Tutorial letter 102/3/2019

Advanced Programming

COS3711

Semester 1 and 2

School of Computing

Practical Guide

CONTENTS

Page

1. INTRODUCTION

The purpose of this document is to help you get going using Qt Creator. After getting Qt Creator installed (Section 2), Sections 3 and 4 introduce Qt Creator. Section 5 deals with creating, building, and executing Qt applications without Qt Creator, and Section 6 includes step-by-step instructions to install the Ezust libraries which are discussed in chapter 7 of Ezust.

2. INSTALLING QT CREATOR

If you have not previously installed the Qt Creator, or have an older version installed, you will need to do this installation first. The software is distributed on Disk 2017 – note that you do not have to wait for this disk as you can access it at http://osprey.unisa.ac.za/download/Disk/. Once you have successfully set up the environment as explained here, you should be able to do the following using the Qt Creator (using the *Hello Qt* program).

- Create a project
- Add a file to the project
- Build the project
- And execute the project

2.1 Prescribed software for COS3711

The prescribed software for COS3711 is called Qt Creator, and it is provided on Disk 2017. Qt Creator is actually part of a software development kit (SDK) consisting of a number of different tools for creating applications using the Qt GUI framework. The main components of the SDK are:

- a C++ compiler, namely the MinGW port of the Gnu Compiler Collection (GCC)
- the GUI framework, namely Qt 5
- an IDE, namely Qt Creator

There are numerous other development environments that one can use to develop Qt applications. There are even versions of Qt Creator available for Linux and other operating systems. The particular version provided on this disk works on the Windows 32-bit platform. You are welcome to install and use other versions and combinations of development tools, but if you expect to be supported with installation and other difficulties, you must install the prescribed software as explained below.

2.2 MinGW

MinGW (which stands for Minimalist Gnu for Windows) is a Windows port of the Gnu Compiler Collection. GCC is a collection of compiler tools developed for the Unix operating systems, and has been ported to run on Windows platforms in the form of MinGW.
All C++ programming courses at Unisa use some version of MinGW. When you install Qt Creator, the MinGW port of GCC 4.8.2 is installed automatically within the \Qt directory. This will not interfere with other versions of MinGW that you may already have on your computer.
More information on the MinGW compiler is available from http://www.mingw.org/.

2.3 Qt 5

Qt 5 is a C++ framework for developing applications with graphical user interfaces. The Qt class libraries make available approximately 400 object-oriented classes with most of the functionality needed to build any GUI applications. In fact, the libraries contain classes for GUI, layout, database, internationalization, networking, XML, and much more.
The version of Qt included with Qt Creator is the Qt Open Source Edition, meant for the development of free and open source software only. It is provided free of charge under the terms of both the Qt Public License and the Gnu General Public License.
Information on the Qt framework is available from https://www.qt.io/.

2.4 Qt Creator

The environment supplied on the disk is Qt Creator v 3.1.1-1. It is a cross-platform integrated development environment entirely dedicated to Qt 5. The IDE consists of a Qt project manager, a source

code editor, build-automation tools, and an interface to the GDB debugger. The aim of Qt Creator is to be a powerful development tool that works the same way on all platforms.
More information on Qt Creator is available from http://www.qt.io/ide/.

2.5 Installing the software

An all-in-one self-extracting installer for Qt Creator is available on the disk. Double-click the file/icon labelled qt-opensource-windows-x86-mingw482_opengl-5.3.0.exe and follow the instructions after that. During installation, you will be given a number of options. We recommend that you accept all the defaults.

