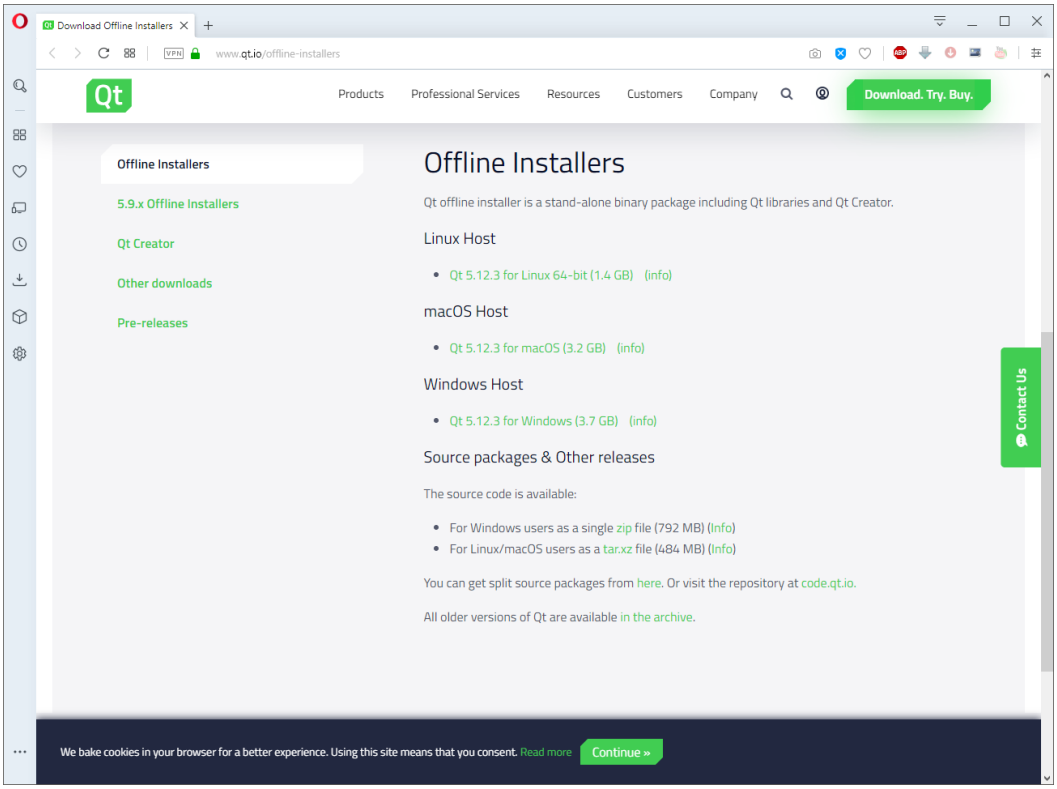
Установка Qt Windows + Visual Studio 2017, Linux, MacOS + XCode.

Оглавление

Скачивание дистрибутивов Qt

Для всех платформ скачать дистрибутивы Qt можно по ссылке: https://www.qt.io/offline-installers
(или гуглить на сочетание «Qt offline installer»).
Официальная страница с оффлайн дистрибутивами Qt под разные платформы выглядит следующим образом:



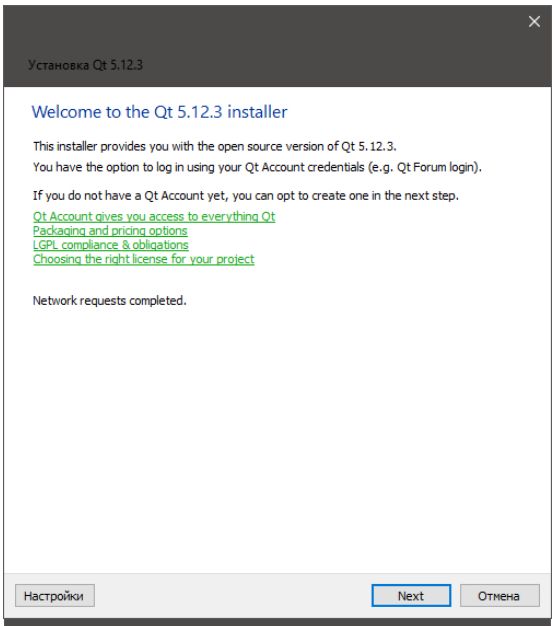
Необходимо с этой страницы скачать свой дистрибутив.
Ниже я рассмотрю процесс установки, настройки среды программирования и создания проекта под все платформы.

Установка Qt 5.12.3 под ОС Windows.

Запускаю дистрибутив:
qt-opensource-windows-x86-5.12.3.exe

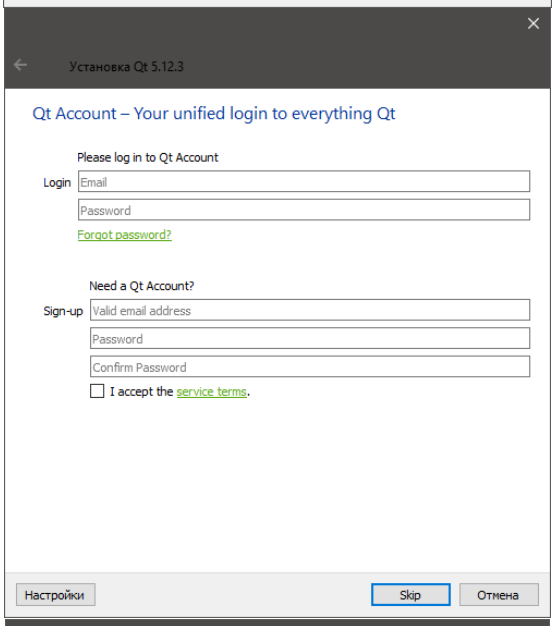
Дистрибутив занимает более 3.5 Гб, ему нужно время, чтобы загрузиться в память, процесс не быстрый.

Админских прав дистрибутив не потребует, так как его задача, по сути, просто распаковаться.



Нажимаем Next.

Предлагает ввести логин и пароль, если вы зарегистрированы на сайте Qt как разработчик. Я нажимаю Skip.

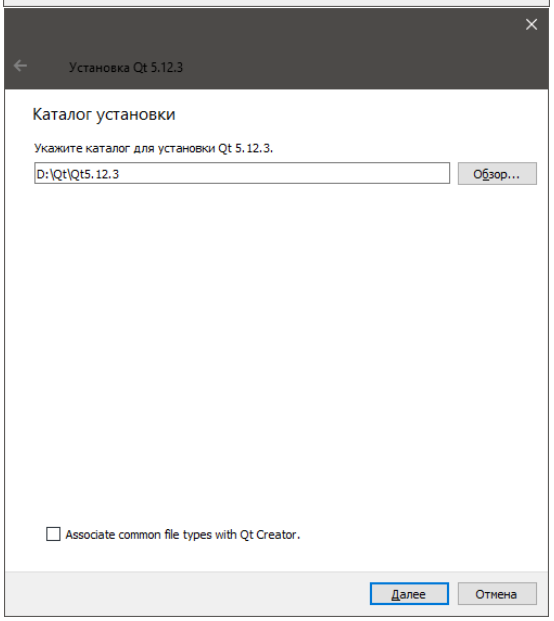


Следующее окно пропускаем, ничего информативного там нет, кроме надписи «Welcome to open source Qt 5.12.3 setup»

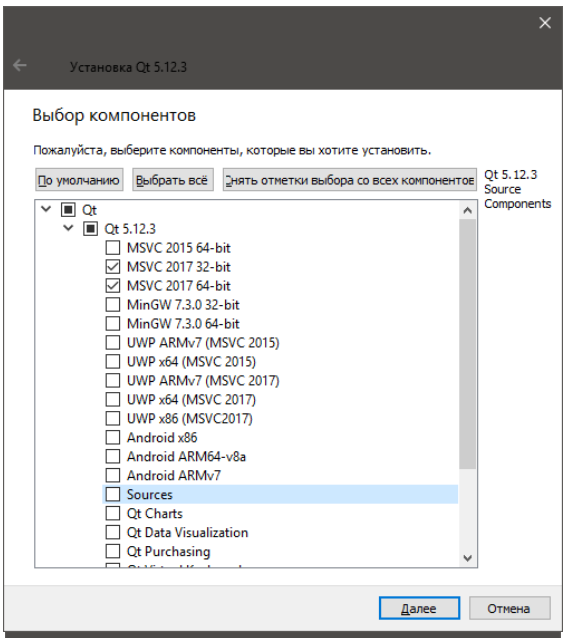
Далее – важный пункт, выбор пути остановки.

Ставьте в корень диска! На новых дистрибутивах не проверял, но после установки дистрибутивов версий до 5.0 в длинные пути, в пути с пробелами, в пути с русскими буквами - были проблемы.
У меня диск D отведён под программирование и библиотеки, я ставлю по пути: «D:\Qt\Qt5.12.3»

Нижнюю галочку рекомендую снять, иначе он расширения файлов перепишет в Visual Studio на Qt Creator (среда разработки, идущая в составе с Qt).



