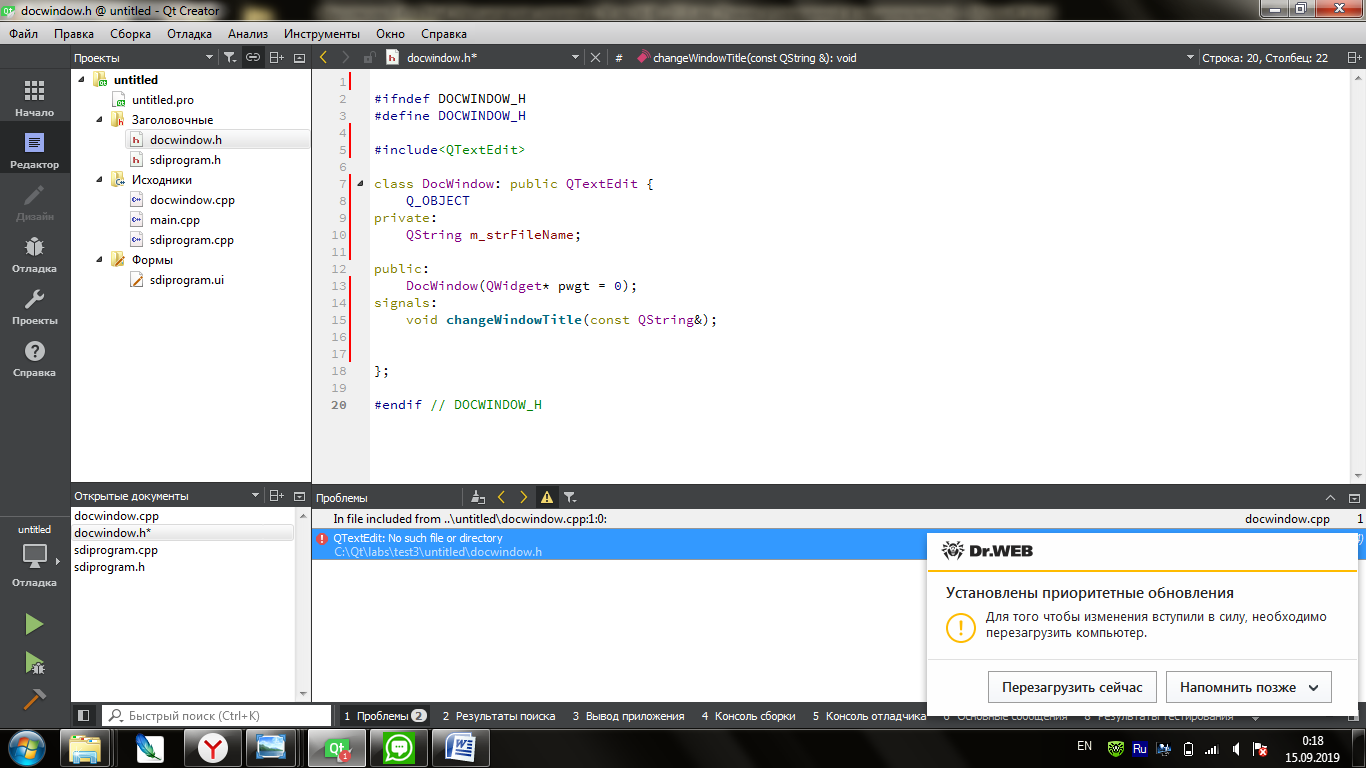
В названиях создаваемых классов добавлять свои фамилии.

**Задание:**

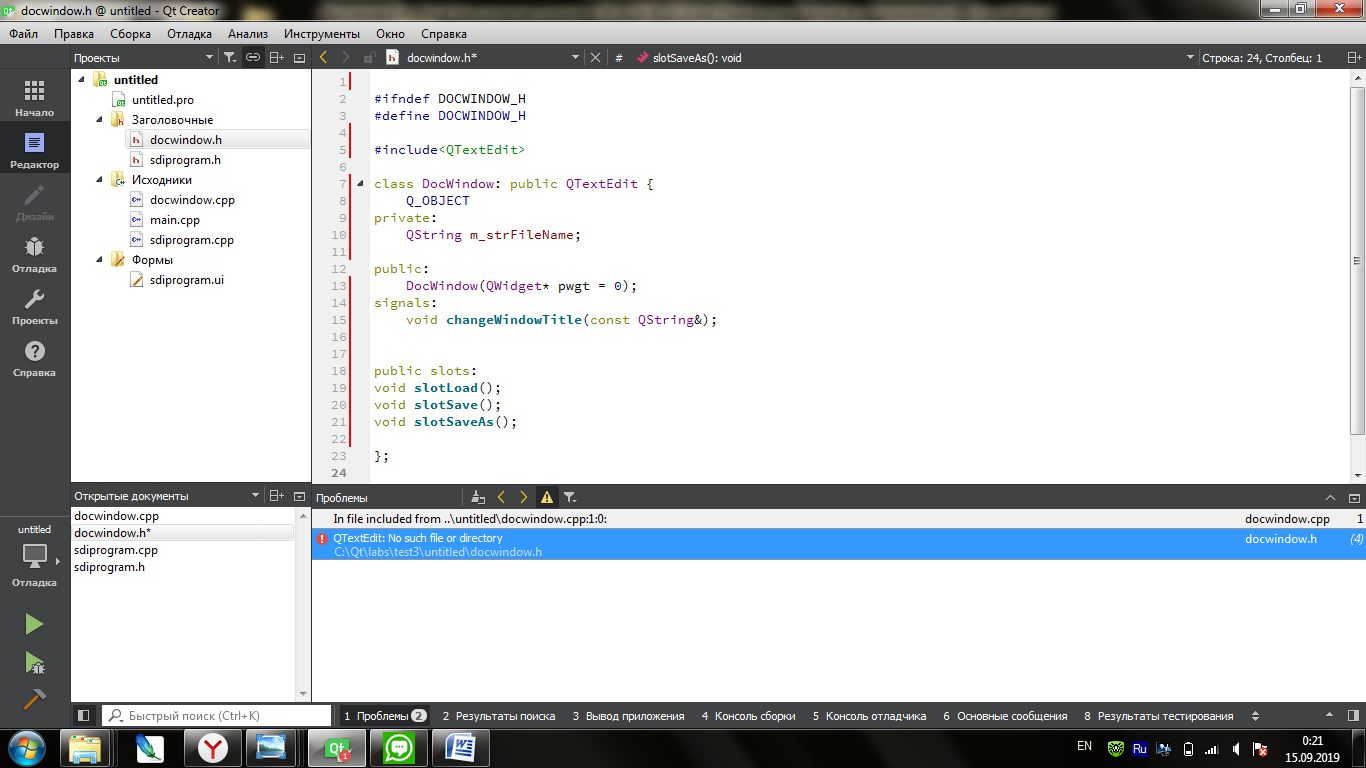
1. В SDI-приложениях рабочая область одновременно является окном приложения, а это значит, что в одном и том же таком приложении невозможно открыть сразу два документа. Типичным примером SDI-приложения является программа Блокнот (Notepad) из состава ОС Windows. Реализуем упрощенный вариант этой программы – простой текстовый редактор. Запустите QtCreator и создайте приложение Qt Widgets. При этом, смените имя класса на SDIProgram (к названию добавить свои фамилии):