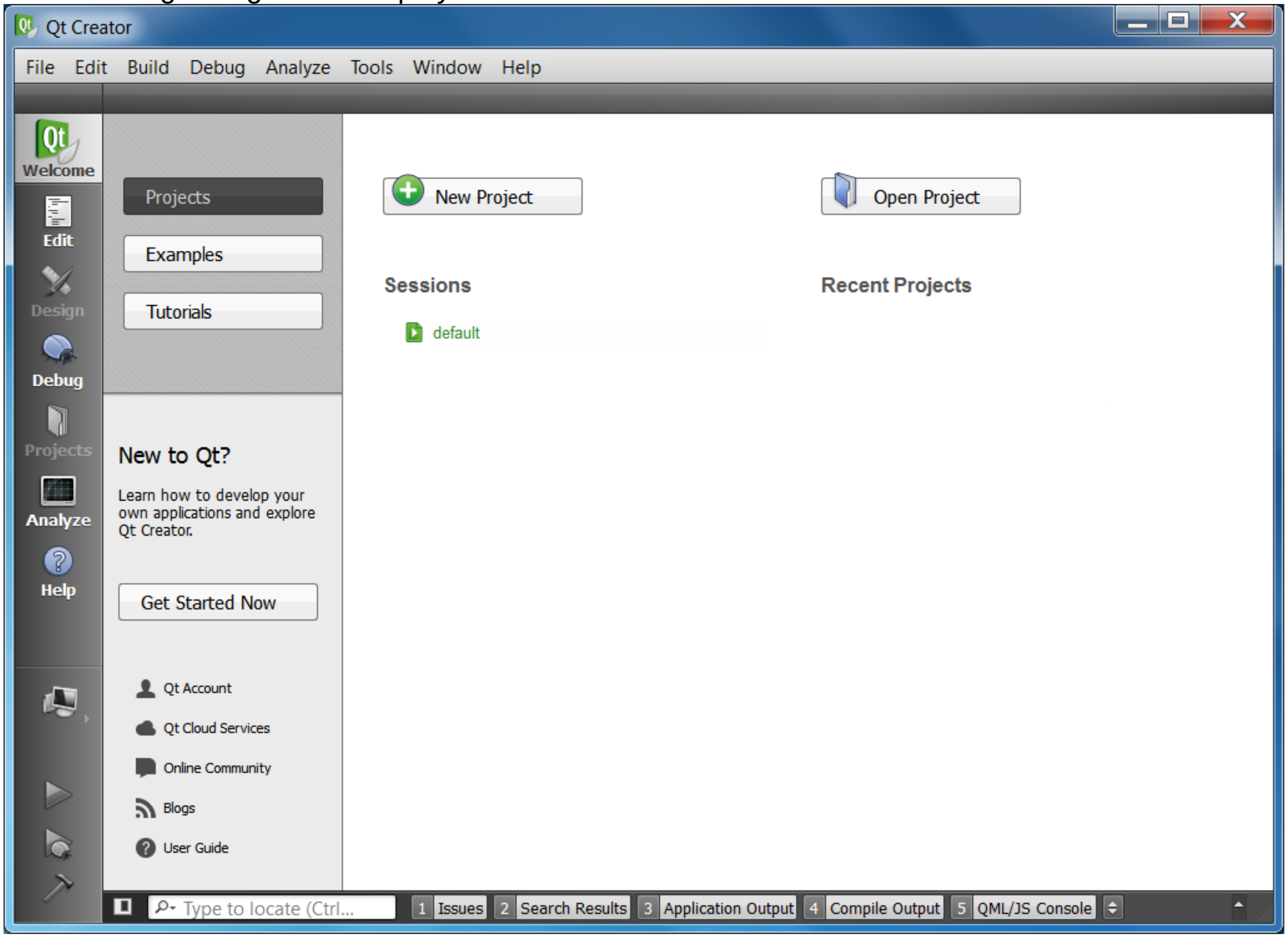
3. USING THE SOFTWARE

To help you get started, we present a short guide to your first "Hello Qt" program.