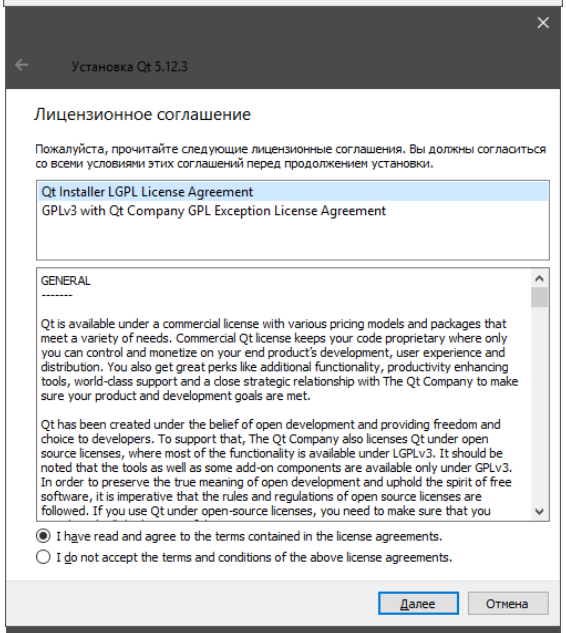
Дальше у нас спрашивают, какие компоненты мы будем ставить. Второй блок (нет на фото, ниже, называется Development and Designer Tools) оставляем без изменения, если планируете использовать с Qt среду разработки Visual Studio.
Если хотите использовать Qt Creator, рекомендую выбрать какой-нибудь компилятор (в самом Qt Creator его нет).



В этой версии на выбор:
MinGW 7.3.0 32-bit и MinGW 7.3.0 64-bit.

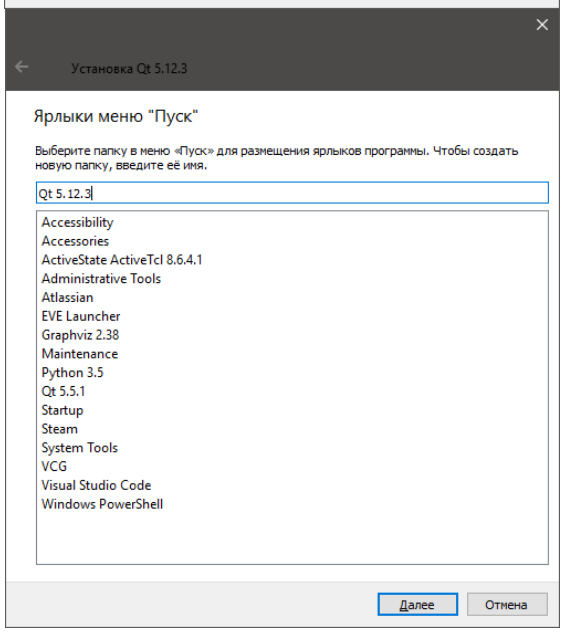
В верхен блоке я выбрал скомпилированные компоненты для Visual Studio 2017 (32 и 64 бит).

Стандартный диалог с лицензионным соглашением.



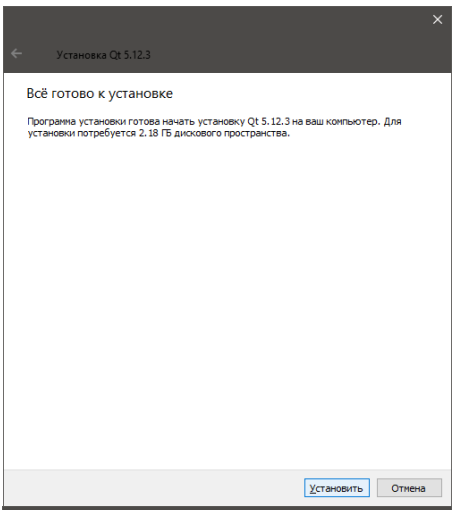
Сами знаете, что нужно делать.

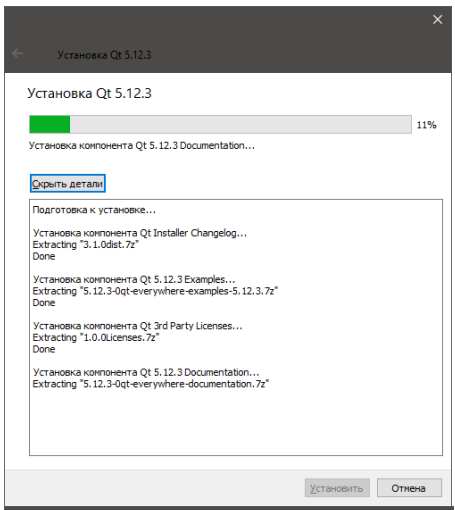
Пункт меню, можно оставить, как есть.



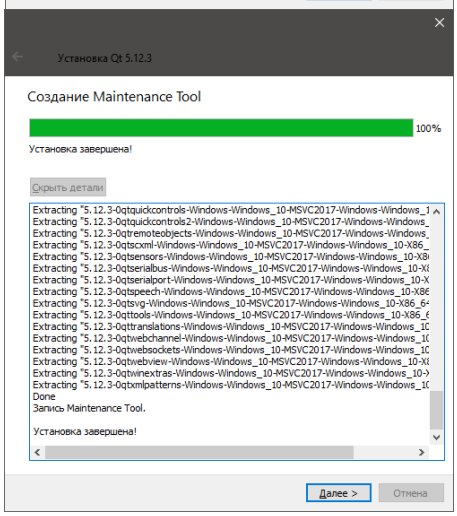
Ставим.

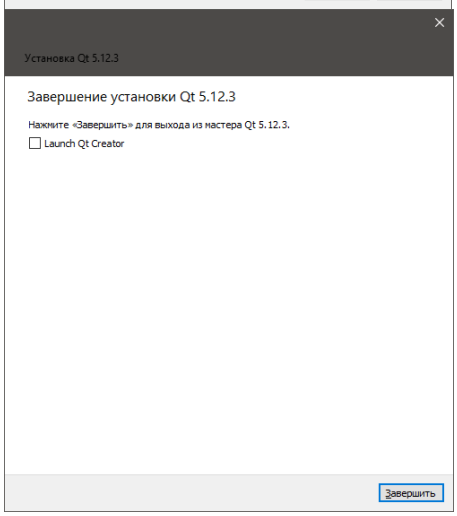
У меня установка заняла почти 15 минут.





Говорим далее и снимаем галочку с опции «Launch Qt Creator»



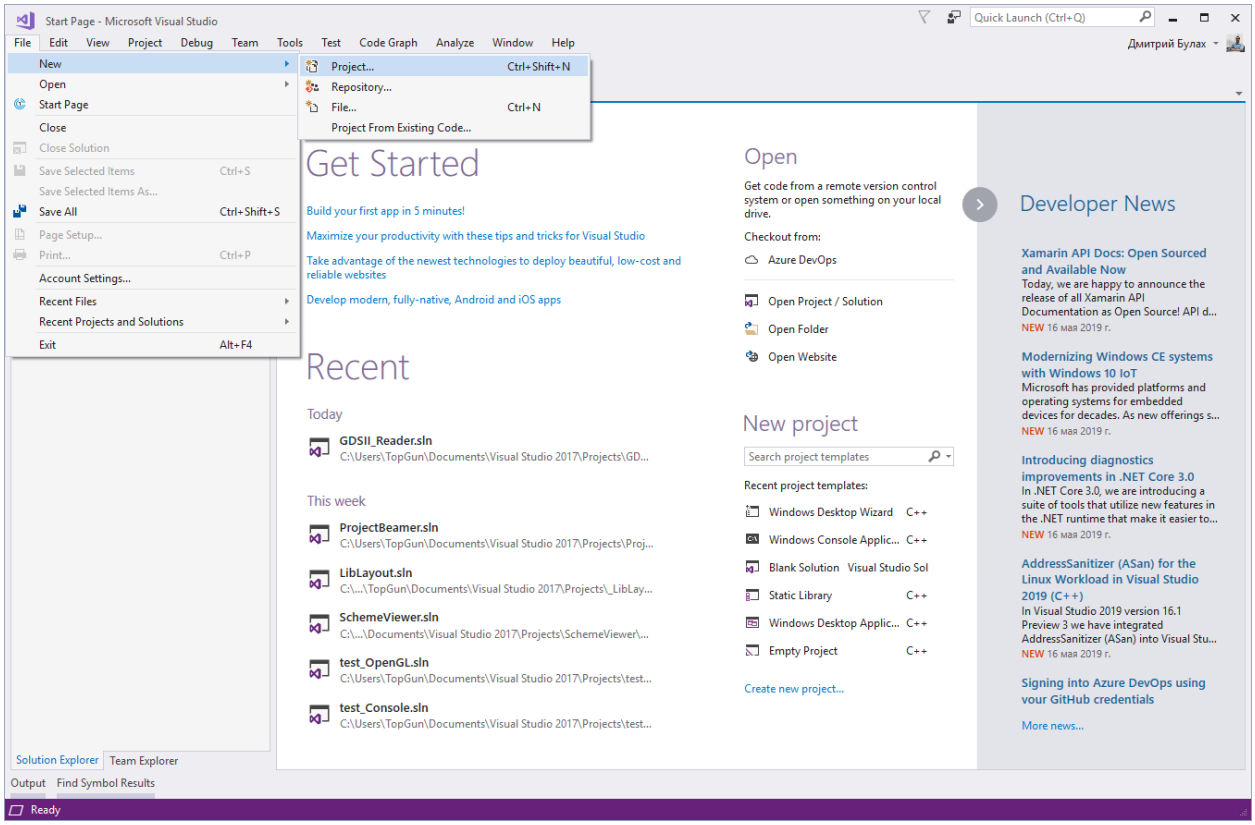


Всё, установка завершена.

