Связка Windows+Eclipse+QT+MinGW+CMake:

1. Скачиваем QT, сейчас актуальна версия **Qt 5.3.1 for Windows 32-bit (MinGW 4.8.2, OpenGL, 735 MB)**
2. Скачиваем CMake, сейчас актуальная версия cmake-3.0.0-win32-x86.exe (генерирует файлы управления сборкой из файлов `CMakeLists.txt`)
3. Прописываем пути в PATH до cmake (можно задать при установке), до mingw (идет вместе с Qt, сейчас ставится в C:\Qt\Qt5.3.1\Tools\mingw482_32\bin), до qmake (C:\Qt\Qt5.3.1\5.3\mingw482_32\bin),
4. Проверяем из консоли, что пути выставлены правильно:

>cmake --version
cmake version 3.0.0

CMake suite maintained and supported by Kitware (kitware.com/cmake).

>g++ --version
g++ (i686-posix-dwarf-rev3, Built by MinGW-W64 project) 4.8.2
Copyright (C) 2013 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

>qmake --version
QMake version 3.0
Using Qt version 5.3.1 in C:\Qt\Qt5.3.1\\5.3\mingw482_32\lib

5. Добавляем переменную окружения QTDIR=C:\Qt\Qt5.3.1\5.3\mingw482_32 (понадобится потом в Eclipse),
6. Создаём тестовый файл **CMakeLists.txt** в обычном **блокноте** и сохраним его в папке C:\HelloWorld :

```
cmake_minimum_required (VERSION 2.8.8)

cmake_policy (SET CMP0020 NEW)

SET (CMAKE_AUTOMOC ON)

SET (CMAKE_BINARY_DIR bin)

SET (CMAKE_INCLUDE_CURRENT_DIR ON)

FIND_PACKAGE (Qt5Widgets REQUIRED)

SET (PROJECT           "HelloWorld")

FILE (GLOB HEADERS     "${PROJECT}/*.h")

FILE (GLOB SOURCES     "${PROJECT}/*.cpp")

FILE (GLOB FORMS "${PROJECT}/*.ui")

FILE (GLOB RESOURCES"${PROJECT}/*.rc")

QT5_WRAP_UI (UI_HEADERS ${FORMS})

ADD_EXECUTABLE (${PROJECT} WIN32 ${HEADERS} ${SOURCES} ${UI_HEADERS} ${RESOURCES})

QT5_USE_MODULES (${PROJECT} Widgets)
```

7. **mainwindow.h**:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

Сохраняем файл **mainwindow.h** в C:\HelloWorld\HelloWorld

8. **mainwindow.cpp:**

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}
MainWindow::~MainWindow()
{
    delete ui;
}
```

Сохраняем файл **mainwindow.cpp** в C:\HelloWorld\HelloWorld

9. **main.cpp:**

```
#include "mainwindow.h"
#include <QApplication>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Сохраняем файл **main.cpp** в C:\HelloWorld\HelloWorld

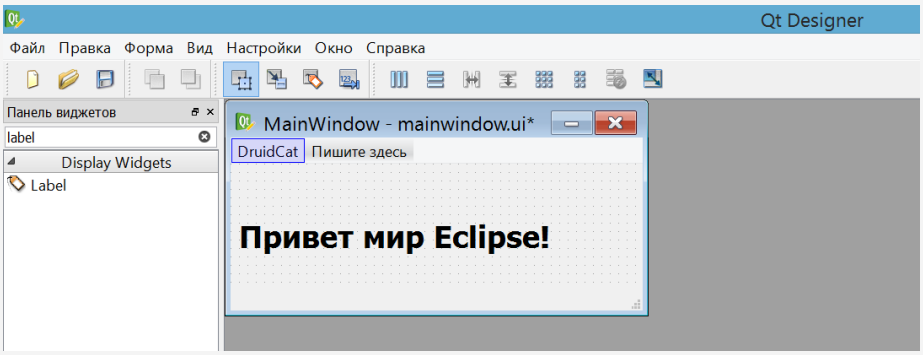
10. **mainwindow.ui**

А теперь самое интересное, запускаем QT Designer. Находится он в:

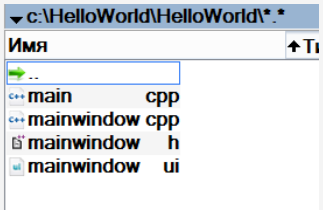
C:\Qt\Qt5.3.1\5.3\mingw482_32\bin\designer.exe

Появляется окно новой формы, выберем **Main Window**.