3.1 Starting Qt Creator

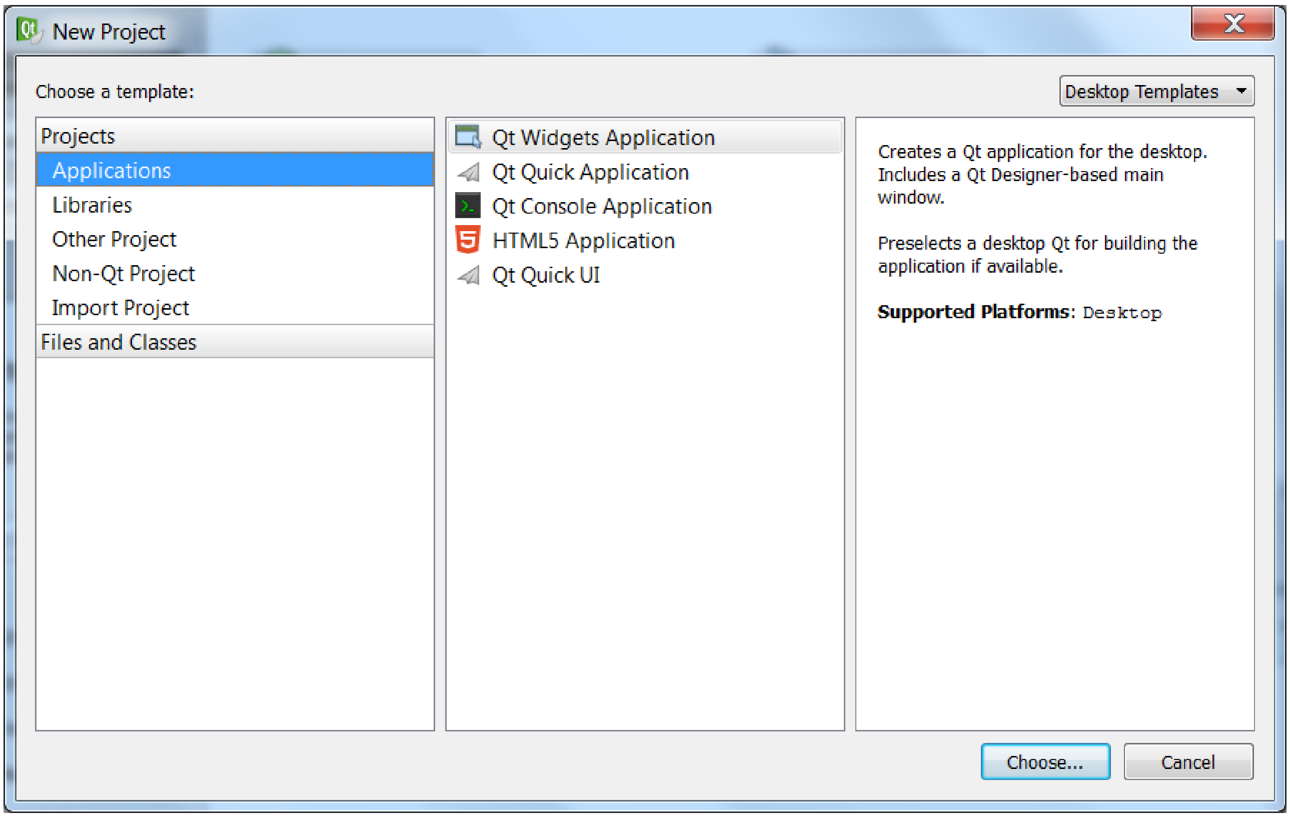
Double-click the Qt Creator shortcut on your desktop.

The following dialog will be displayed:



3.2 Creating a project

Click on the New Project button. The following dialog will be displayed:

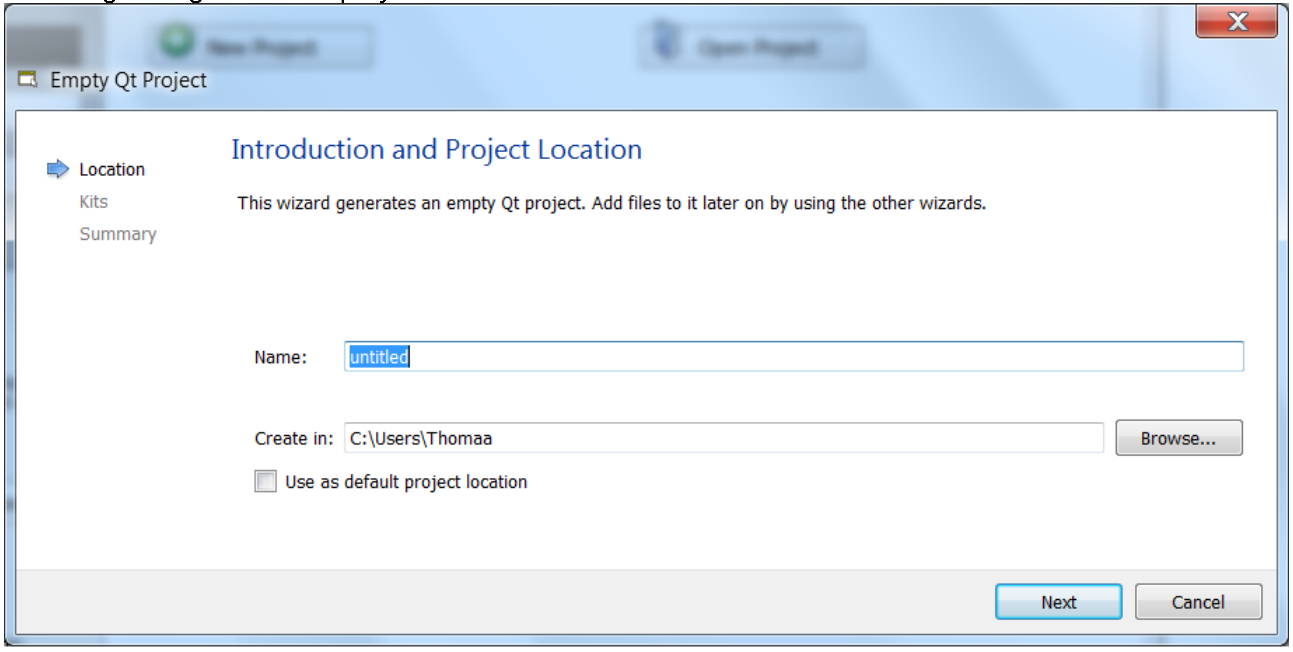


Creates a Qt application for the desktop. Includes a Qt Designer-based main window.

Preselects a desktop Qt for building the application if available.

Supported Platforms: Desktop

Select *Other Project* and then *Empty Qt Project* from the list of options and click on *Choose...*. The following dialog will be displayed:



Introduction and Project Location

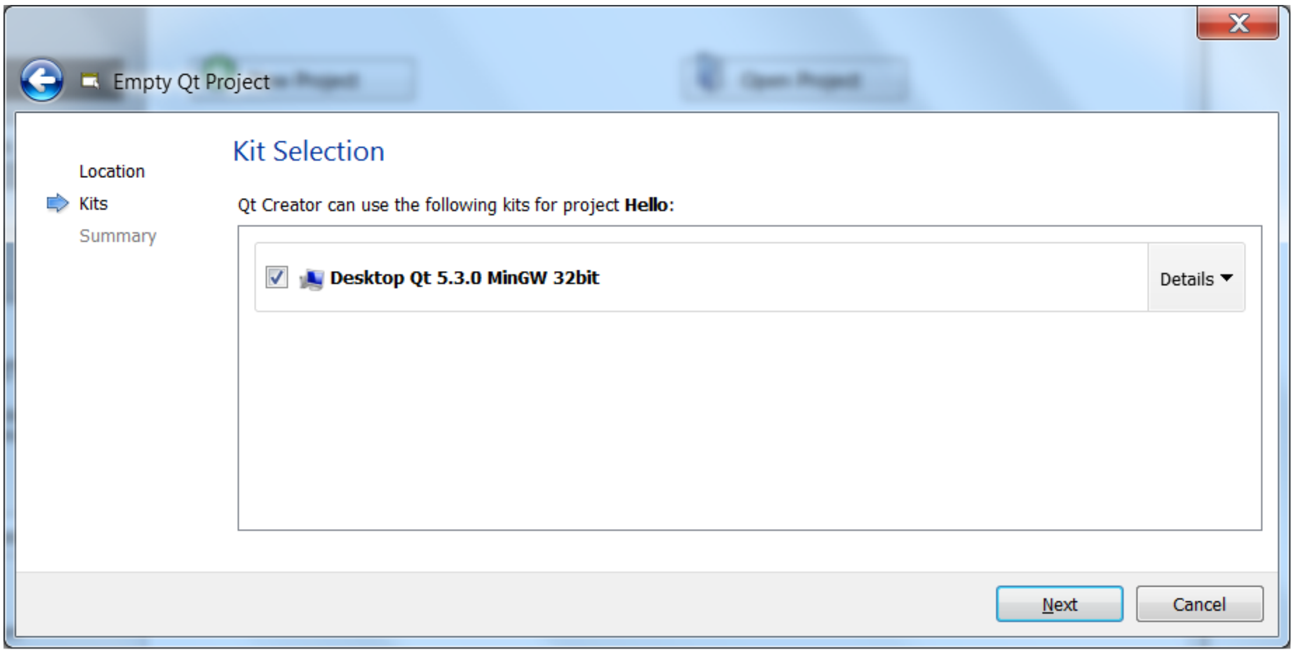
This wizard generates an empty Qt project. Add files to it later on by using the other wizards.