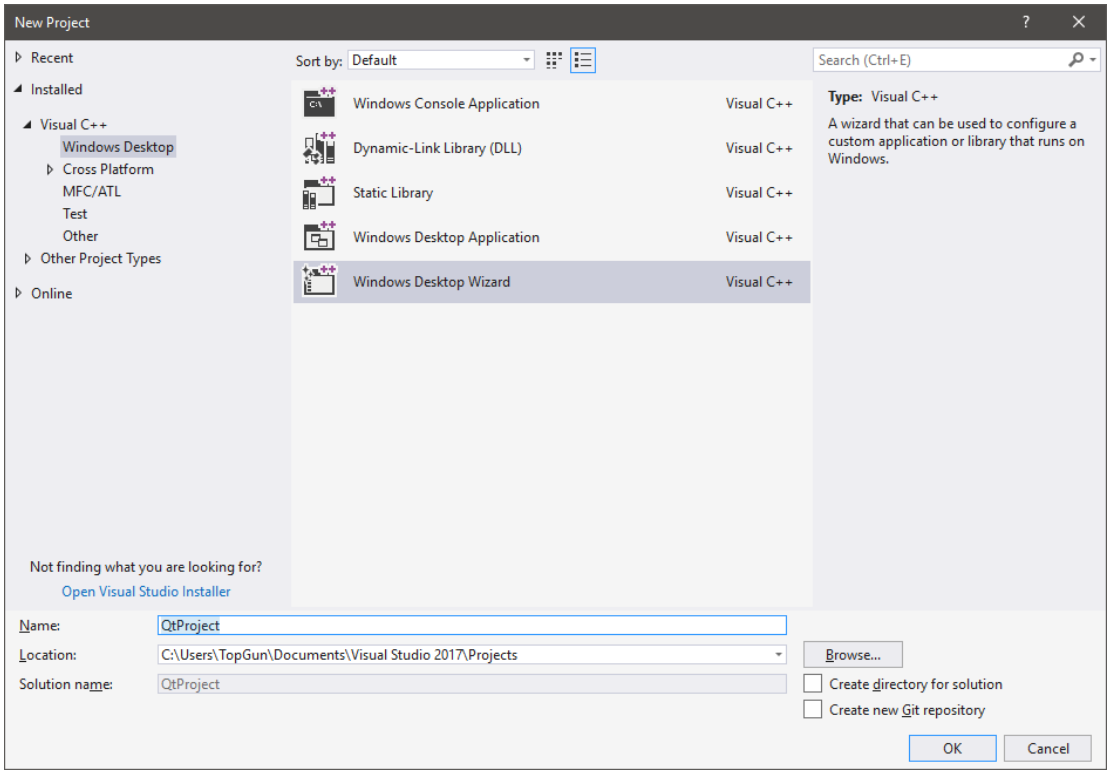
Установленная библиотека заняла 6Гб (с учётом того, что я выбрал две архитектуры: 32-bit и 64-bit)

Создание и настройка проекта в MS Visual Studio 2017.

Запускаем Visual Studio 2017, создаём новый проект («File» → «New» → «Project»)



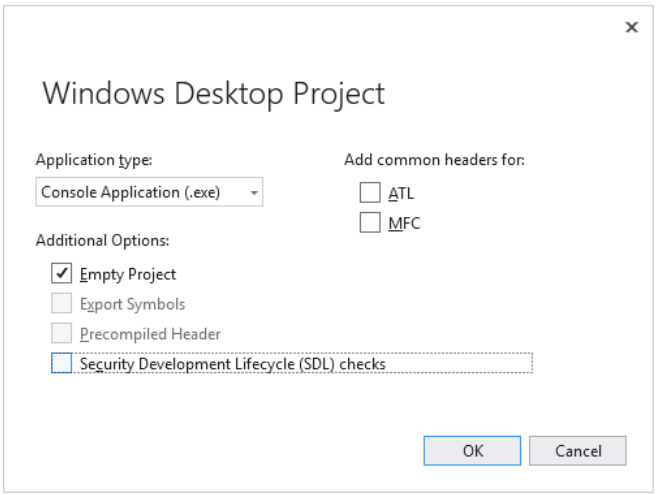
Откроется следующий диалог создания приложения:



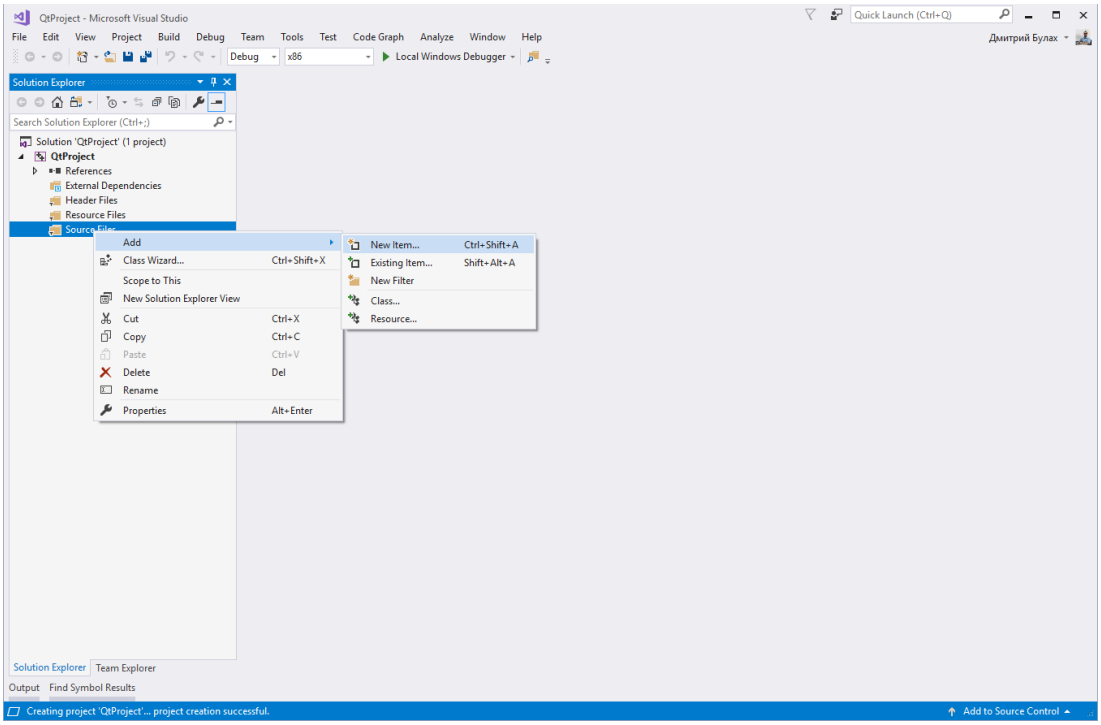
Можно пойти двумя путями.

1. Создать консольное приложение (верхний пункт), тогда Visual Studiuo сгенерирует каркас приложения с некоторым набором файлов. Ничего плохого в этом нет, но я не люблю такой вариант, так как я не люблю лишние файлы, мне приходится вычищать проект.
2. Выбрать Windows Desktop Wizard (как на рисунке выше).

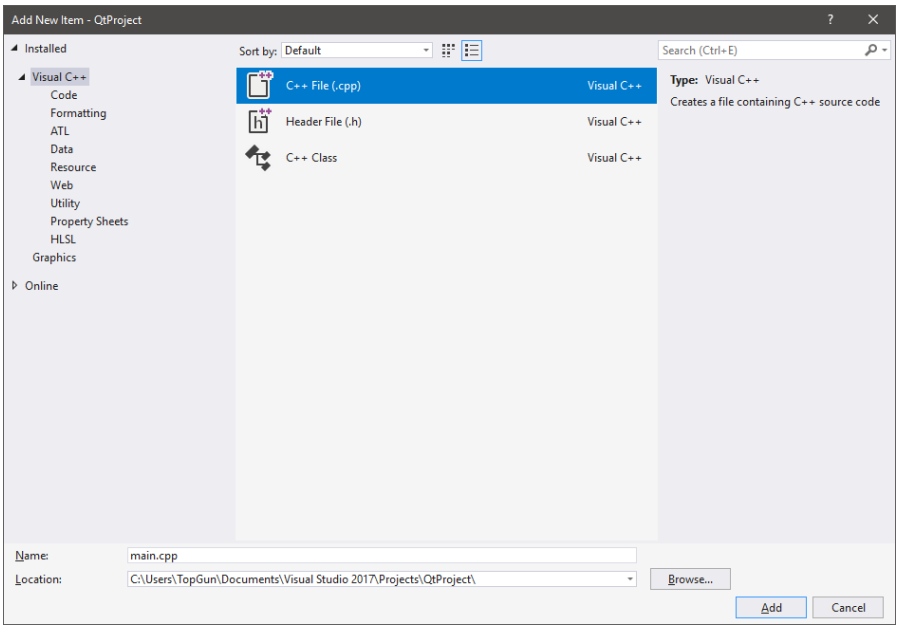
Откроется диалог настройки создаваемого проекта. Я выбираю «Пустрой проект», все остальные галочки снимаю. Убедитесь, что вверху, в выбрасывающемся списке, выбрано «Console Application (.exe)»



В созданный проект добавляем новый файл исходного кода. В дереве проекта правой кнопкой мышки на папке «Source Files» → «Add» → «New item…»