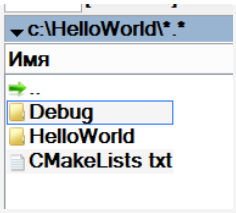
И создадим незатейливую форму, как нарисовано на картинке ниже.



И сохраняем ее под именем **mainwindow.ui** в C:\HelloWorld\HelloWorld



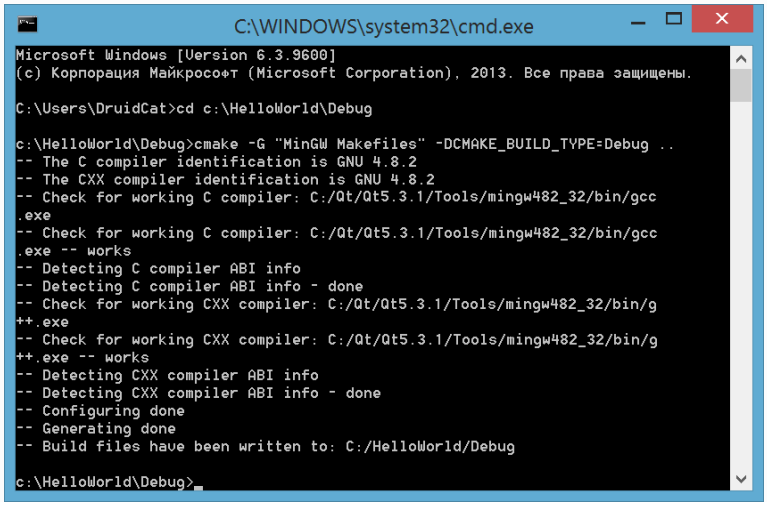
11. Создаём папку **Debug** для сборки в папке C:\HelloWorld



Заходим в папку **Debug** через терминал и запускаем cmake командой:

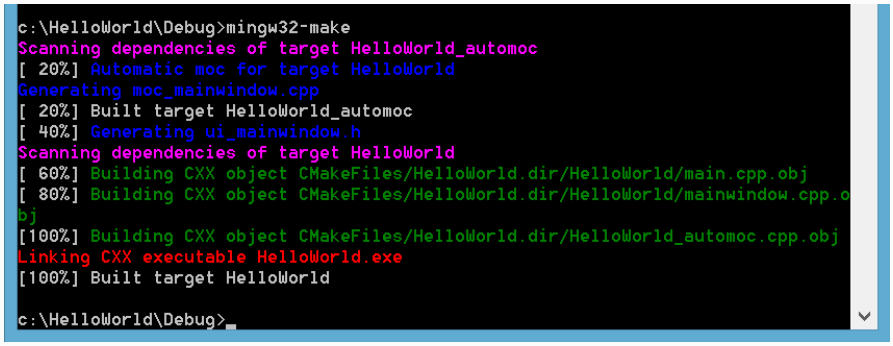
cmake -G "MinGW Makefiles" -DCMAKE_BUILD_TYPE=Debug ..

После этой команды, cmake в папке **Debug** сгенерирует Makefiles для MinGW. Хочу подметить, что название папки Debug я взял потому, что в этой папке будет создаваться отладочная версия программы, так как cmake мы даем команду на создание отладочной версии программы (**=Debug ..**). Ну и визуально очень наглядно, все выглядит как VS или QT Creator.

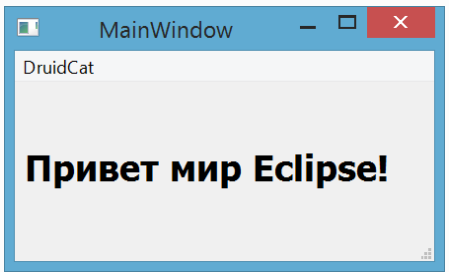


12. Запускаем сборку через mingw32-make:

mingw32-make



Запускаем **HelloWorld.exe**, чтобы проверить собранный exe.

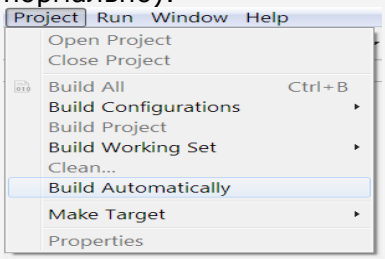


Всё это нужно сделать один раз, чтобы проверить корректность установки Qt/CMake/MinGW.