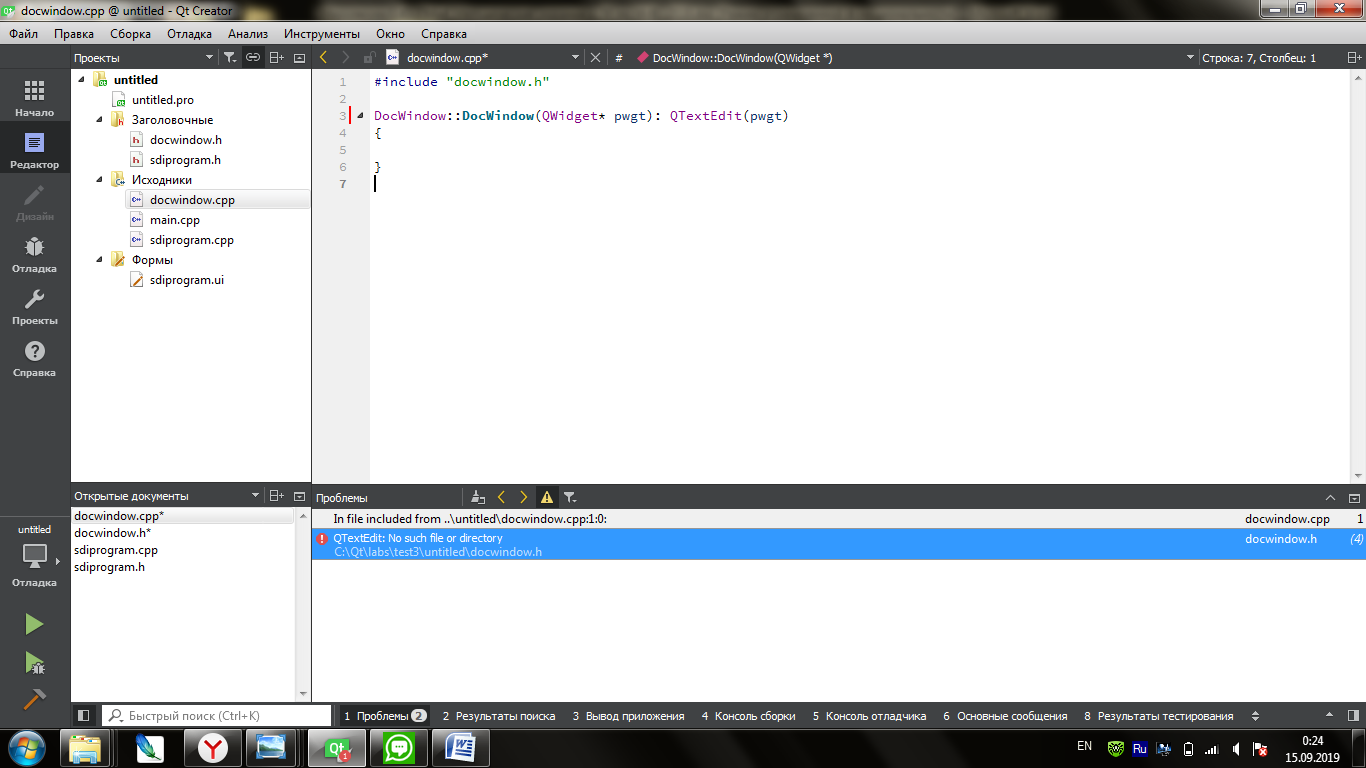
1. Создайте класс DocWindow (к названию добавить свои фамилии), унаследованный от класса QTextEdit. Он представляет собой окно для редактирования. В его определении содержится атрибут m\_strFileName, в котором хранится имя изменяемого файла. Сигнал changeWindowTitle() предназначен для информирования о том, что текстовая область заголовка должна быть изменена. Файл DocWindow.h



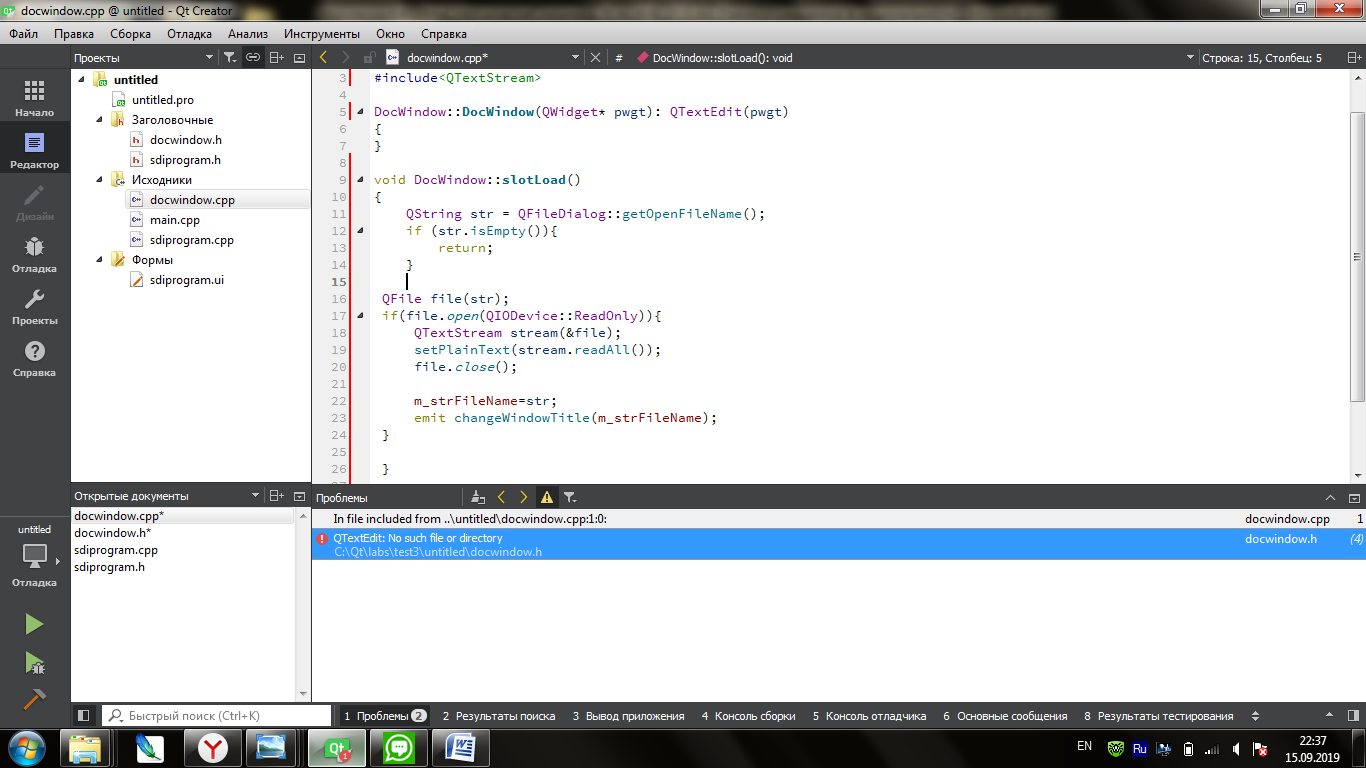
1. Слоты slotLoad(), slotSave() и slotSaveAs() необходимы для проведения операций чтения и записи файлов:



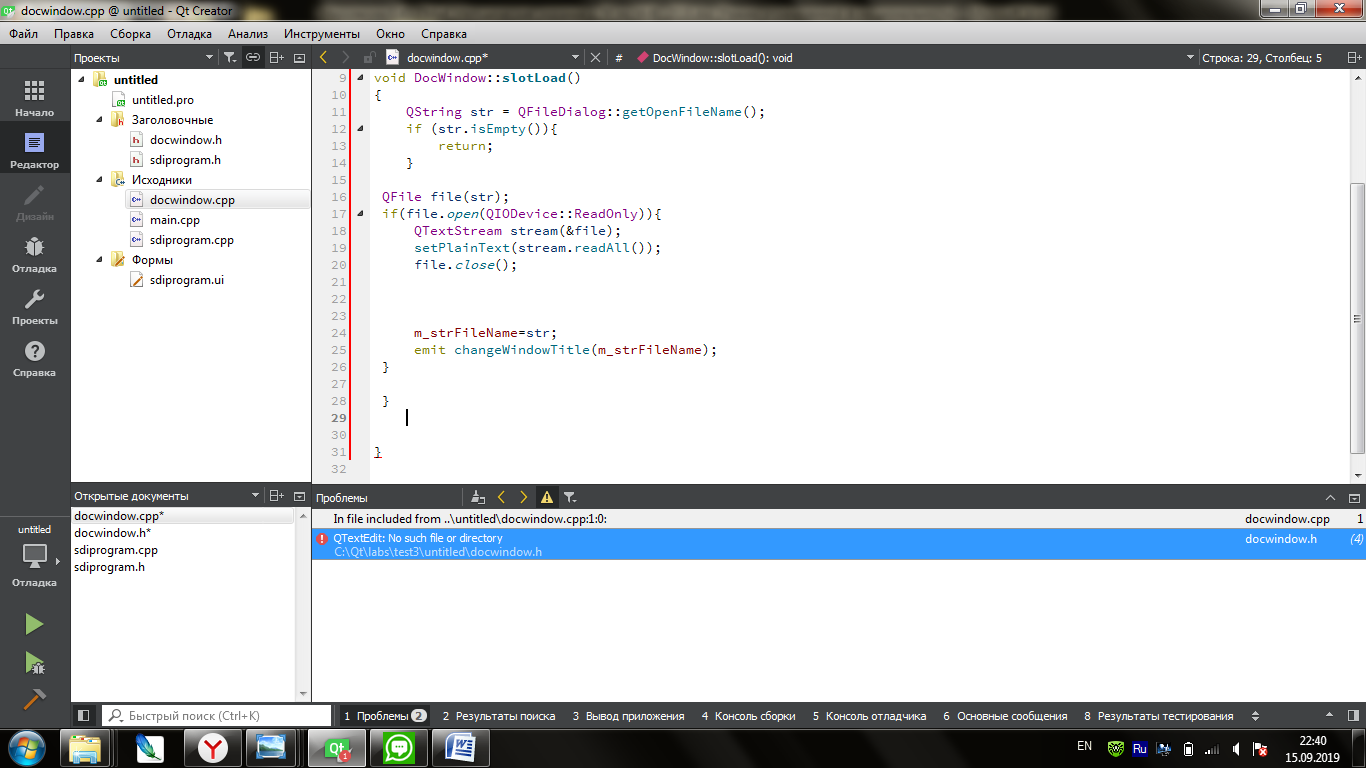
1. В конструктор класса (DocWindow.cpp) передается указатель на виджет предка:



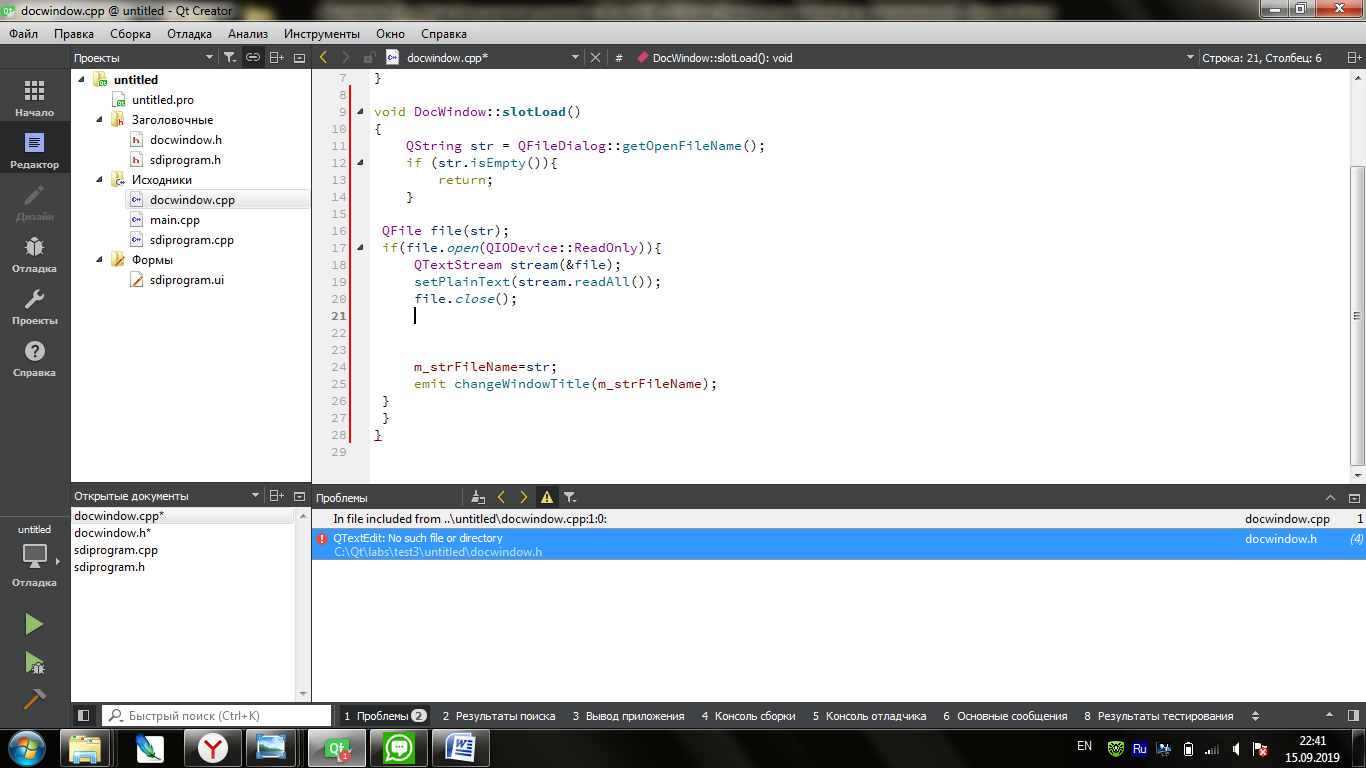
1. Метод slotLoad() отображает диалоговое окно открытия файла вызовом статического метода QFileDialog::getOpenFileName(), с помощью которого пользователь выбирает файл для чтения. В том случае, если пользователь отменит выбор, нажав на кнопку Cancel, этот метод вернет пустую строку. В данном случае это проверяется с помощью метода QString::isEmpty():



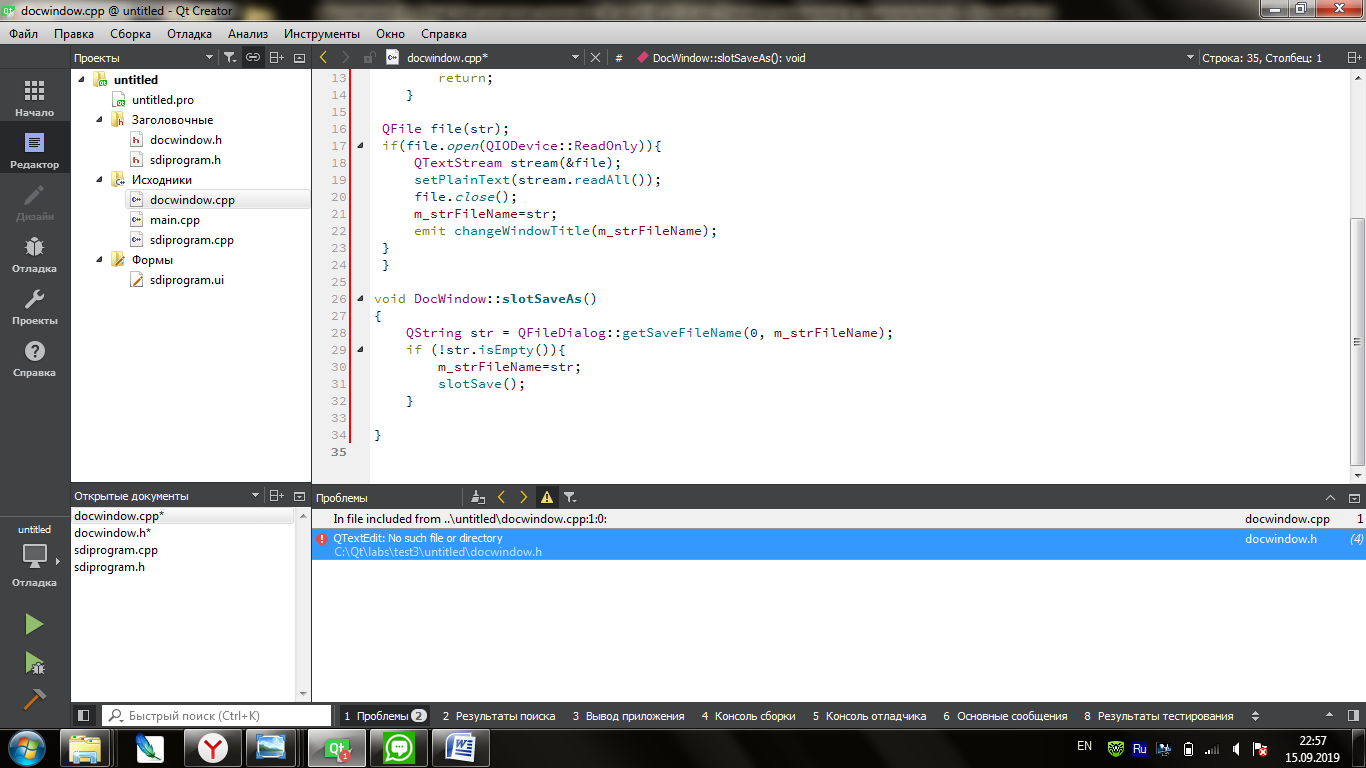
1. Если метод getOpenFileName() возвращает непустую строку, то будет создан объект класса QFile, проинициированный этой строкой. Передача QIODevice::ReadOnly в метод QFile::open() говорит о том, что файл открывается только для чтения. В случае успешного открытия файла создается объект потока stream, который здесь используется для чтения текста из файла. Чтение всего содержимого файла выполняется при помощи метода QTextStream::readAll(), который возвращает его в объекте строкового типа QString. Текст устанавливается в виджете методом setPlainText(). После этого файл закрывается методом close():