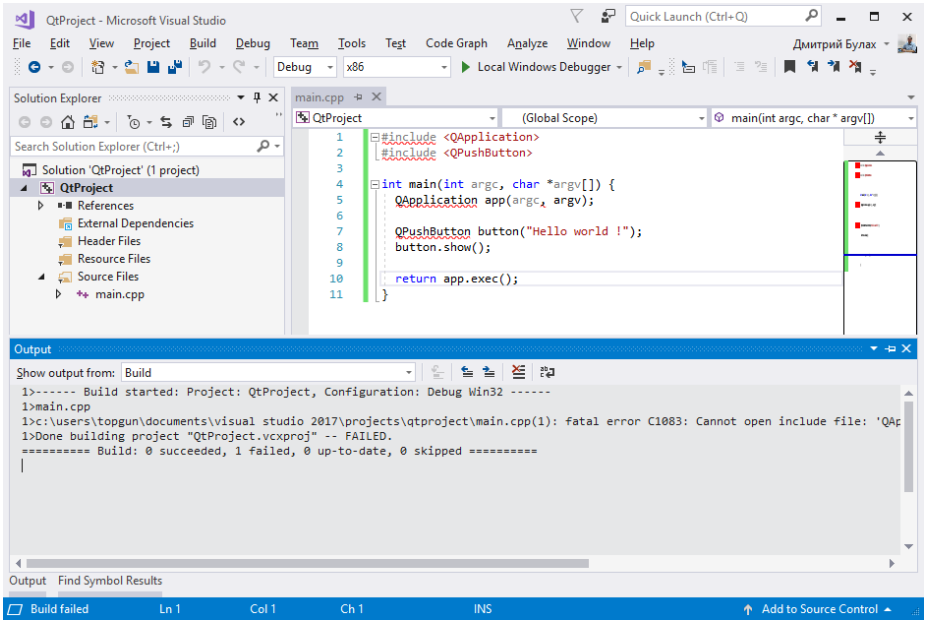
В диалоге выбираем «C++ File (.cpp)», внизу пишем имя файла.



В созданном файле пишем следующий код:

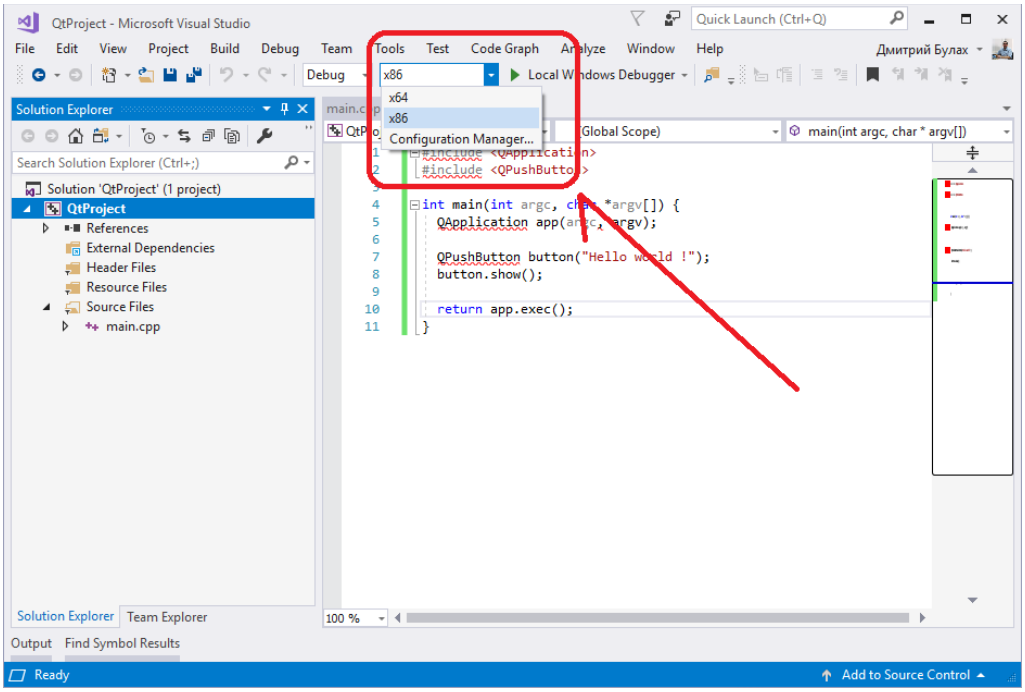
```
#include <QApplication>
#include <QPushButton>

int main(int argc, char *argv[]) {
  QApplication app(argc, argv);

  QPushButton button("Hello world !");
  button.show();

  return app.exec();
}
```

При попытке его скомпилировать, Visual Studio 2017 ругается всеми возможными способами:

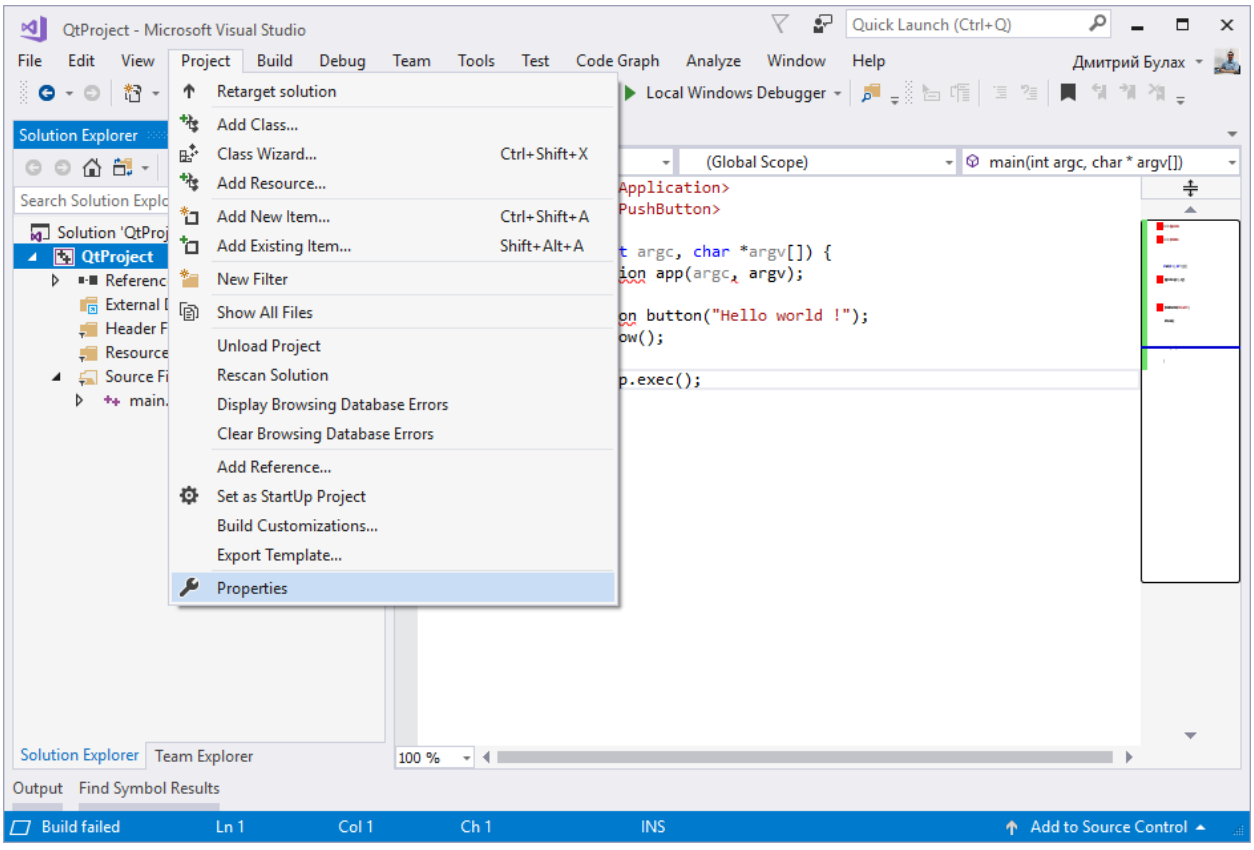


Это вполне логично. Мы поставили Qt, но не объяснили Visual Studio, где брать заголовочные файлы и библиотеки Qt.

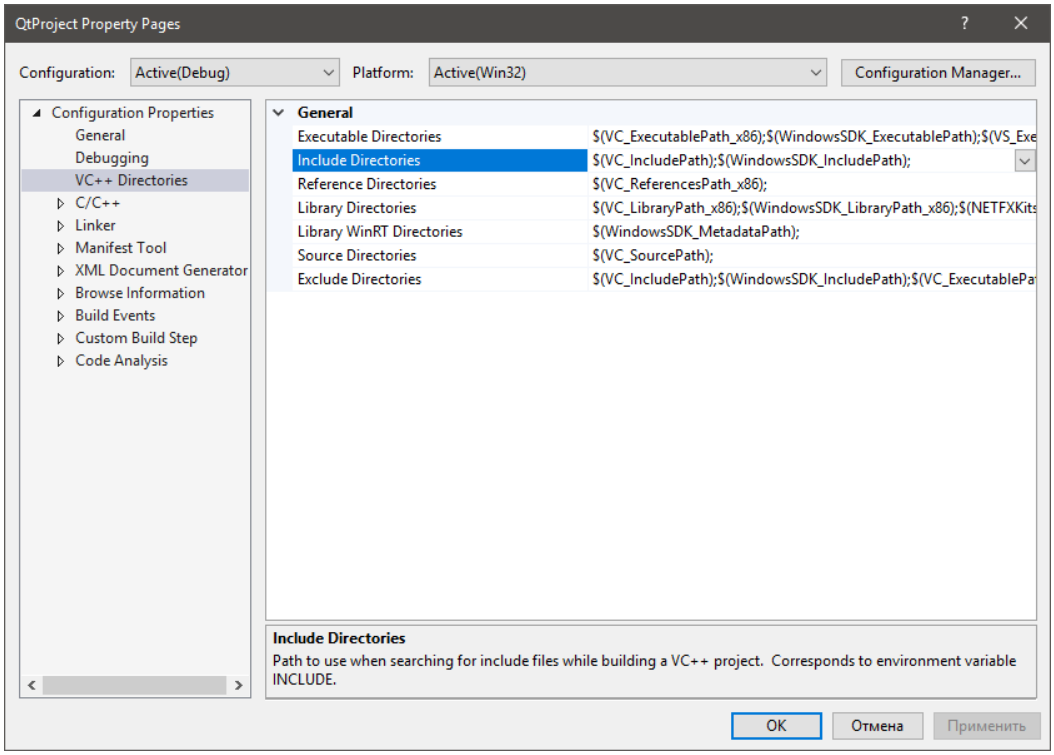
Прежде, чем это сделать, необходимо определить архитектуру, для которой компилируется проект: x86 или x64. Сделать это можно в панели инструментов.