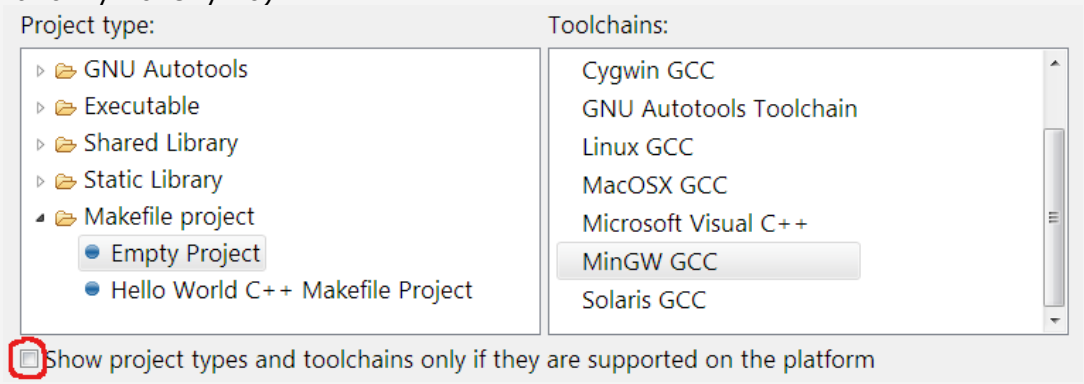
Теперь про то как будем работать в Eclipse, тестировал на Kepler и Luna:

1. Запускаем Eclipse вашем workspace.

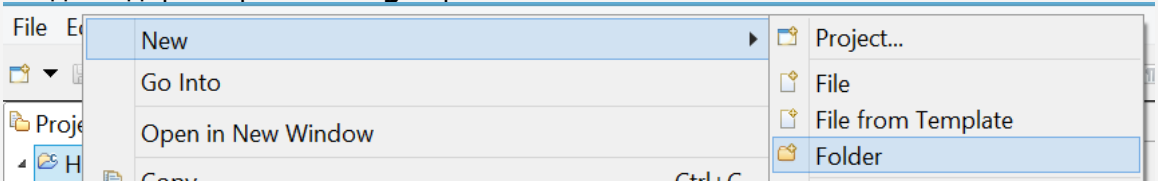
2. Сразу отключаем Project->Build Automatically (это только для Java работает нормально):



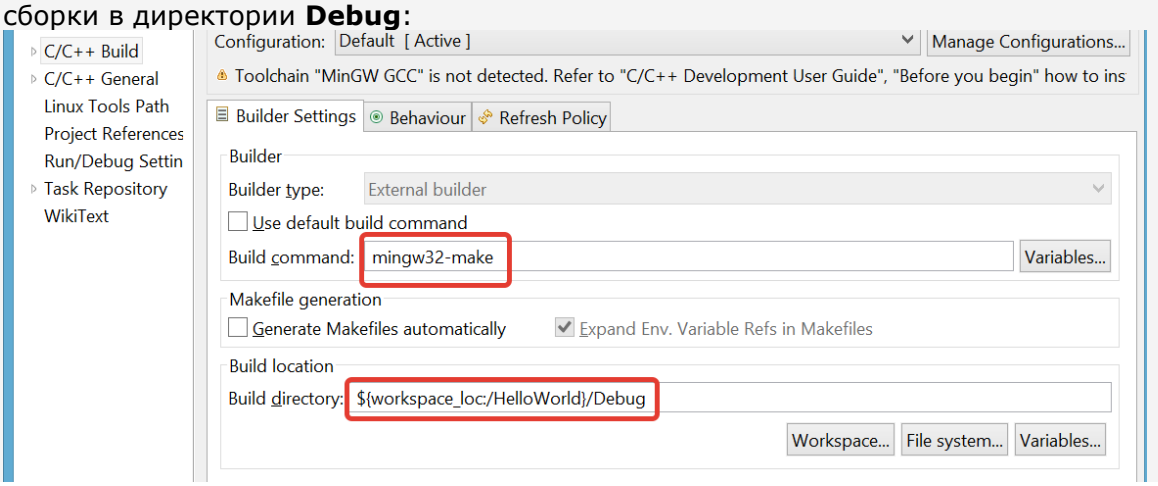
3. Создаём новый проект через **File->New->C++ Project** (приходится снимать галочку почему-то):