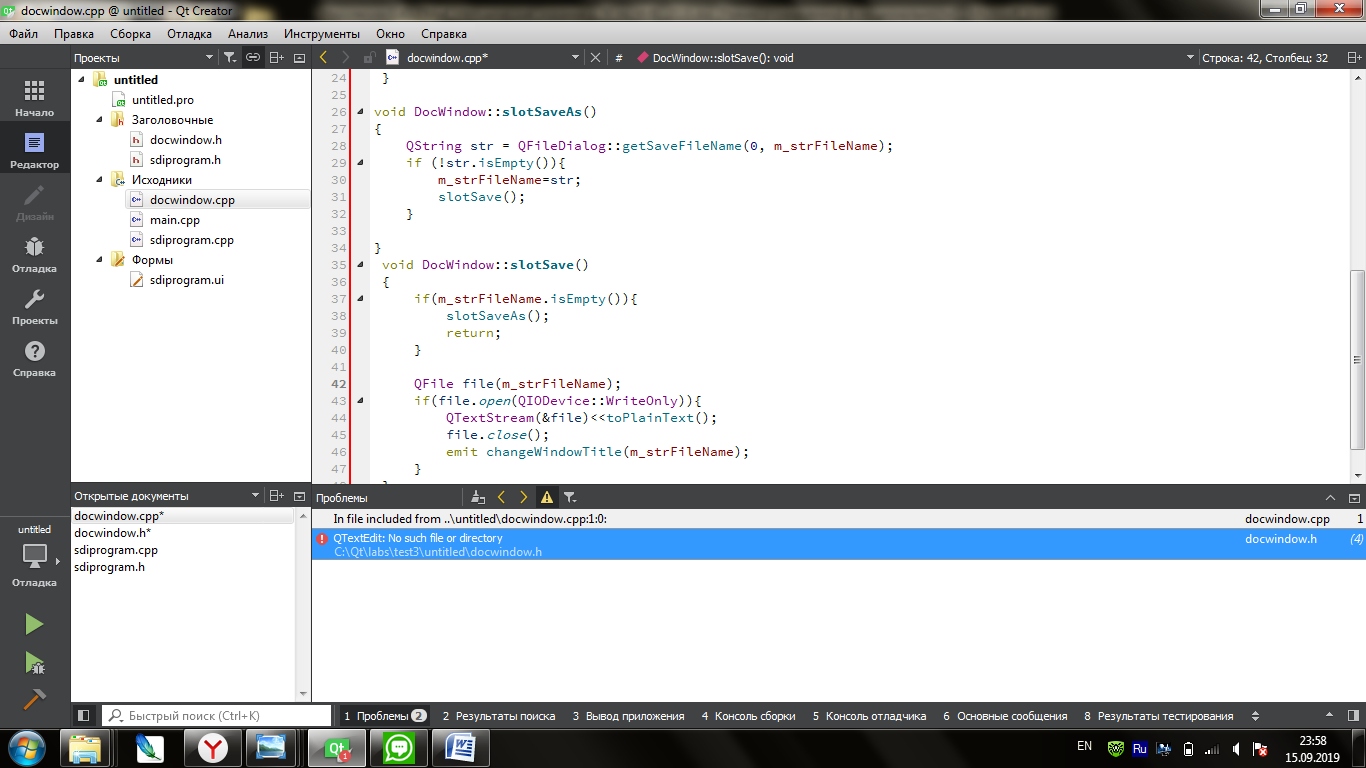
1. Об изменении местонахождения и имени файла оповещается отправкой сигнала changeWindowTitle(), для того чтобы использующее виджет DocWindow приложение могла отобразить эту информацию, изменив заголовок окна:



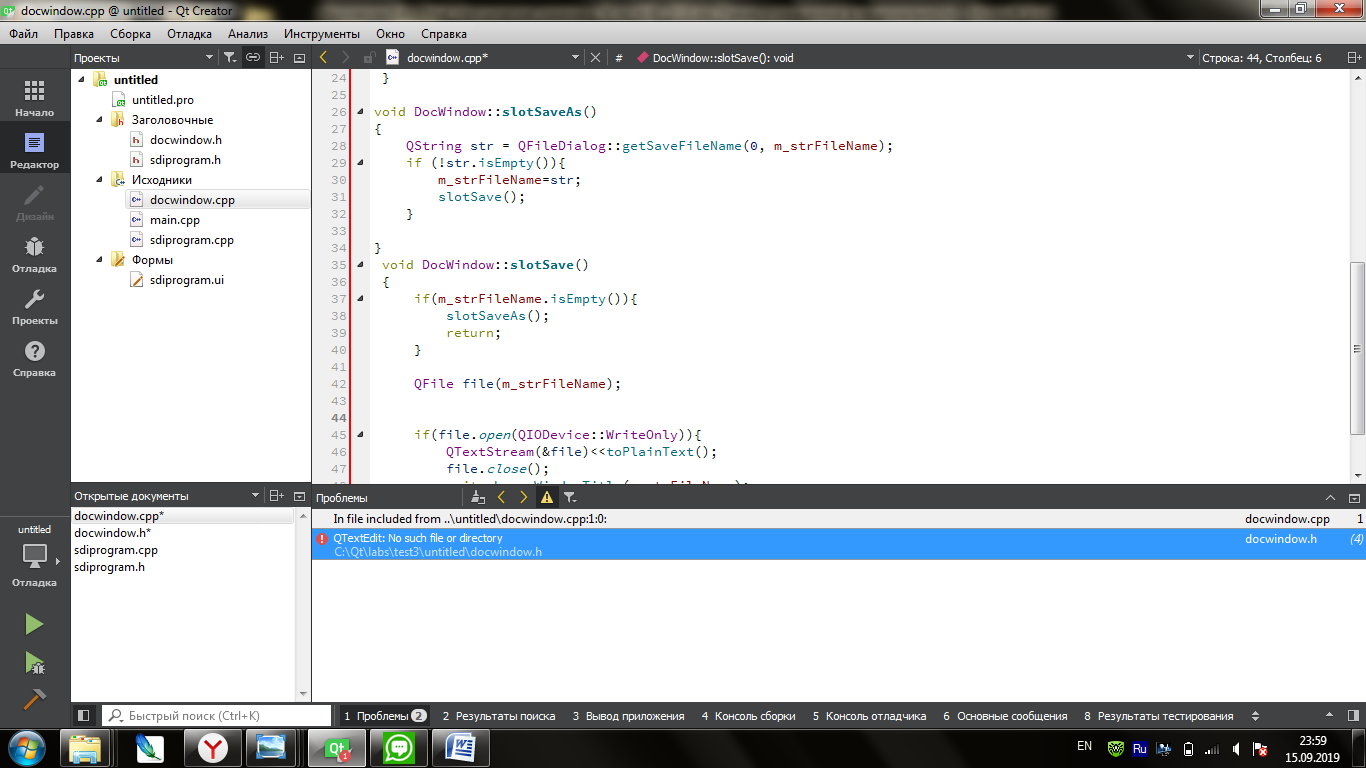
1. Следующий слот slotSaveAs() отображает диалоговое окно сохранения файла с помощью статического метода QFileDialog::getSaveFileName(). Если пользователь не нажал в этом окне кнопку Cancel, и метод вернул непустую строку, то в атрибут m\_strFileName записывается имя файла, указанное пользователем в диалоговом окне, и вызывается слот slotSave():