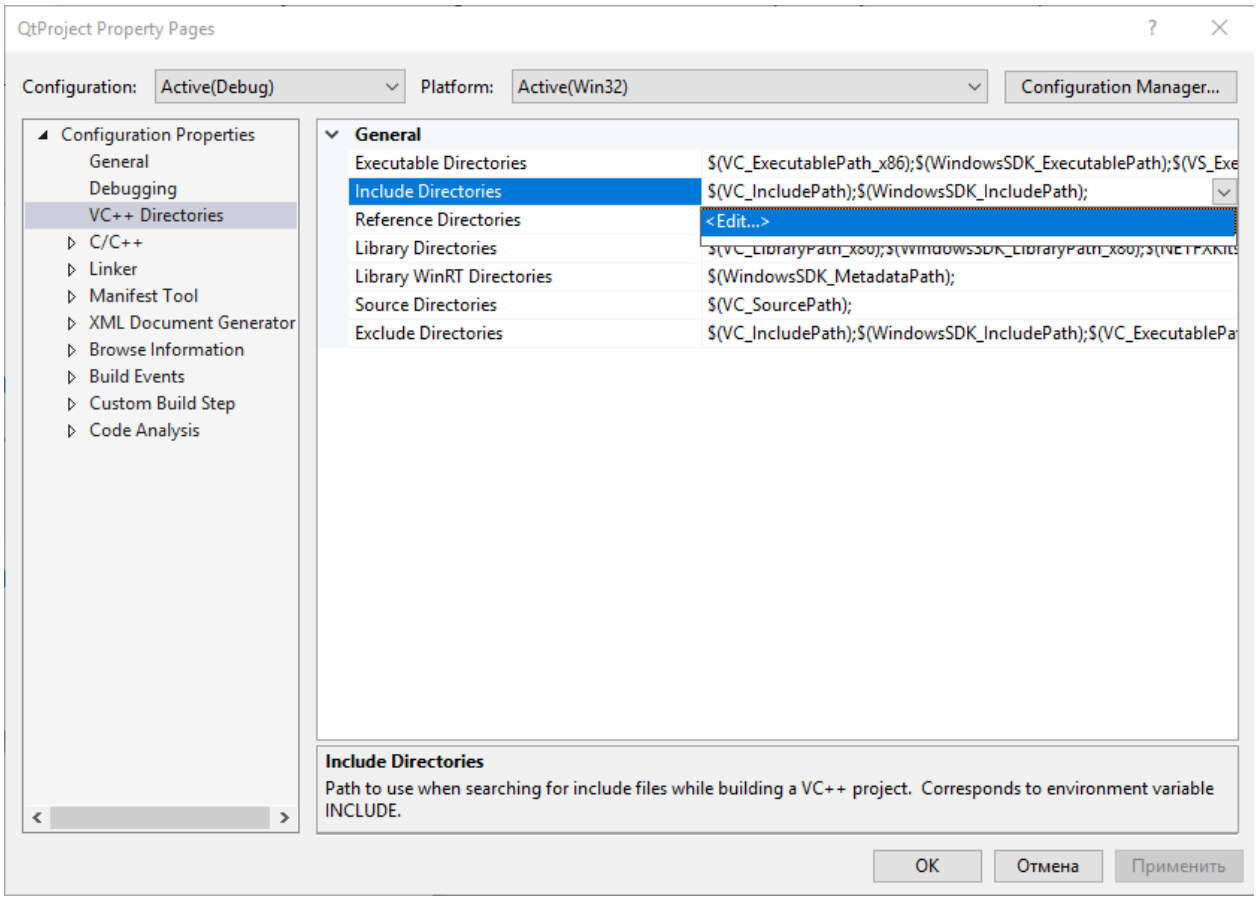
Теперь, когда мы знаем платформу, нужно настроить пути. Чтобы это сделать, необходимо в меню выбрать пункт «Project» → «Properties»



В открывшемся диалоге нужно выбрать слева пункт «VC++ Directories». В этом разделе нас будут интересовать два пункта: «Include Directories» и «Library Directories».



Редактируем пути к заголовочным файлам (Include Directories):



В это поле нужно занести следующие пути (помним, что я поставил Qt по пути D:\Qt\Qt5.12.3, если вы поставили в другое место, прописывайте свои пути).

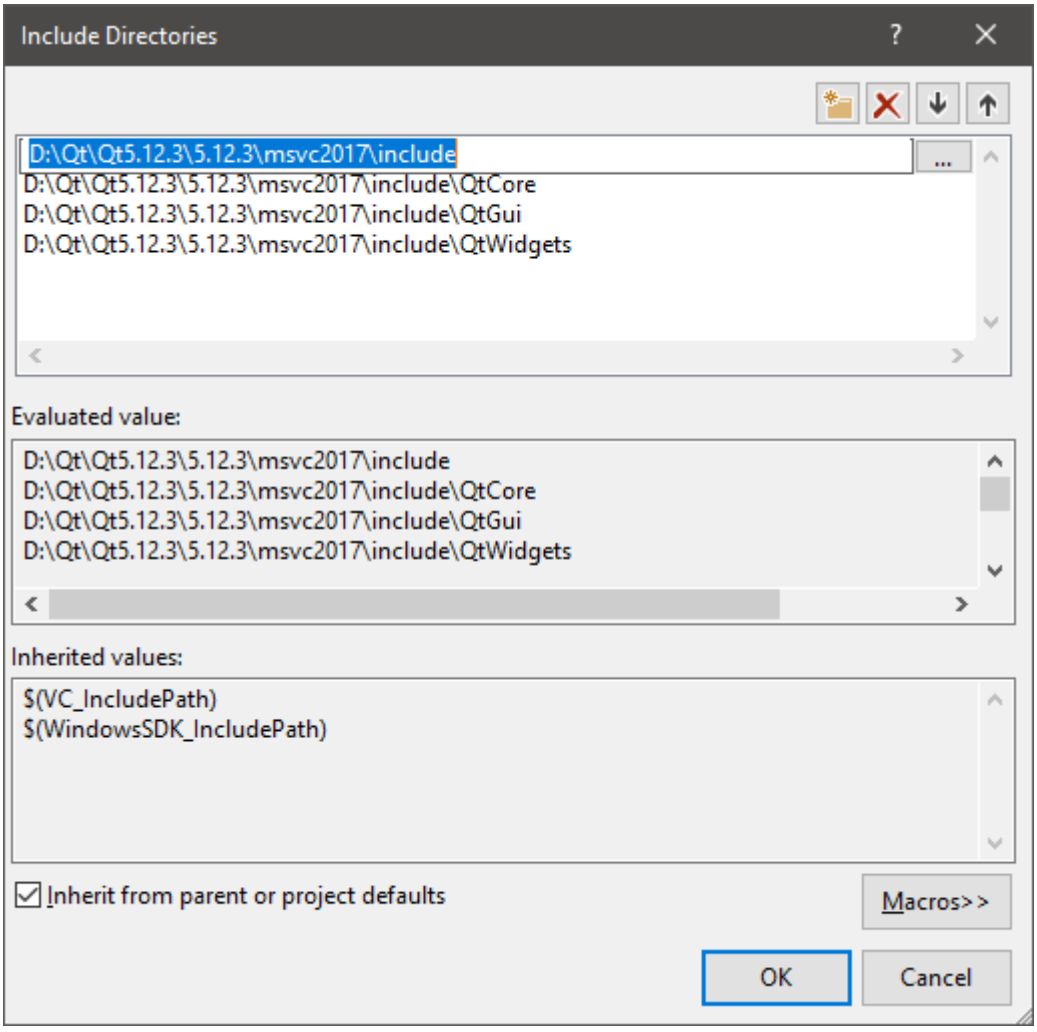
Для архитектуры x86 (порядок перечисления папок не важен):

- D:\Qt\Qt5.12.3\5.12.3\msvc2017\include
- D:\Qt\Qt5.12.3\5.12.3\msvc2017\include\QtCore
- D:\Qt\Qt5.12.3\5.12.3\msvc2017\include\QtGui
- D:\Qt\Qt5.12.3\5.12.3\msvc2017\include\QtWidgets

Для архитектуры x64 (порядок перечисления папок не важен):