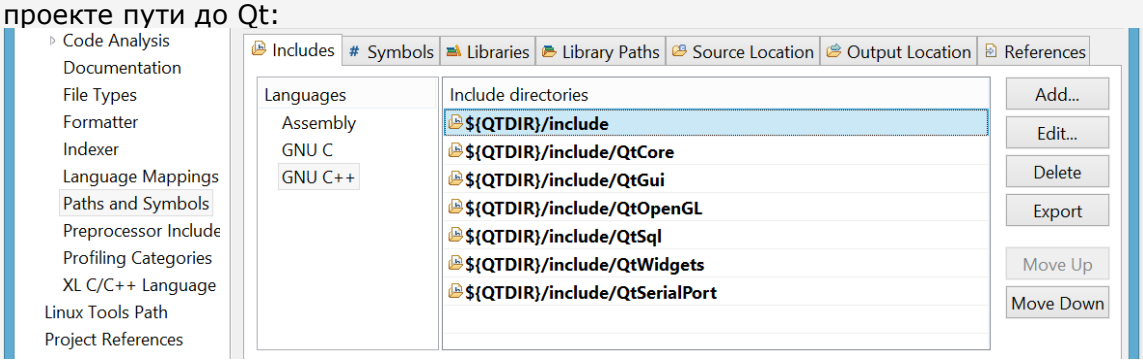
4. Создаём директорию **Debug** через контекстное меню:



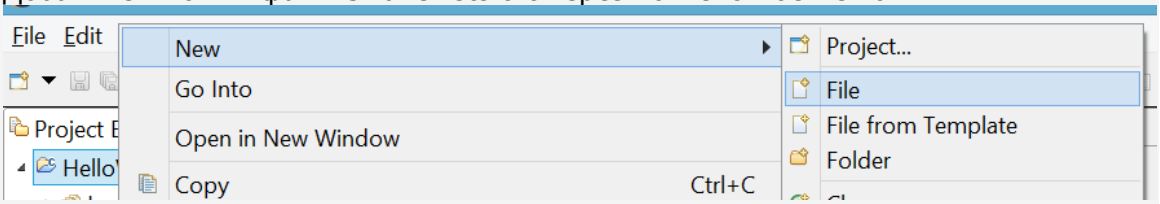
5. В настройках проекта (**C/C++ Build**) выставляем команду **mingw32-make** для сборки в директории **Debug**:



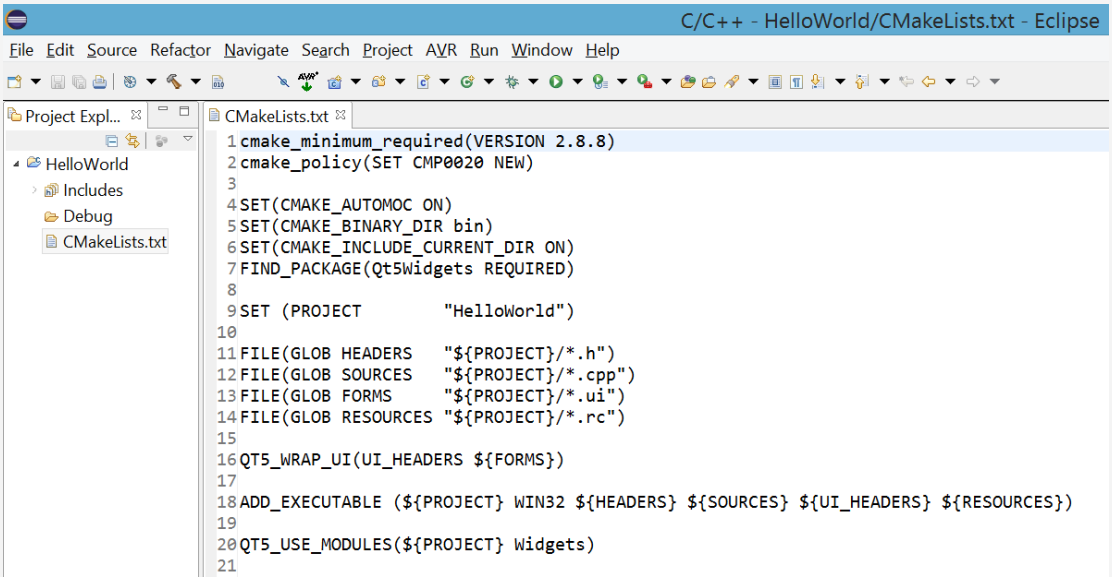
6. В настройках проекта (**Paths and Symbols**) добавляем необходимые в вашем проекте пути до Qt:



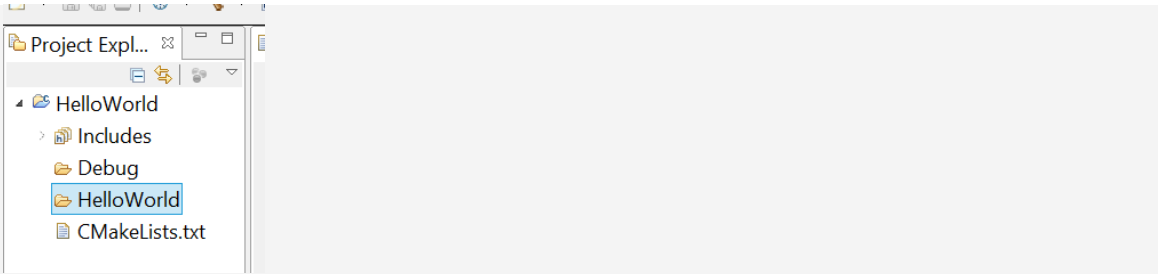
7. Добавляем новый файл CMakeLists.txt через контекстное меню:



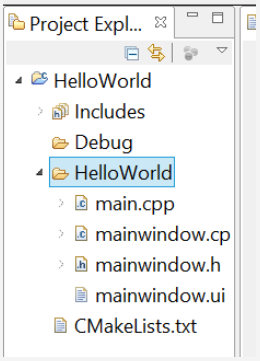
Содержимое то же, что и выше было.



8. Создаём директорию **HelloWorld** через контекстное меню:

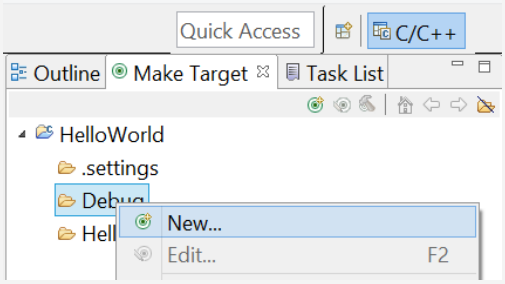


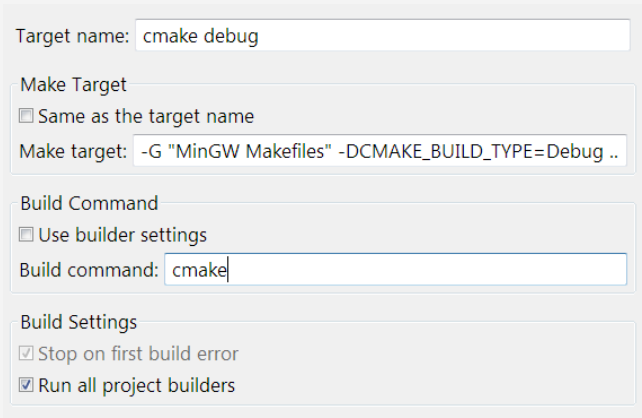
9. Добавляем в эту директорию (**HelloWorld**) файлы **mainwindow.h mainwindow.cpp mainwindow.ui main.cpp** с тем же содержанием, что и выше было:



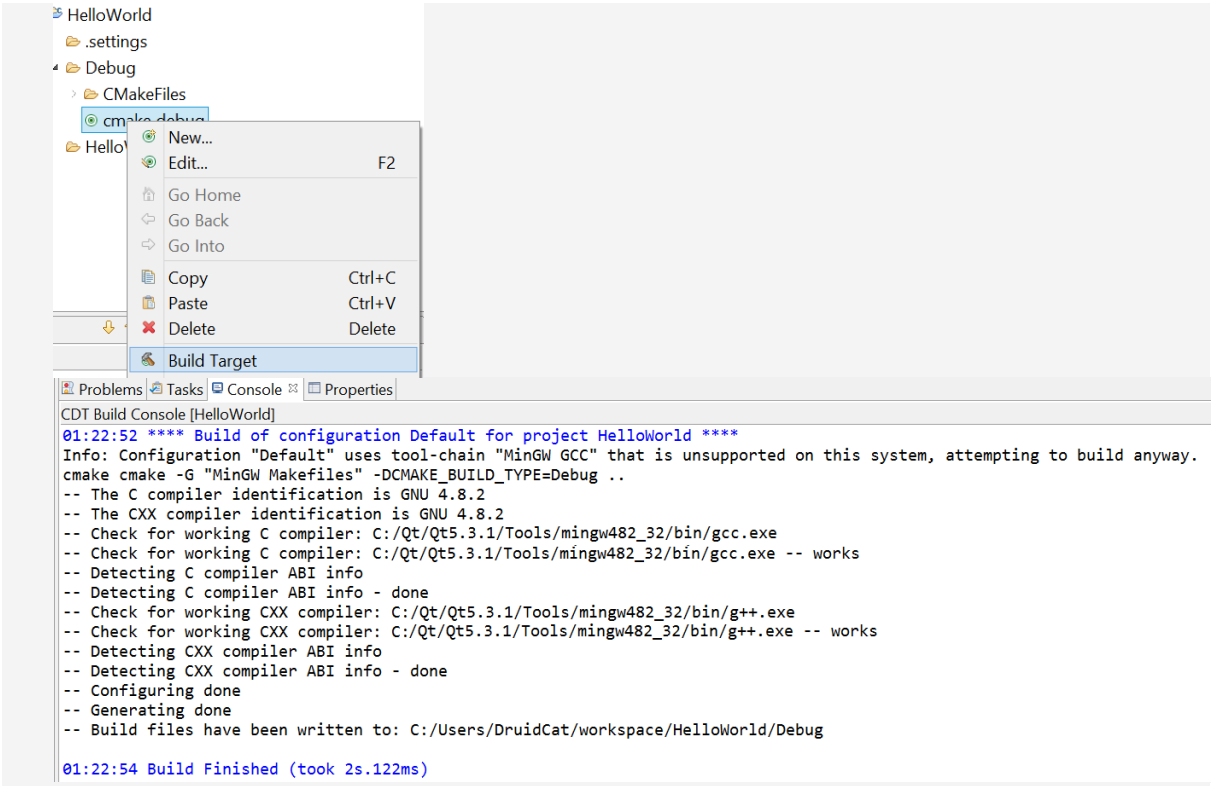
10. Добавляем команду для запуска CMake в Make Target:

cmake -G "MinGW Makefiles" -DCMAKE_BUILD_TYPE=Debug ..





11. Запускаем cmake:

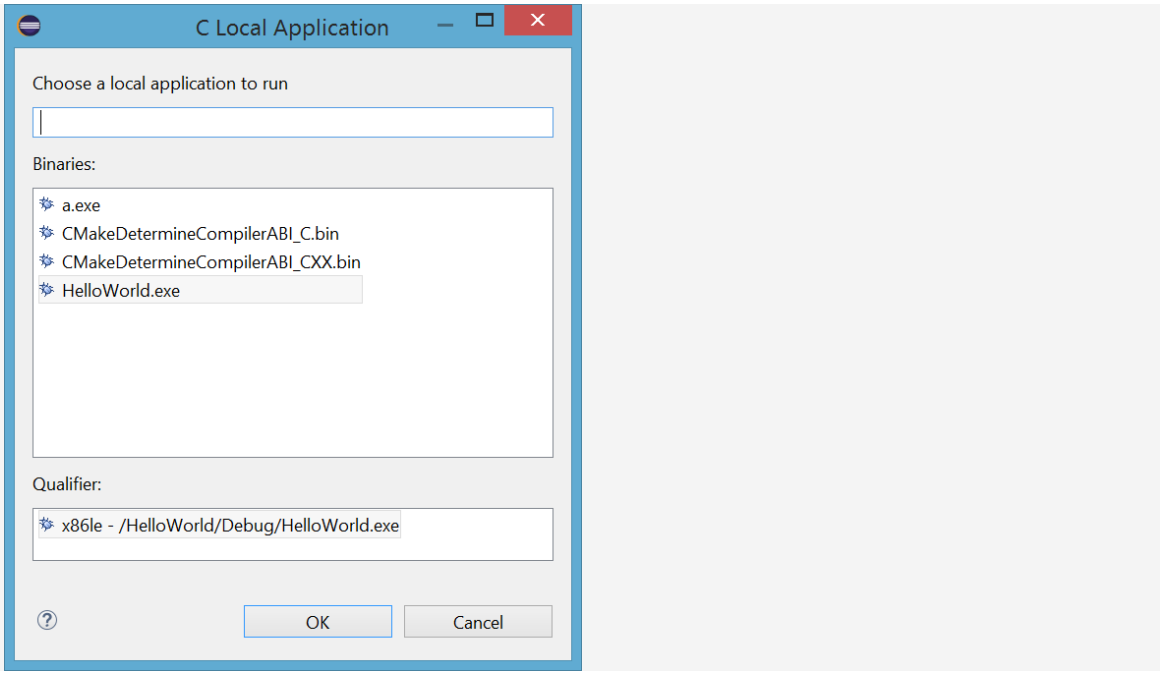


12. Собираю проект через Ctrl+B:

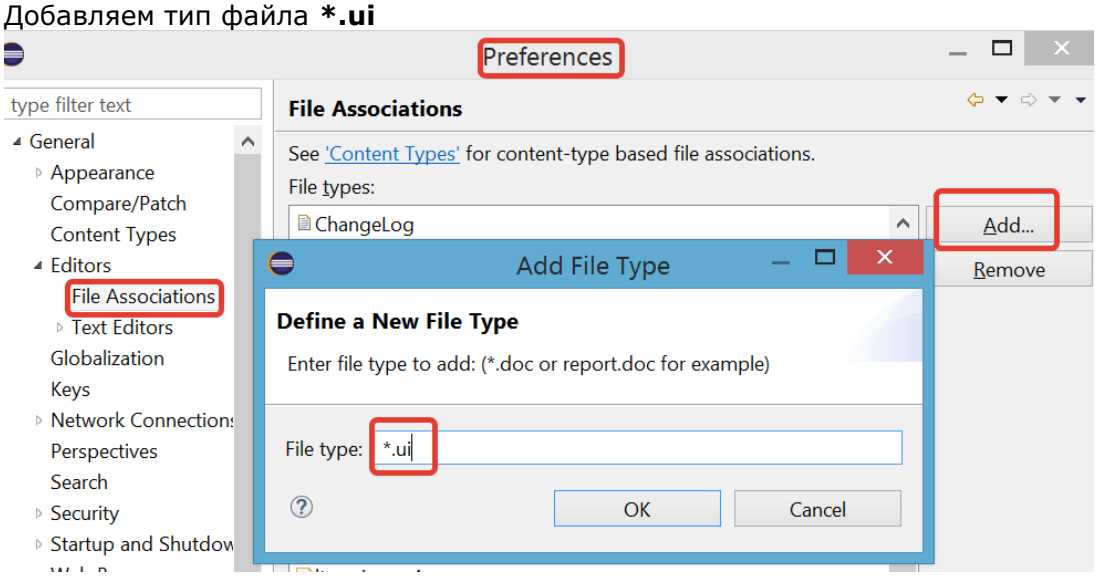
```
01:39:18 **** Incremental Build of configuration Default for project HelloWorld ****
Info: Configuration "Default" uses tool-chain "MinGW GCC" that is unsupported on this system, attempting to build anyway.
mingw32-make -j8 all
Scanning dependencies of target HelloWorld_automoc
[ 20%] Automatic moc for target HelloWorld
Generating moc_mainwindow.cpp
[ 20%] Built target HelloWorld_automoc
[ 40%] Generating ui_mainwindow.h
Scanning dependencies of target HelloWorld
[ 60%] [ 80%] [100%] Building CXX object CMakeFiles/HelloWorld.dir/HelloWorld/main.cpp.obj
Building CXX object CMakeFiles/HelloWorld.dir/HelloWorld/mainwindow.cpp.obj
Building CXX object CMakeFiles/HelloWorld.dir/HelloWorld_automoc.cpp.obj
Linking CXX executable HelloWorld.exe
[100%] Built target HelloWorld

01:39:19 Build Finished (took 1s.680ms)
```