Name: untitled

Create in: C:\Users\Thomaa

☐ Use as default project location

Type Hello as the project name and change the location to your choice (e.g. C:\unisa\cos3711) and click on *Next*. The following dialog will be displayed:

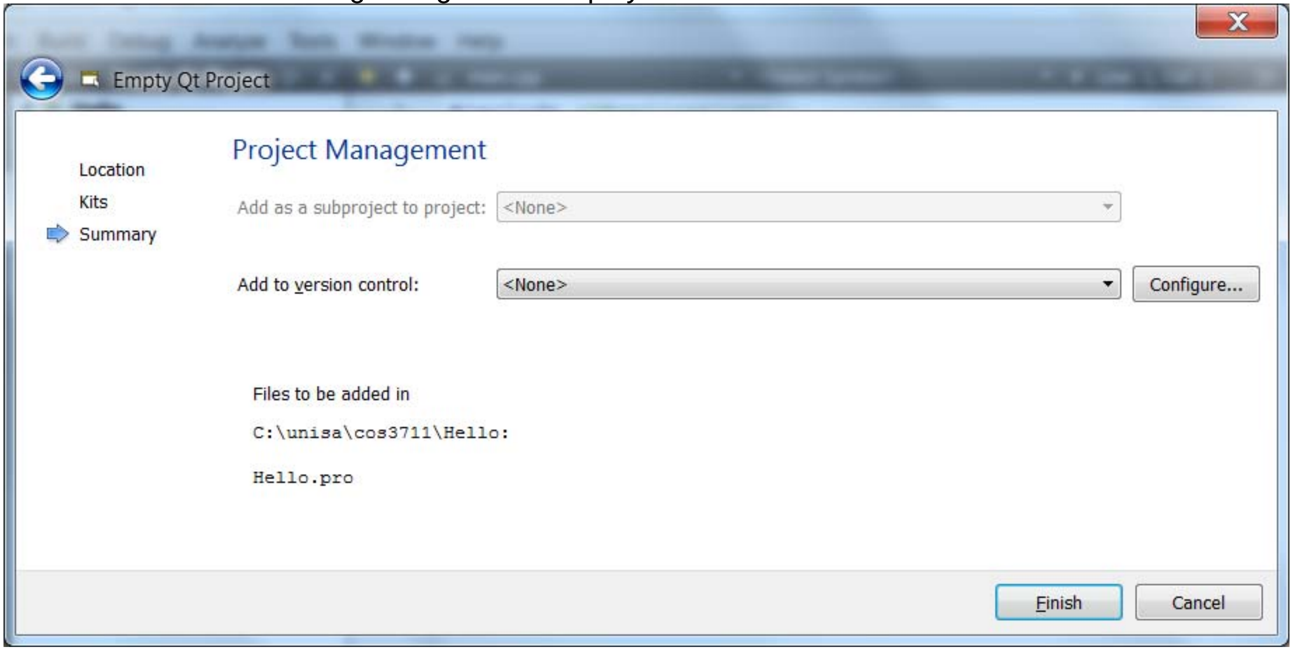


Kit Selection

Qt Creator can use the following kits for project **Hello**:

☑ **Desktop Qt 5.3.0 MinGW 32bit**

Click on *Next*. The following dialog will be displayed:



Project Management

Add as a subproject to project: <None>

Add to version control: <None>

Files to be added in

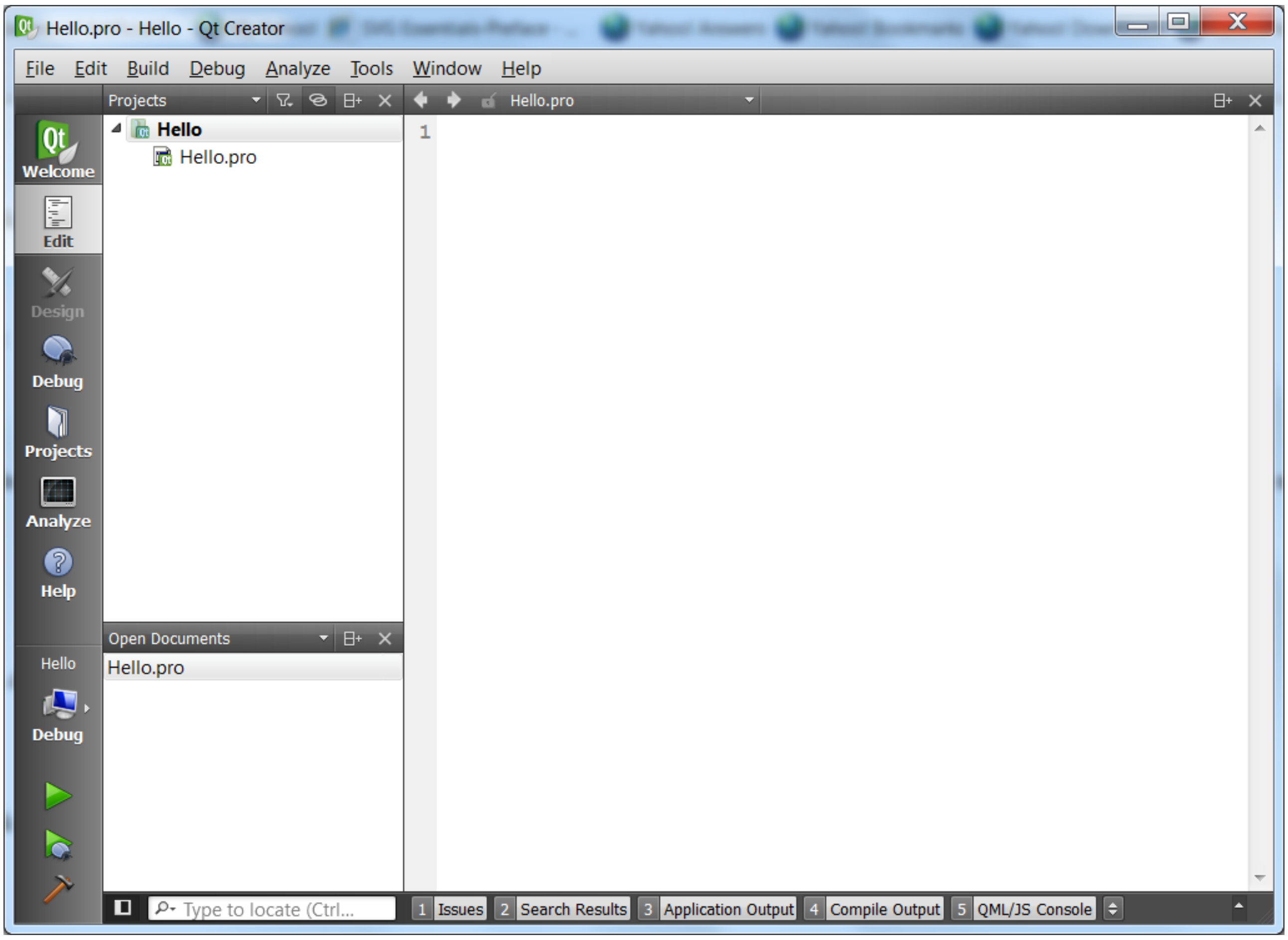
```
C:\unisa\cos3711\Hello:

Hello.pro
```

Leave everything as is on the Project management dialog and click on *Finish*.