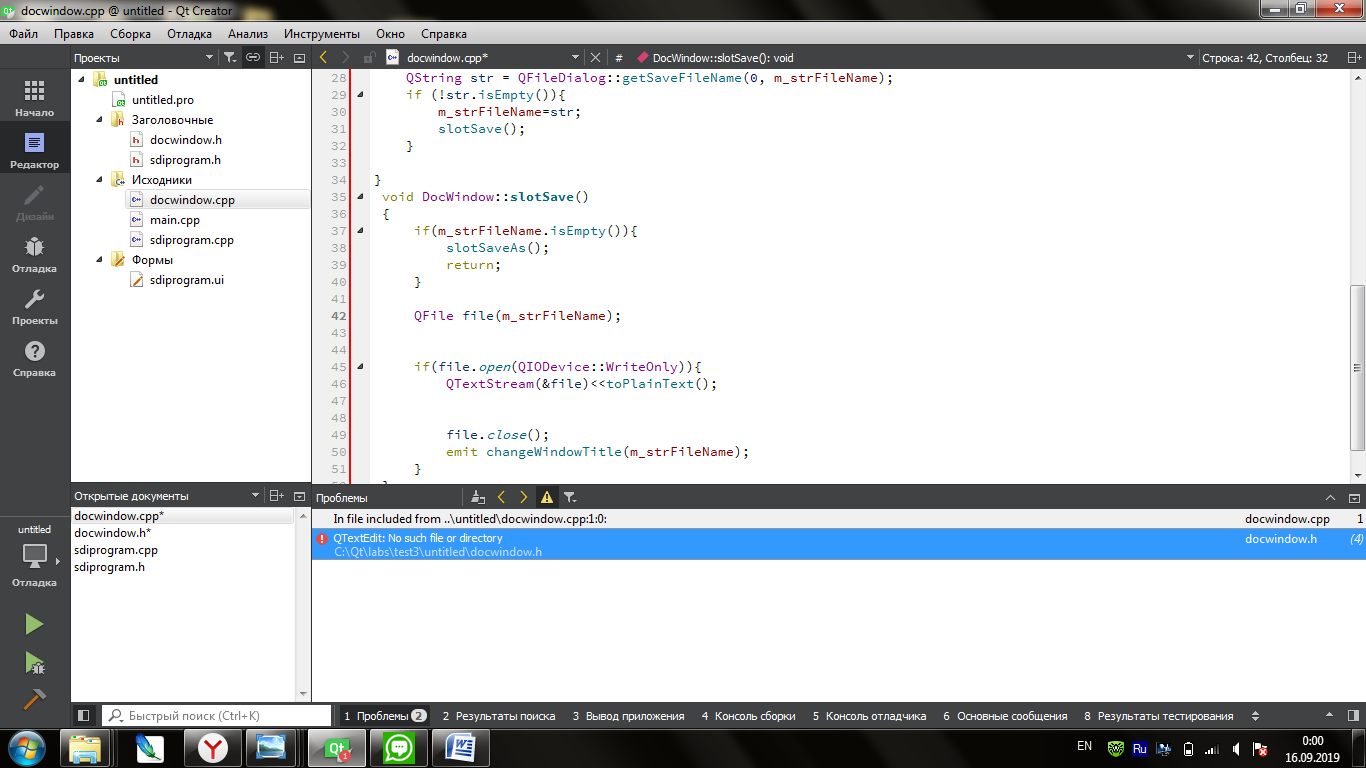
1. Запись в файл представляет собой более серьезный процесс, чем считывание, так как она связана с рядом обстоятельств, которые могут сделать ее невозможной. Например, на диске не хватит места, или он будет не доступен для записи. Сначала проверим, не пустая ли строка с именем файла. В этом случае нужно вызвать другой слот:



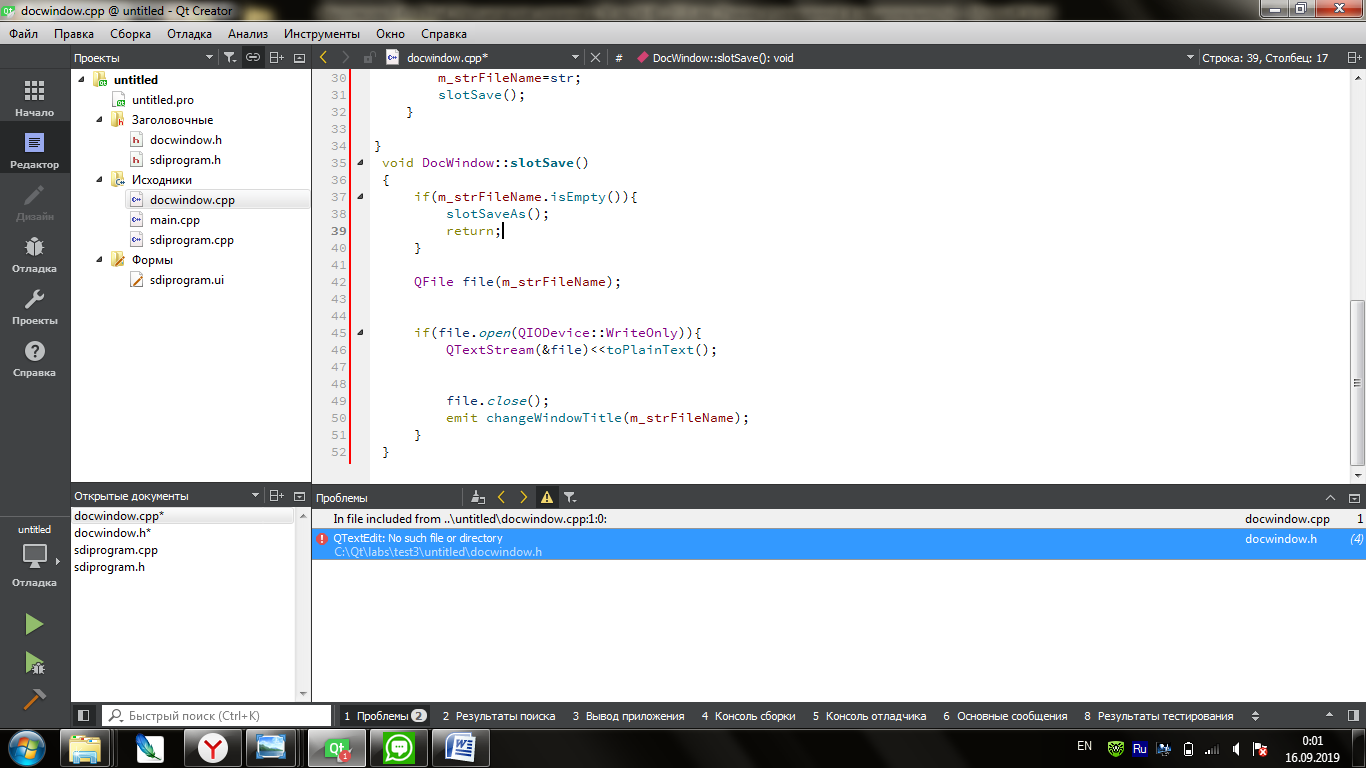
1. Затем для записи в файл нужно создать объект класса QFile и передать в него строку с именем файла.



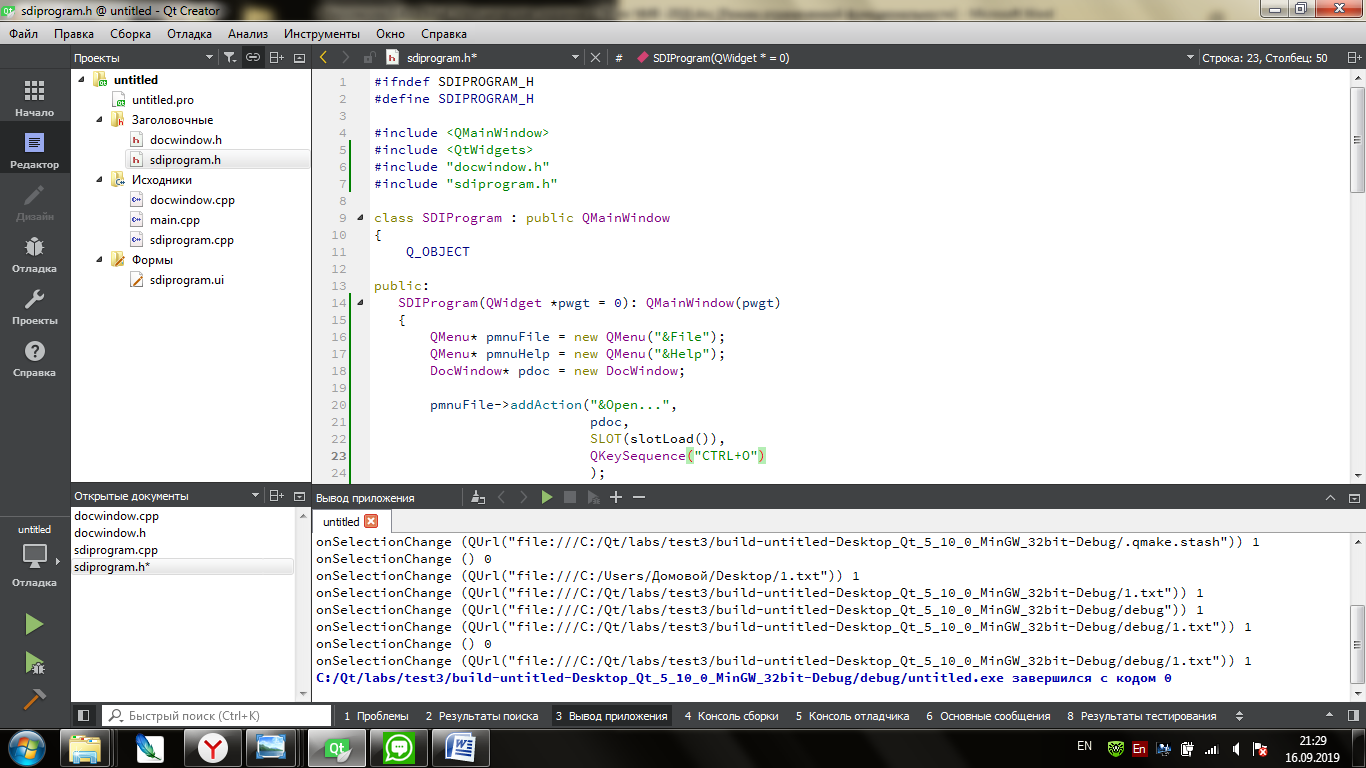
1. Затем надо вызвать метод open(), передав в него значение QIODevice::WriteOnly (флаг, говорящий о том, что будет выполняться запись в файл). В том случае, если файла с таким именем на диске не существует, он будет создан. Если существует – он будет открыт для записи. Если файл открыт успешно, то создается промежуточный объект потока, в который при помощи оператора << передается текст виджета, возвращаемый методом toPlainText().