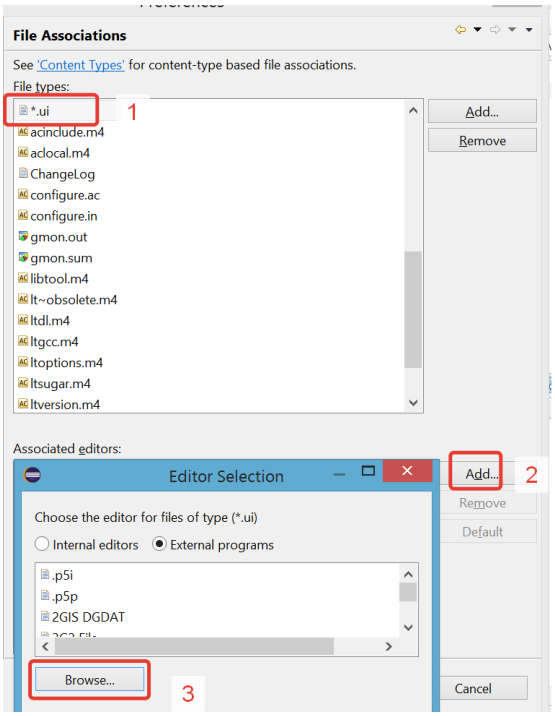
13. Запускаю полученный exe файл через Ctrl+F11 (при этом создается новая конфигурация для запуска):



14. Теперь поговорим о файле **mainwindow.ui**, чтоб пользовать QT Designer в Eclipse настраиваем ассоциацию ui файлов с QT Designer:
Window->Preferences->General->Editors->File Associations->Add
Добавляем тип файла ***.ui**



Жмём **Ок**, в появившемся окне **File types:** выбираем ***.ui**, далее в **Associated editors:** жмём на **Add**, в появившемся окне жмём на **Browse..** и в появившемся окне указываем путь к QT Designer, в моем случае, это:
C:\Qt\Qt5.3.1\5.3\mingw482_32\bin\designer.exe
Теперь, вы можете писать исходный код QT в вашей любимой IDE Eclipse и использовать QT Designer.



15. И еще один момент относительно файла **CMakeLists.txt**, если хотите раскрыть весть потенциал вашего проекта, по изучайте структуру этого файла. Лично я написал этот файл под себя, надеюсь вам это пригодится. Если захотите подгрузить библиотеки QtSerialPort QtSql и т.д., не забывайте их добавлять в **CMakeLists.txt:**

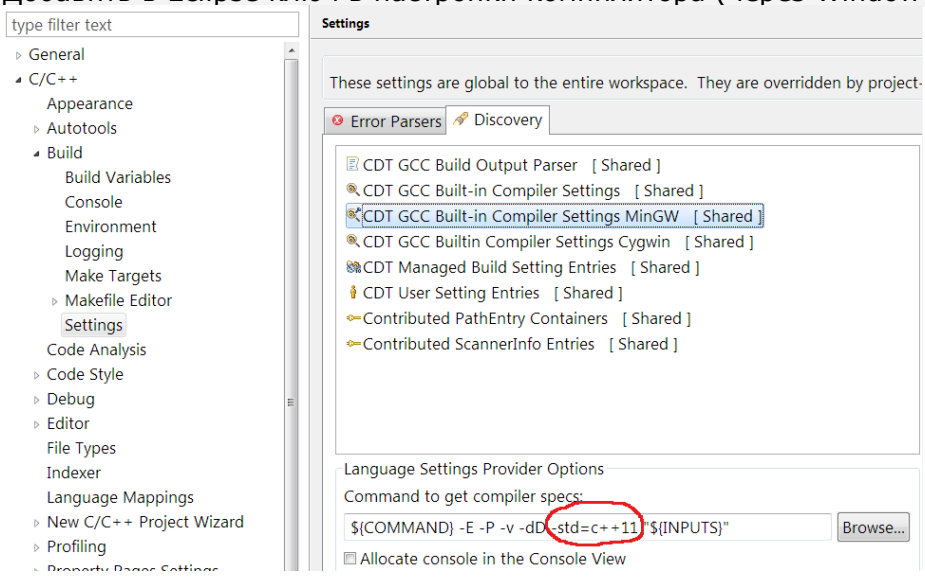
QT5_USE_MODULES(${PROJECT} Widgets SerialPort Sql)

Учтите, регистор букв Widgets SerialPort Sql (и т.д.) крайне важен. Если вы, например, напишите sql, у вас проект не соберется. И не забывайте указать путь этих библиотек в настройках проекта (Пункт №6).

15. Чтобы использовать C++11 нужно добавить в **CMakeLists.txt** строку:

```
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11")
```

Добавить в Eclipse ключ в настройки компилятора (через Window->Preferences):



PS: Данная инструкция актуальна и для Linux, только необходимо изменить некоторые параметры, но если вы поняли как настроить на Windows, то под Linux тем более поймете.

Данную инструкцию написали: kamre и DruidCat.