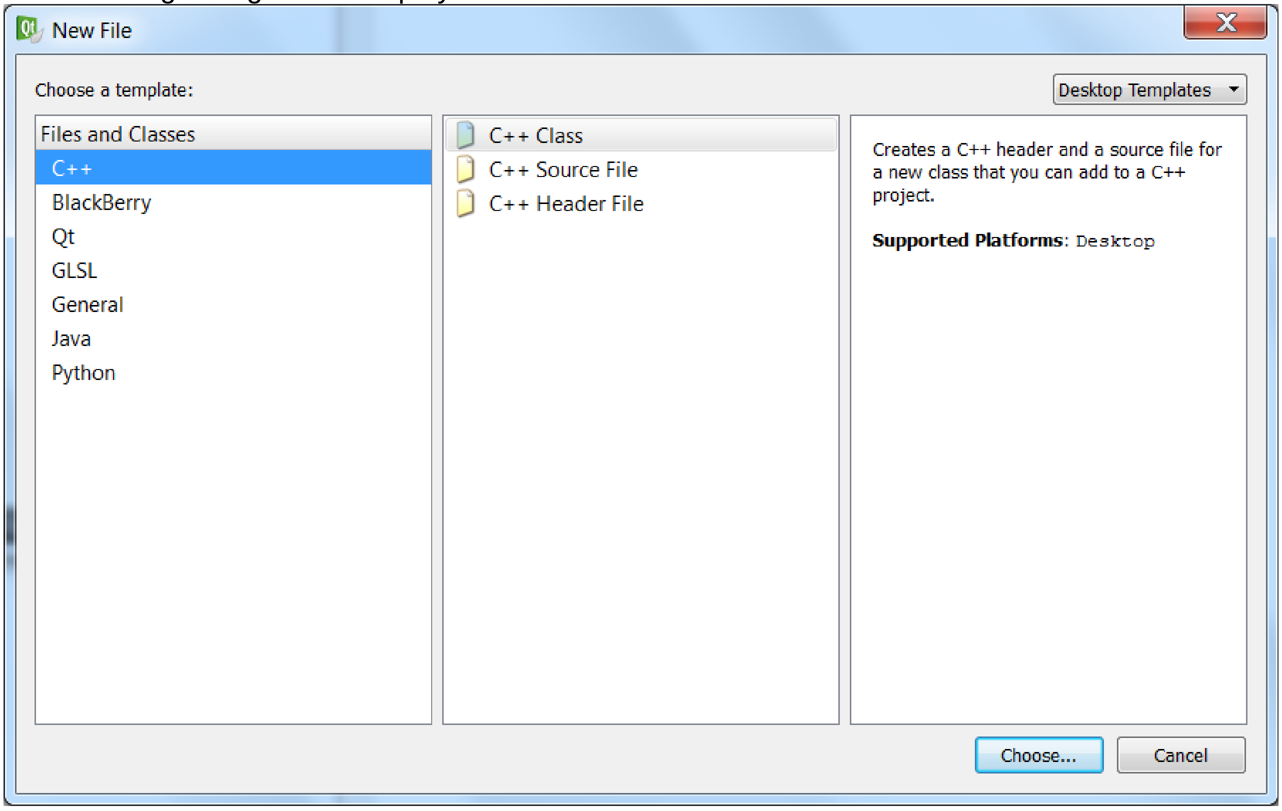
3.3 Editing a C++ source file

At this stage, you have an empty project without any source code:


Hello.pro - Hello - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Projects
Hello.pro
Hello
Hello.pro
Welcome
Edit
Design
Debug
Projects
Analyze
Help
Open Documents
Hello.pro
Hello
Debug
Type to locate (Ctrl...
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console

There are different ways of adding a source file to your project. In this situation, we want to add a new source file, so right-click on the project name Hello and choose *Add New...* on the popup menu.