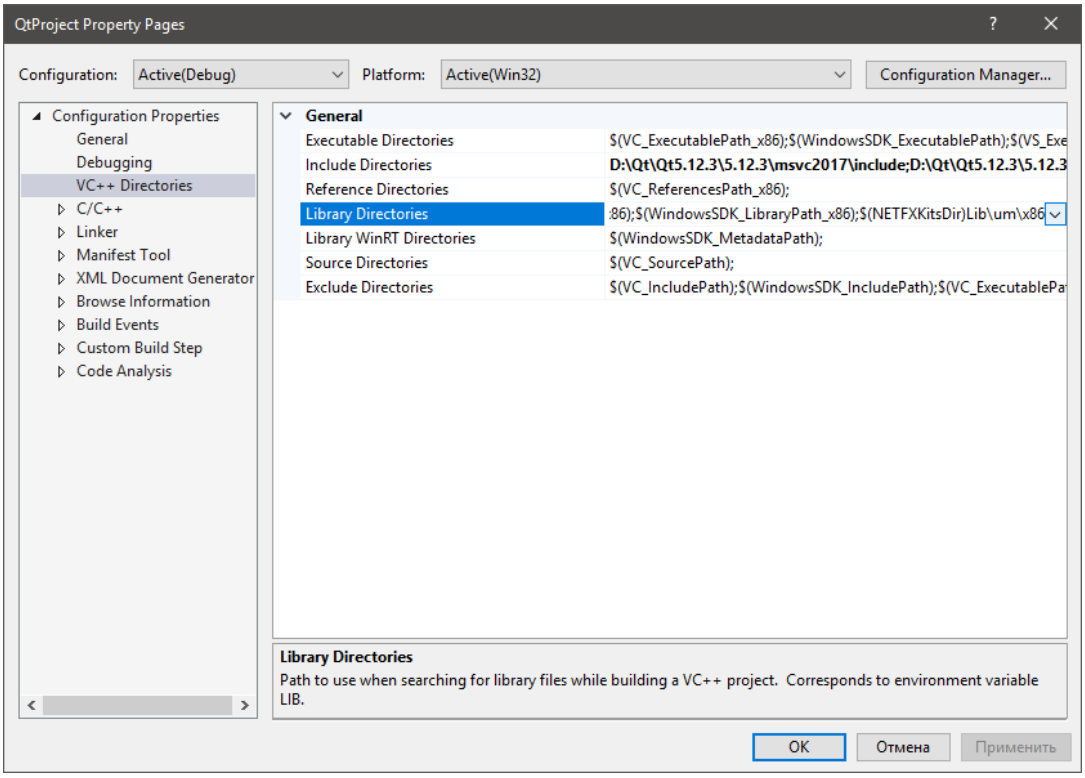
- D:\Qt\Qt5.12.3\5.12.3\msvc2017_64\include
- D:\Qt\Qt5.12.3\5.12.3\msvc2017_64\include\QtCore
- D:\Qt\Qt5.12.3\5.12.3\msvc2017_64\include\QtGui
- D:\Qt\Qt5.12.3\5.12.3\msvc2017_64\include\QtWidgets

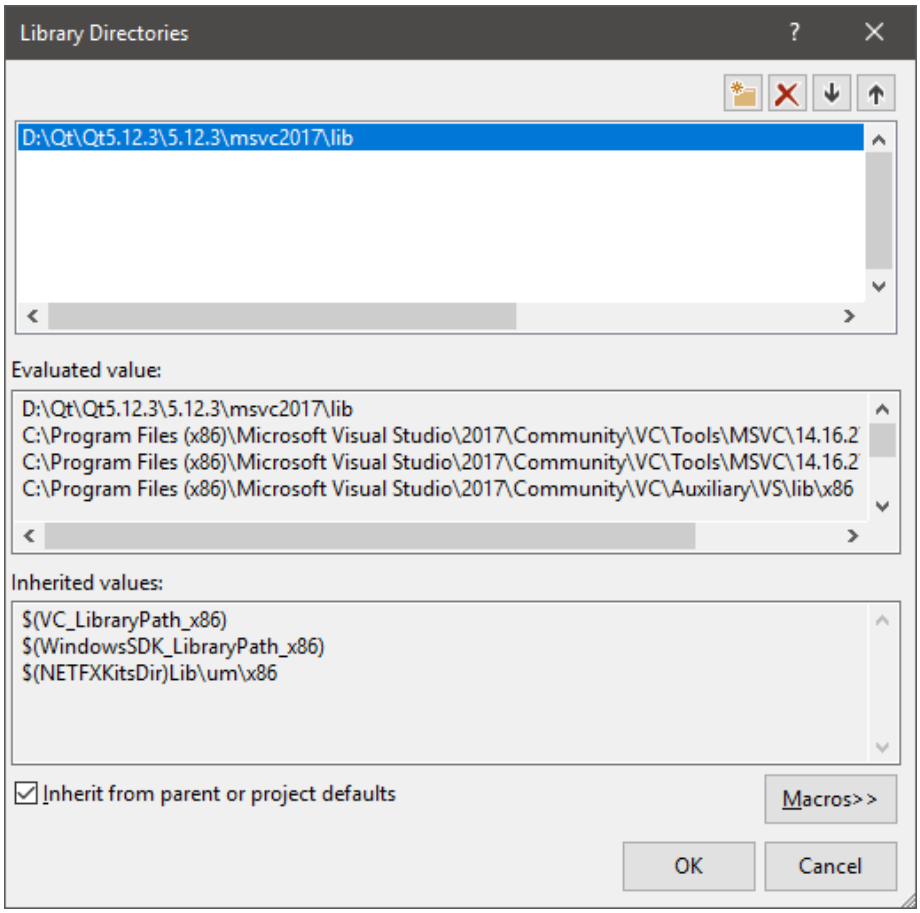
Пример, как должно выглядеть окно настроек путей для 32-х битной архитектуры:



Закрываем диалог.

Аналогичным образом приписываем пути к библиотекам (Library Directories), которые мы должны подключить к проекту (с учётом разрядности пути снова меняются, я привожу пример для моего варианта, архитектуры x86).

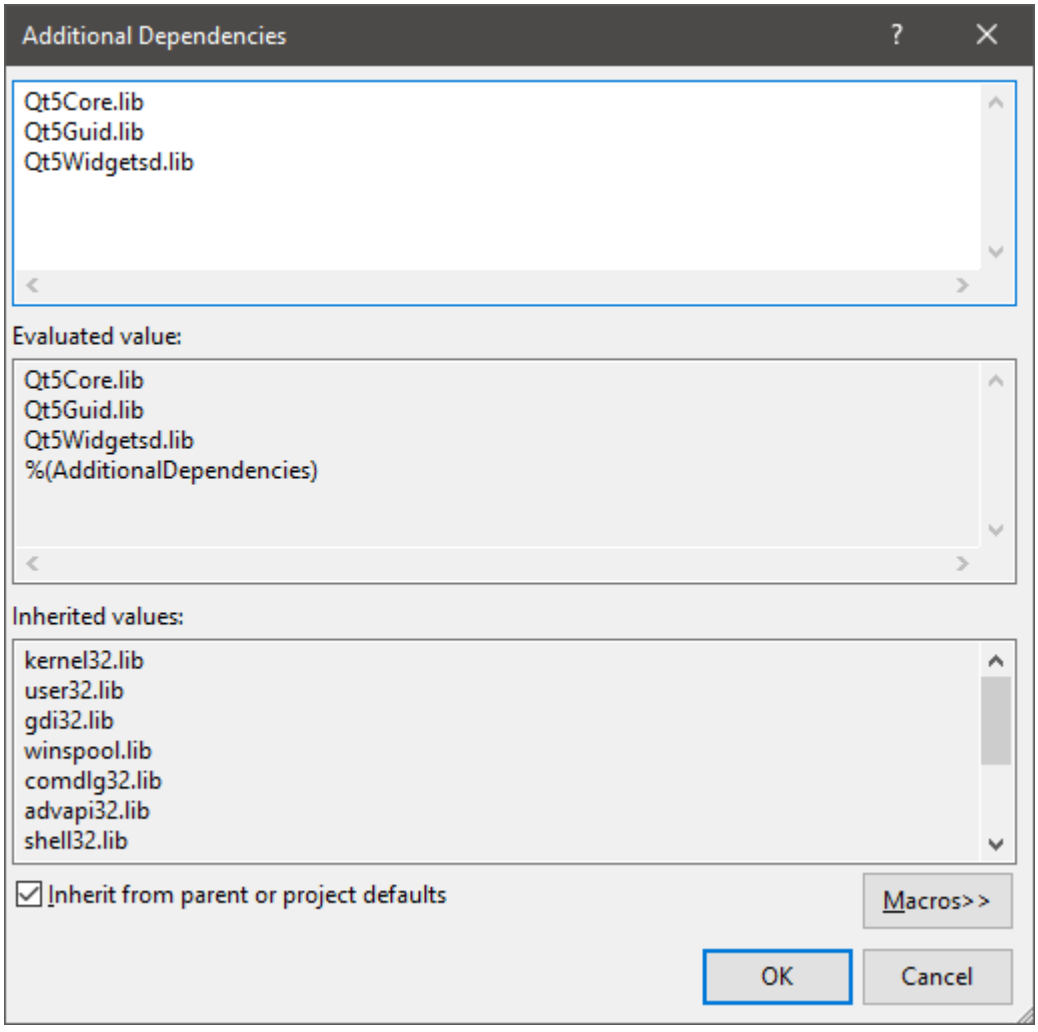




Теперь необходимо объяснить, какие именно библиотеки из пути, указанного в настройках «Library Directories» мы должны подключить к нашему проекту. Перечень подключаемых библиотек сильно зависит от того, какие возможности Qt мы будем использовать. Наш пример – самый простой, нам нужно всего три библиотеки:

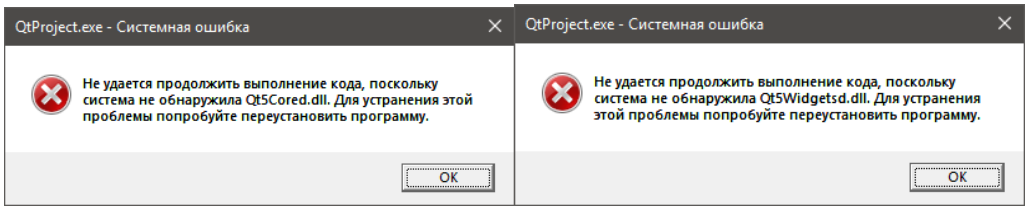
1. Qt5Cored.lib
2. Qt5Guid.lib
3. Qt5Widgetsd.lib

Для этого в свойствах проекта (меню «Propject» → «Properties» в главном окне Visual Studio) выбираем слева раздел «Linker», подраздел «Input». Справа в секции «Additional Dependencies» указываем дополнительные зависимости – имена библиотек.