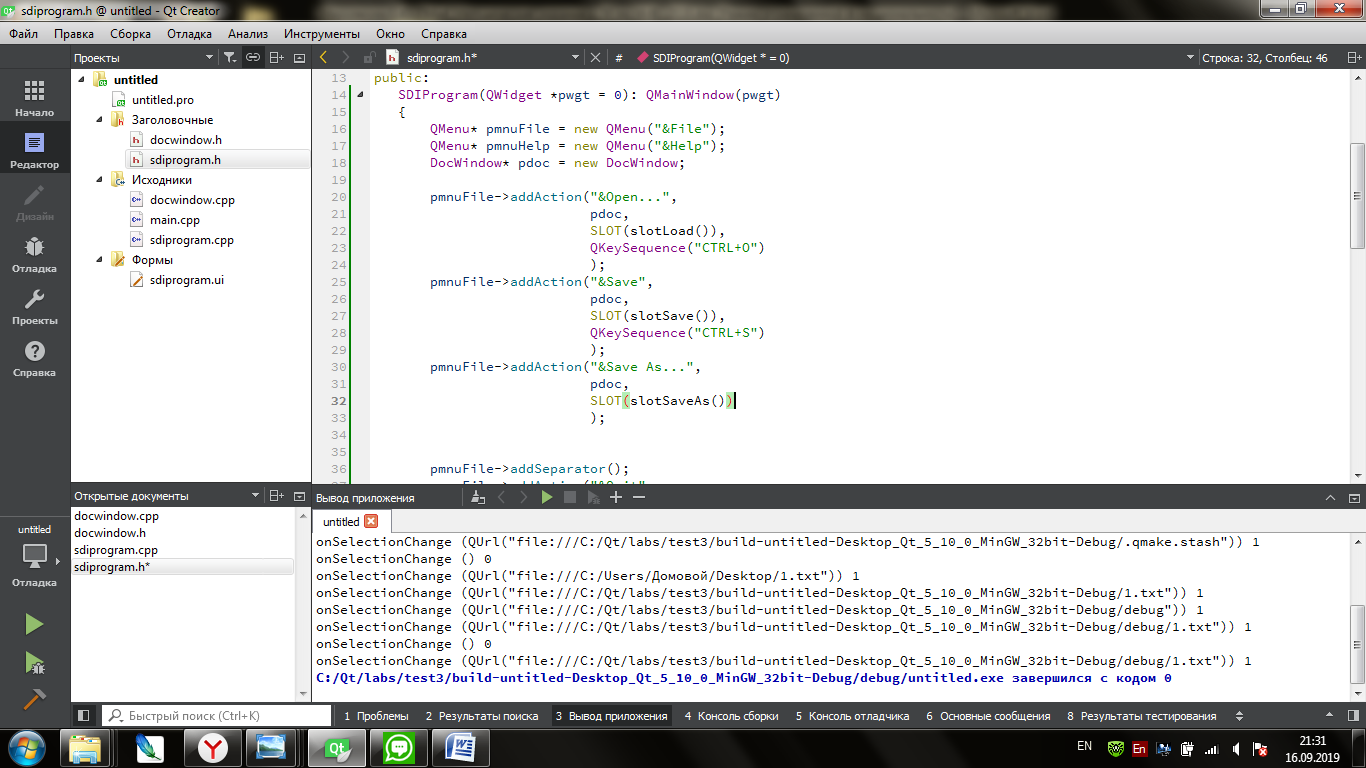
1. После этого файл закрывается методом QFile::close(), и отсылается сигнал с новым именем и местонахождением файла. Это делается для того, чтобы эту информацию могло отобразить приложение, использующее наш виджет DocWindow.



1. Добавьте окно информационного сообщения после успешного сохранения файла.
2. Класс SDIProgram унаследован от класса QMainWindow. В его конструкторе создаются три виджета: всплывающие меню File – указатель pmnuFile, Help – указатель pmnuHelp и виджет созданного вами окна редактирования – указатель pdoc.



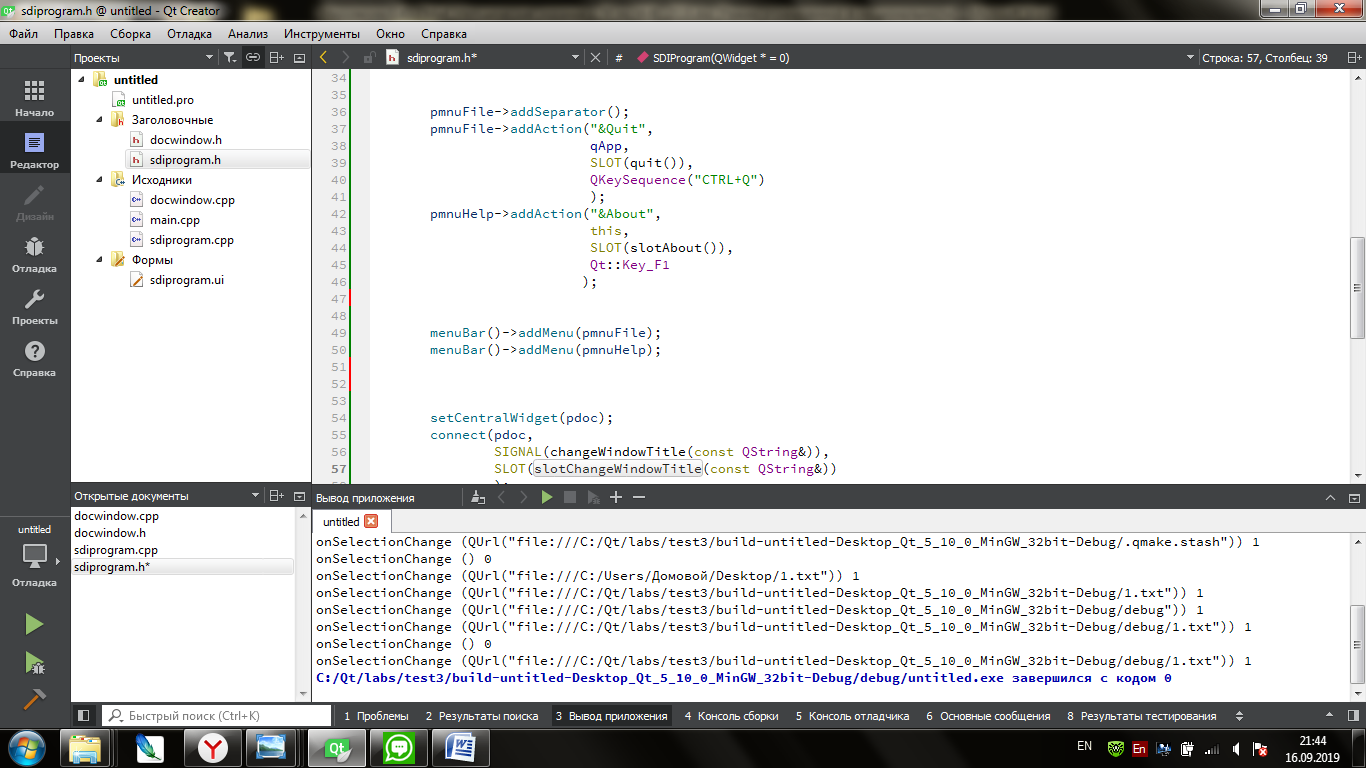
1. Затем несколькими вызовами метода addAction() неявно создаются объекты действий и добавляются в качестве команд меню. Третьим параметром указывается слот, с которым должна быть соединена команда, во втором параметре указан сам объект, который содержит этот слот. Таким образом, команда Open… соединяется со слотом slotLoad():