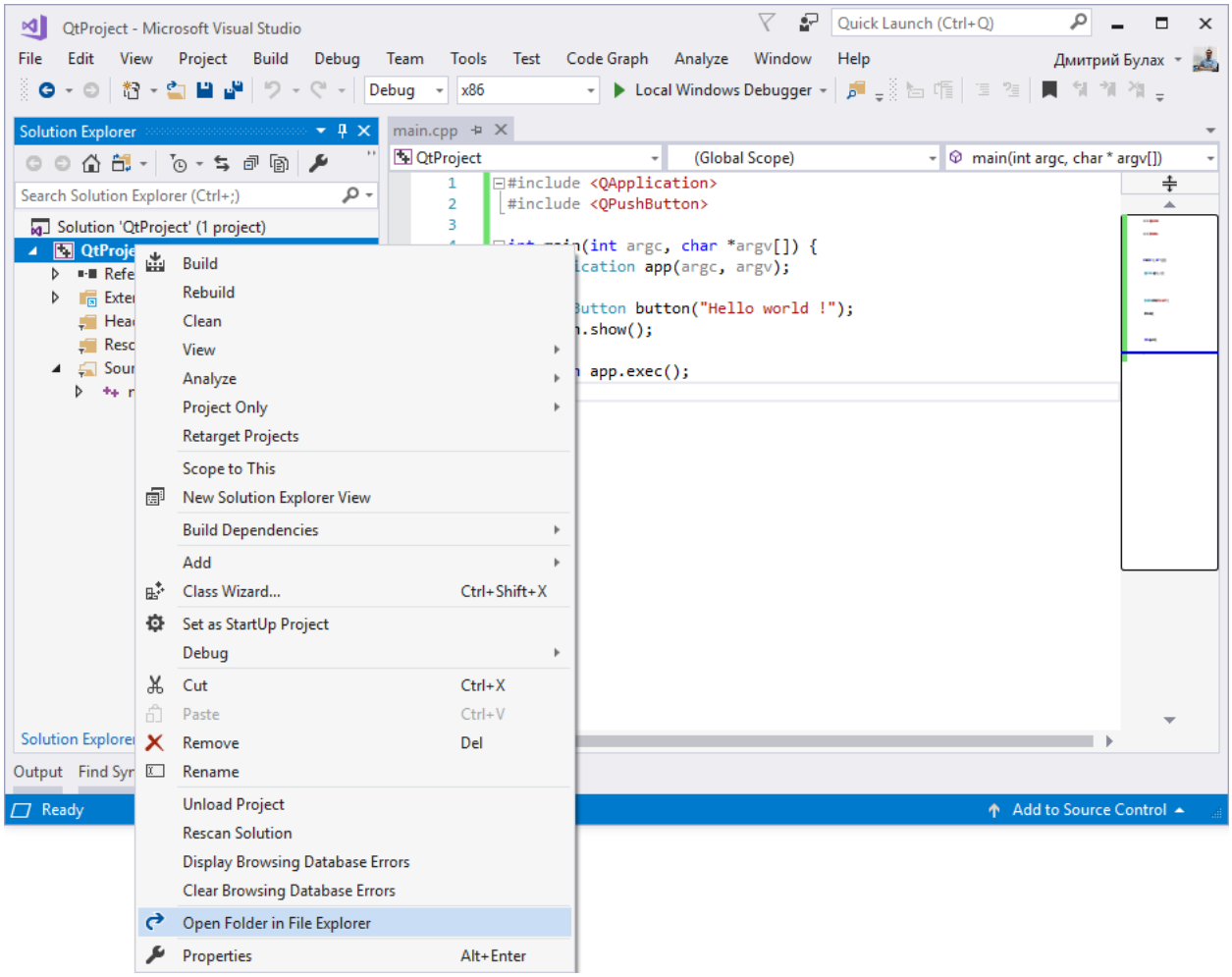
Всё, можно компилировать и собирать проект, компиляция должны пройти без проблем.

Последнее, что нужно сделать, избавиться вот от этих ошибок:



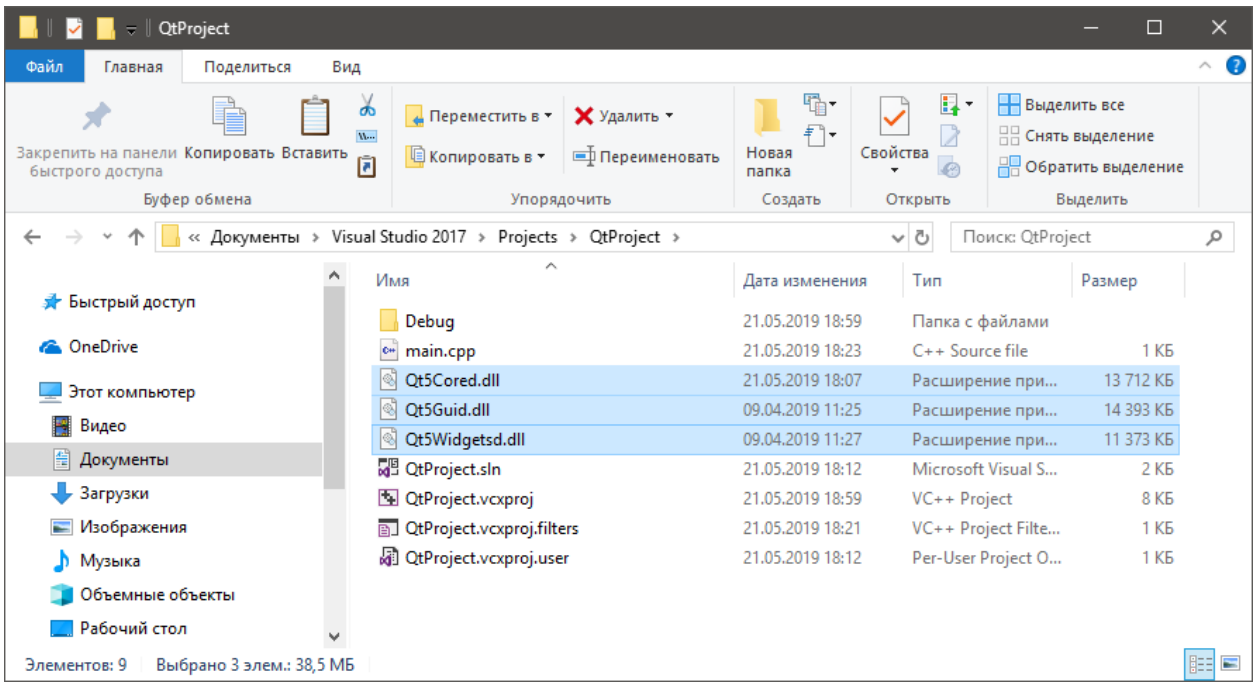
Для работы нашему приложению нужны динамические библиотеки Qt. Их нужно переписать из папки Qt «D:\Qt\Qt5.12.3\5.12.3\msvc2017\bin» для архитектуры x86 и «D:\Qt\Qt5.12.3\5.12.3\msvc2017_64\bin» для архитектуры x64.

Папку проекта, куда нужно переписать, можно открыть очень просто.
В дереве проекта на имени проекта (выделен жирным) правой кнопкой мыши открыть контекстное меню и выбрать пункт «Open Folder in File Explorer».

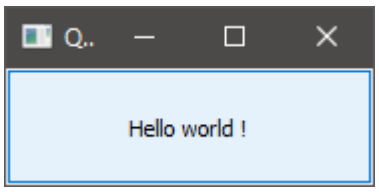


В открывшуюся папку нужно переписать требуемые файлы .dll из папки с бинарниками Qt.

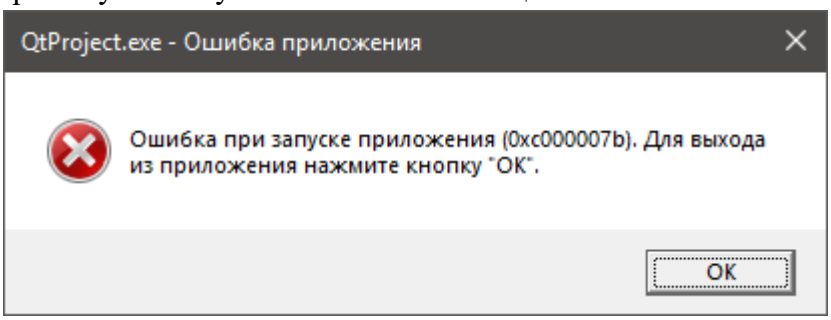
Результат показан на следующим скрине.