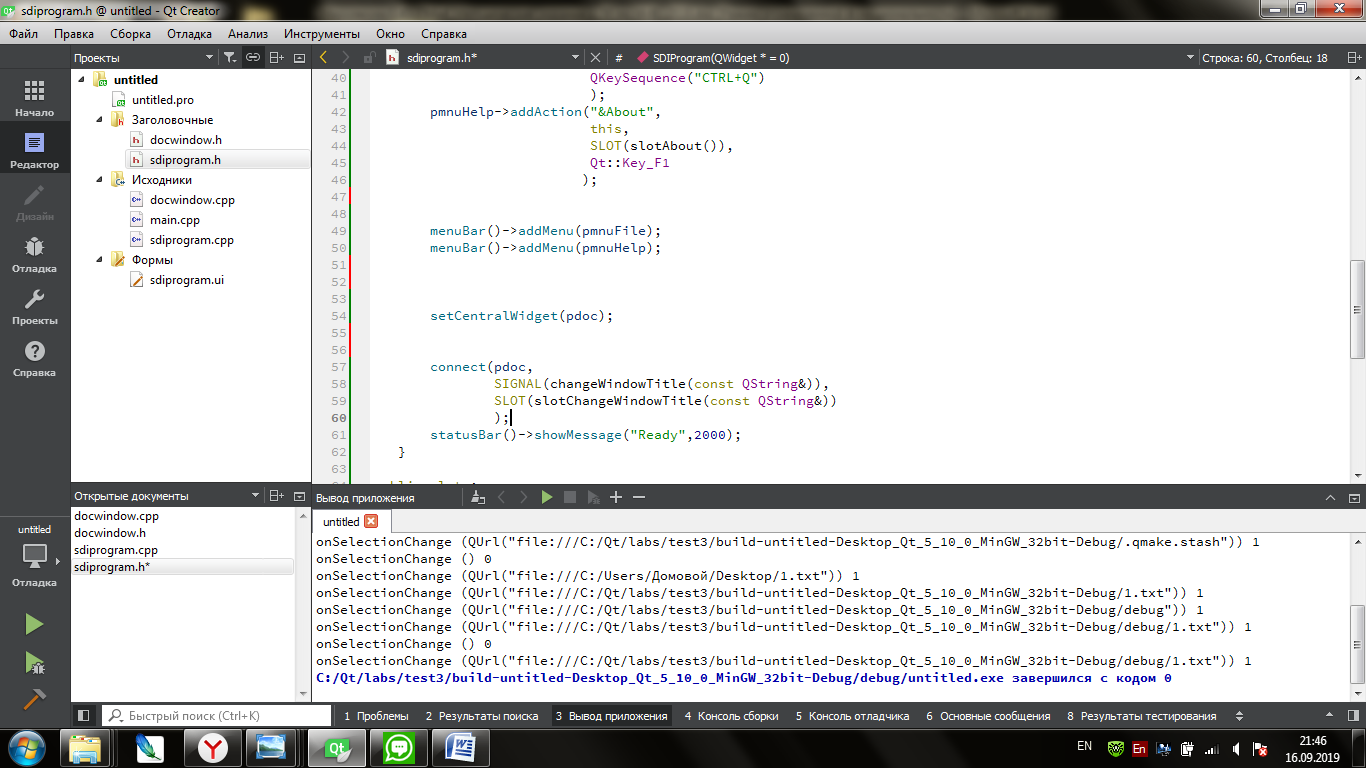
1. Добавьте аналогично остальные четыре действия: команда Save – со слотом slotSave, а команда Save As… - со слотом slotSaveAs(). Все эти слоты реализованы в классе DocWindow (обратите внимание на второй параметр, pdoc). Затем добавьте разделитель в меню. Далее, команда Quit соединяется со слотом quit() (второй параметр уже будет qApp).

Команда About должна находиться в Help (вместо pmnuFile пишем pmnuHelp), она соединяется со слотом slotAbout(), предоставляемым классом SDIProgram (поэтому второй параметр this). Быстрый доступ для About необходимо задать клавишей F1, указав в последнем параметре Qt::Key\_F1.

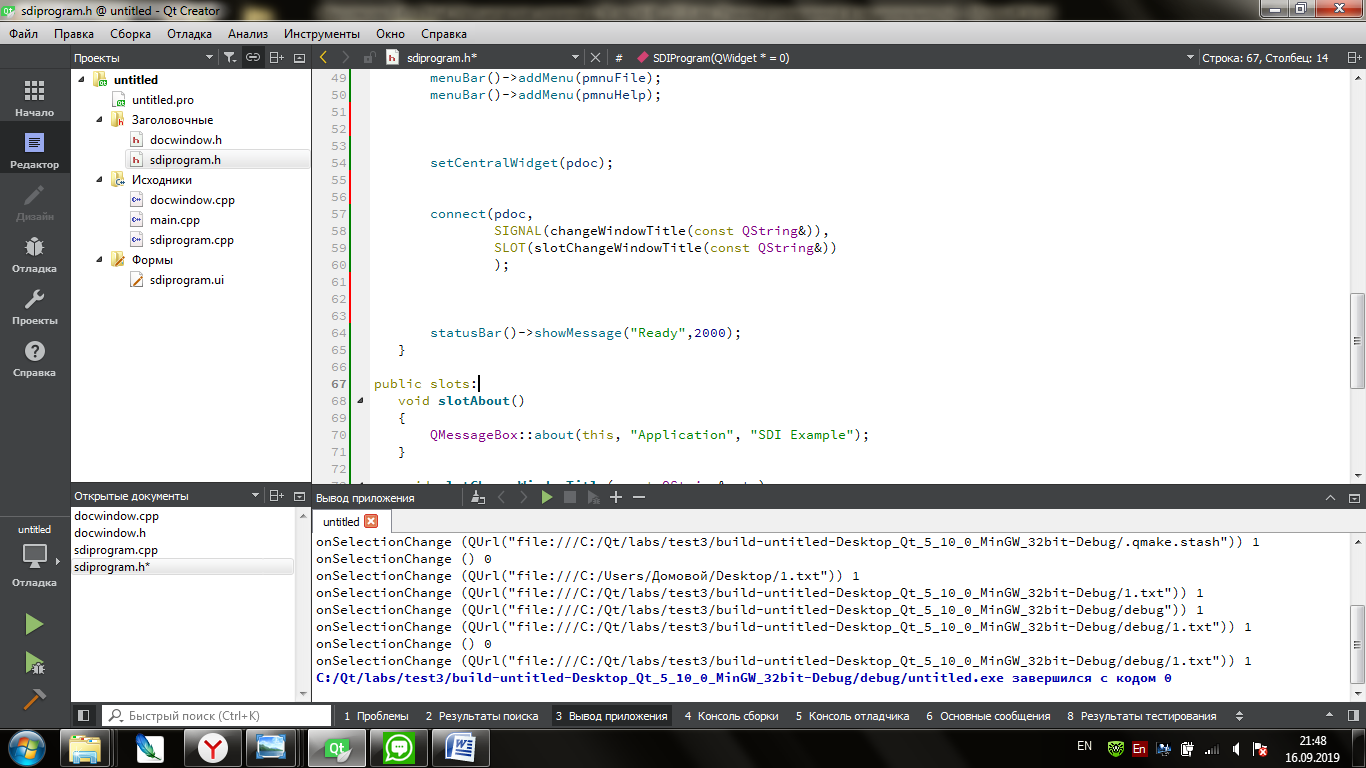
1. Затем menuBar() возвращает указатель на виджет меню верхнего уровня, а вызов методов addMenu() добавляет созданные всплывающие меню File и Help:



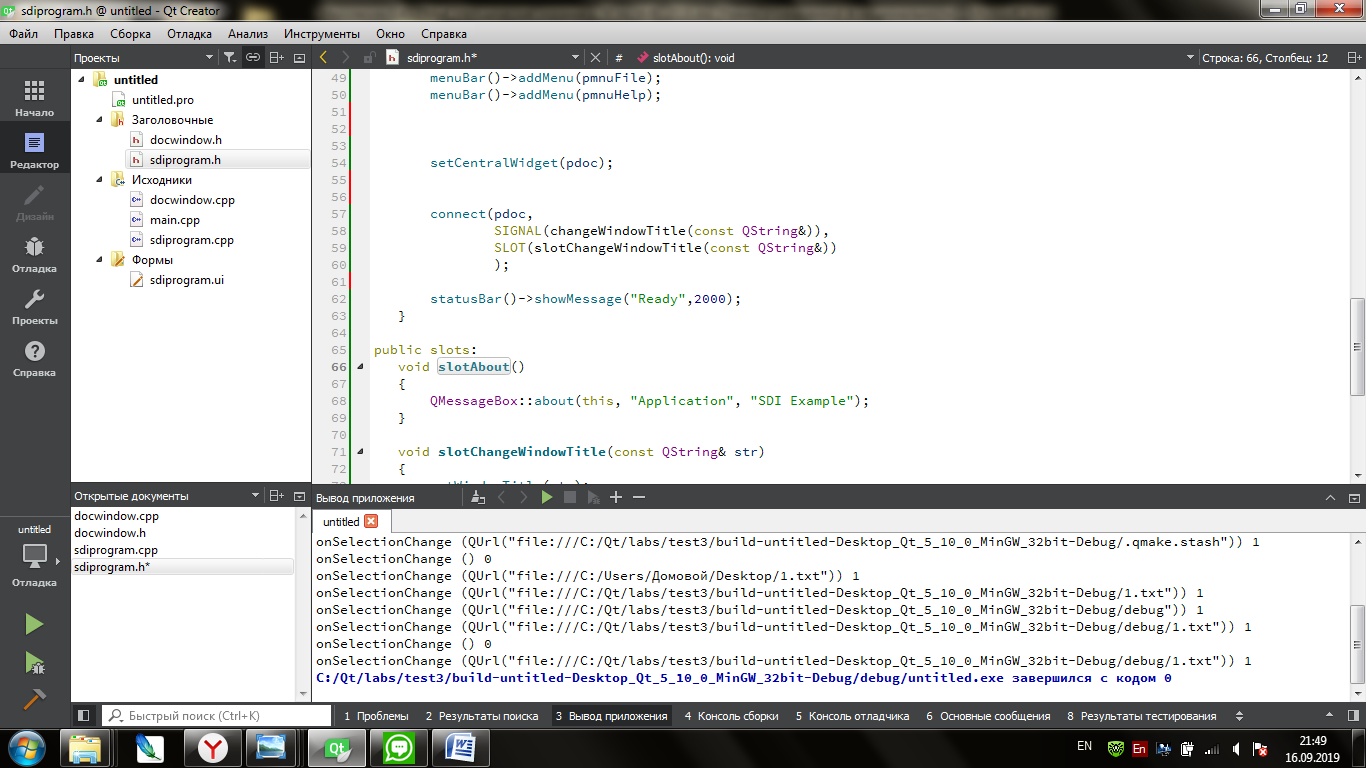
1. Вызов метода setCentralWidget() делает окно редактирования центральным виджетом, то есть рабочей областью вашей программы.



1. Для изменения текстового заголовка программы после загрузки файла или сохранения его под новым именем сигнал changeWindowTitle(), отправляемый виджетом окна редактирования, соединяется со слотом slotChangeWindowTitle().



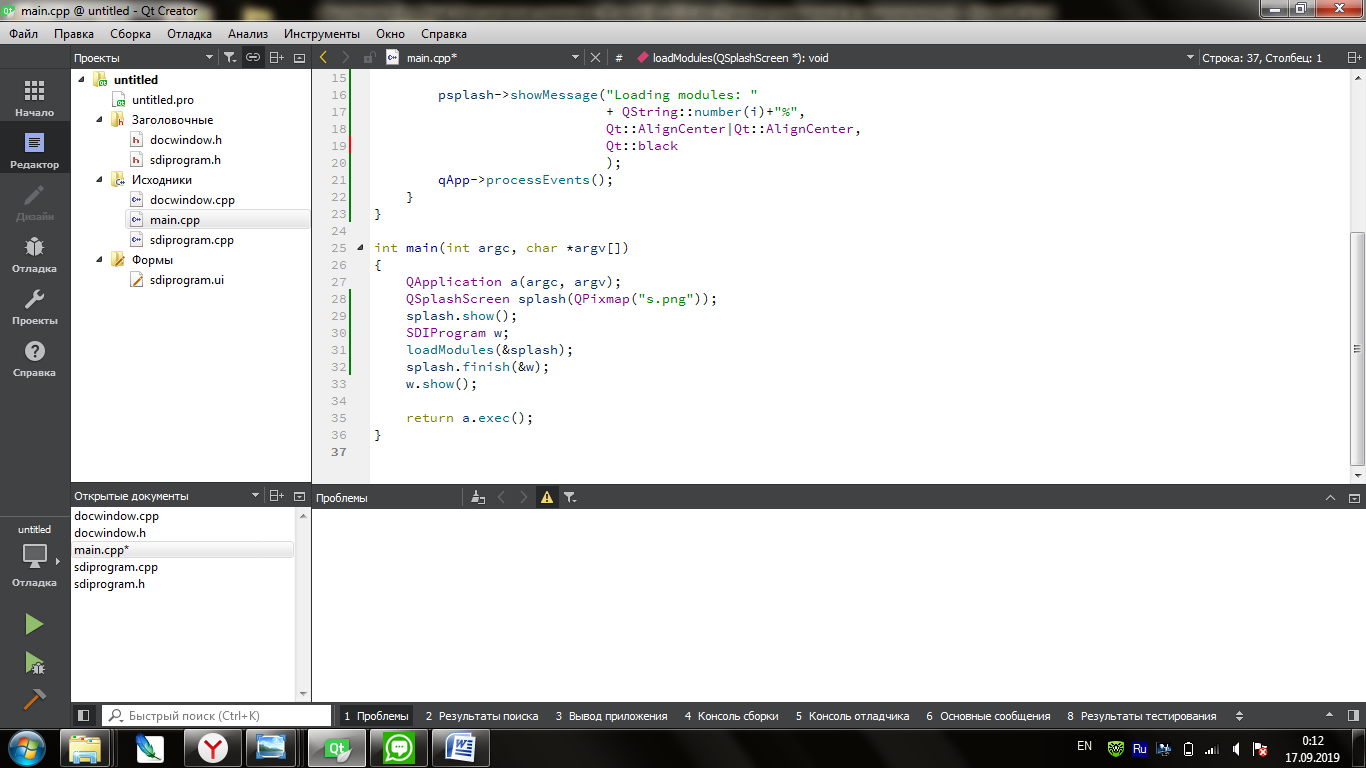
1. Метод showMessage(), вызываемый из виджета строки состояния, отображает надпись Ready на время, установленное во втором параметре (это 2 сек.)



1. Осталось добавить слоты:



1. Теперь можно запустить программу и проверить ее работу.
2. Замените надпись в окне About на свои фамилии и группу.
3. Добавьте аналогичным образом еще один пункт в меню File -> Color. При выборе данного пункта должно открыться диалоговое окно выбора цвета. Затем выбранный цвет должен быть установлен для текста методом setTextColor(). После чего попробуйте напечатать текст.
4. Добавим окно заставки. В библиотеке Qt такое окно реализовано в классе QSplashScreen. Объект этого класса создается в функции main() до вызова метода exec() объекта приложения. В конструктор передадим растровое изображение, которое будет отображаться после вызова метода show(). Самим приложением, которое должно быть запущено является w:



1. Функция loadModules() является эмуляцией загрузки модулей программы, в нее передается адрес объекта окна заставки, чтобы функция могла отображать информацию о процессе загрузки. Объект класса QTime служит для того, чтобы значение переменной i увеличивалось только по истечении 40 мсек. Отображение информации выполняется при помощи метода showMessage(), в который первым параметром передается текст, вторым - расположение текста, а третьим – цвет текста. Вызов метода finish() закрывает окно заставки. В этот метод передается указатель на главное окно приложения, и появление этого окна приводит к закрытию окна заставки:



1. Запустите приложение, проверьте его работу.