Теперь можно запускать, результат показан на рисунке ниже.



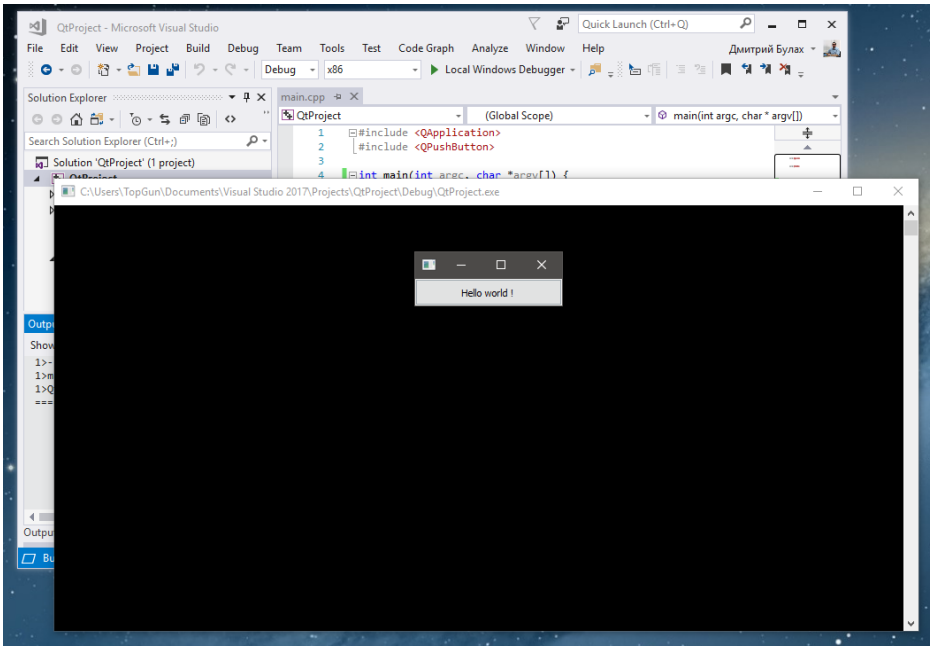
Если вдруг вы при запуске получили вот такое сообщение:



то, с большой долей вероятности, вы перепутали архитектуры переписанных .dll и самого приложения. Например, я такое сообщение получил, скопировав в папку с приложением, скомпилированном под архитектуру x86, файлы из папки для версии Qt для архитектуры x64.

Запуск проекта под Visual Studio без консоли

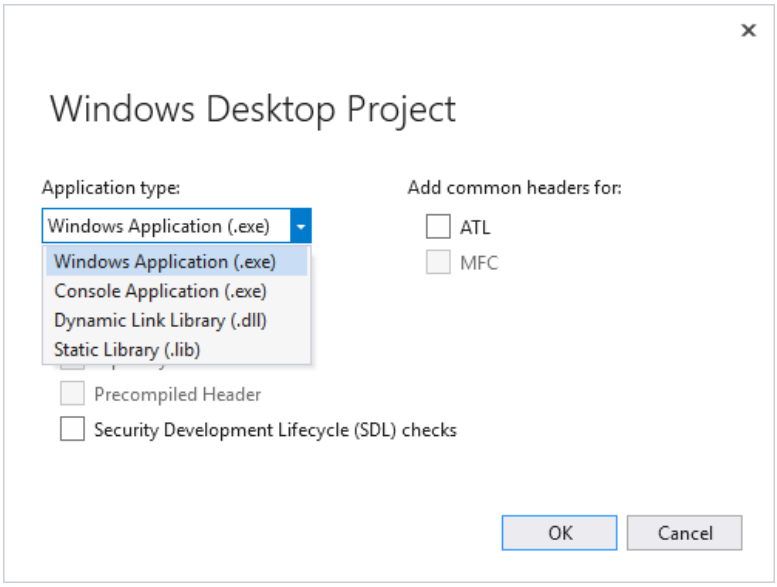
Вы могли заметить, что при запуске приложения также открывается и консоль.



Если есть желание это исправить (сделать чисто виндовое приложение), следует внести в проект следующие изменения.

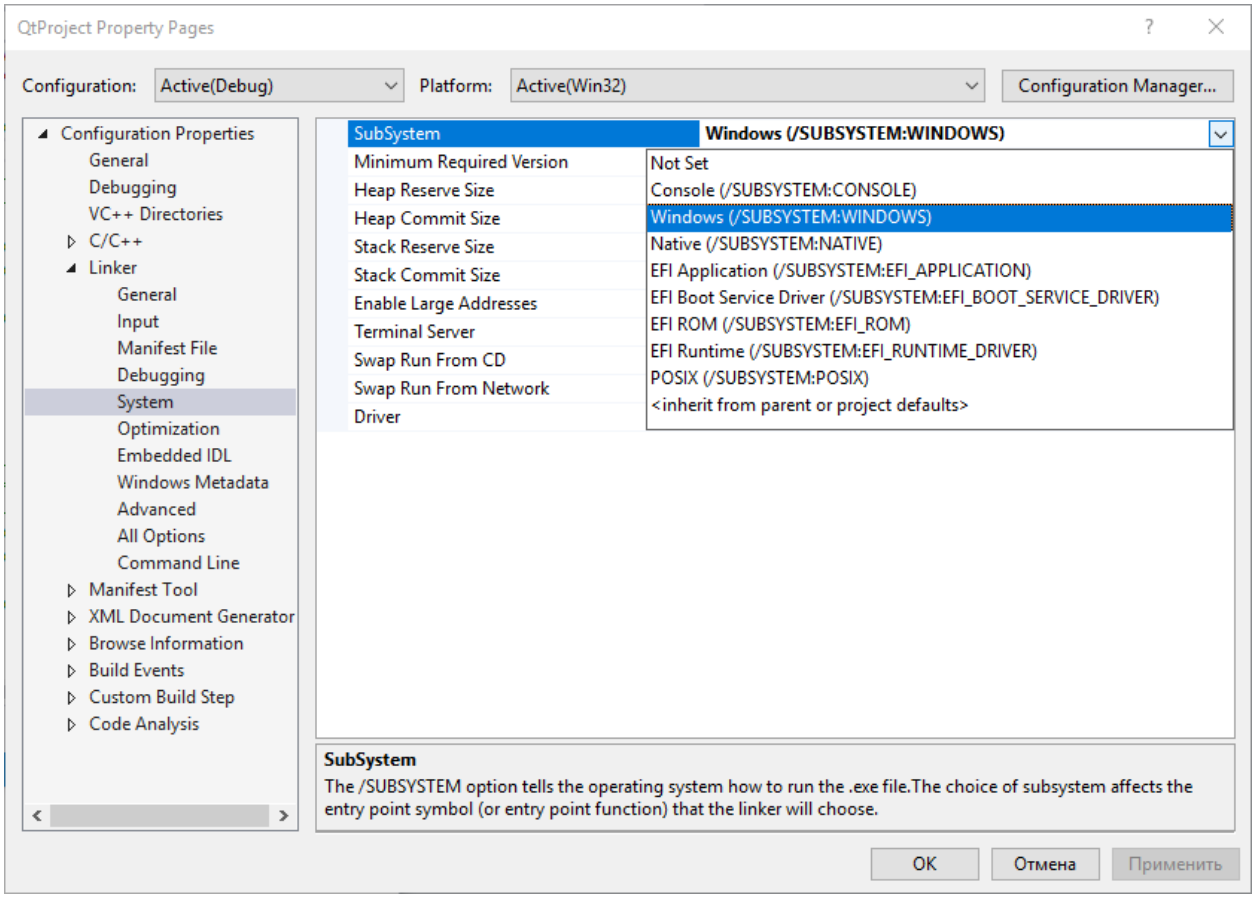
Первое. Необходимо объяснить линковщику, что нужно собирать не консольное приложение, а приложение Windows. Это можно сделать двумя способами.

1. Изначально при создании проекта выбрать не консольное приложение, а приложение Windows (см. прилагаемый скриншот ниже, тогда пункт 2 ниже не нужен)



2. Настроить существующий консольный проект так, чтобы он стал приложением Windows. Для этого необходимо для лонковщика сменить информацию о типе проекта, заменив в настройках (пункт меню главного окна «Project» → «Properties») для раздела слева «Linker» → «System», справа подраздел

«SubSystem» значение «Console (/SUBSYSTEM:CONSOLE)» на «Windows (/SUBSYSTEM:WINDOWS)». Диалог показан на рисунке ниже.



Какой бы вариант вы не выбрали, теперь в коде нужно заменить точку входа с функции main на функцию WinMain. Код должен выглядеть следующим образом:

```
int __stdcall WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, char*, int nShowCmd) {
  int argc = 0;
  QApplication app(argc, nullptr);
  QPushButton button("Hello world !");
  button.show();

  return app.exec();
}
```

Весь остальной код любого более сложного проекта, включая любые другие файлы, остаётся неизменным. Изменения касаются только точки входа в программу.

Установка Qt 5.12.3 под Linux

Установка Qt5.12.3 под MacOS