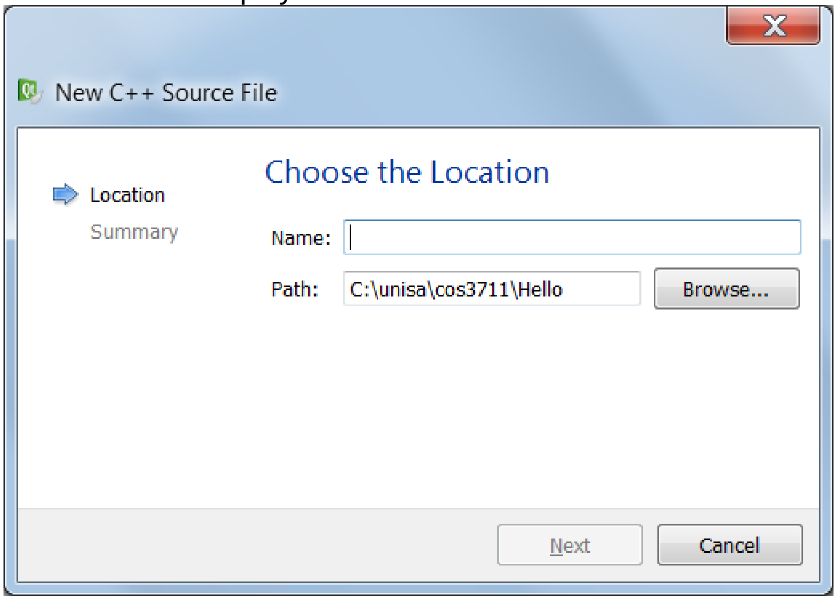
The following dialog will be displayed:



Creates a C++ header and a source file for a new class that you can add to a C++ project.

Supported Platforms: `Desktop`

Select *C++ Source File* from the list and click on *Choose....* A dialog requesting the file name and its location will be displayed:

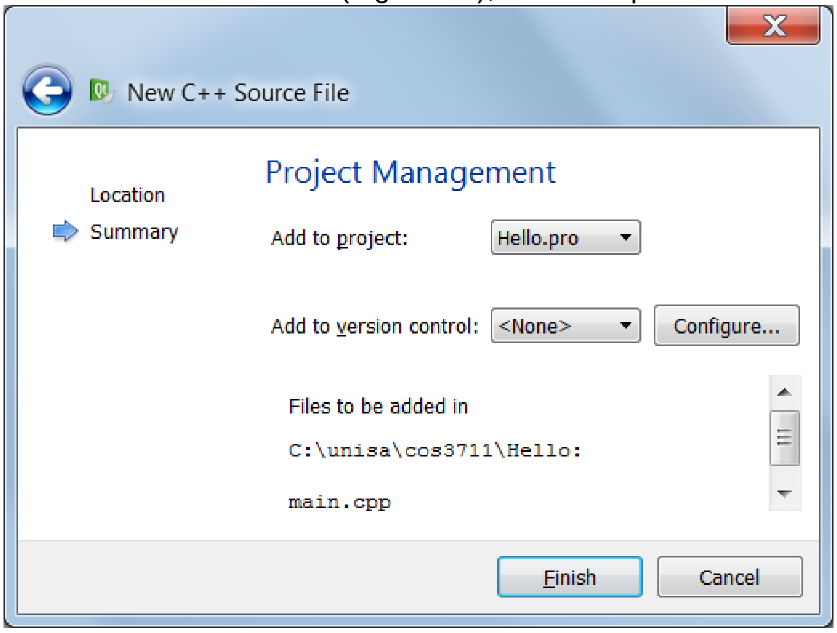


Choose the Location

Name:

Path: C:\unisa\cos3711\Hello

Enter the name of the file (e.g. `main`), leave the path as is and click on *Next*.



Project Management

Add to project: Hello.pro

Add to version control: <None>

Files to be added in

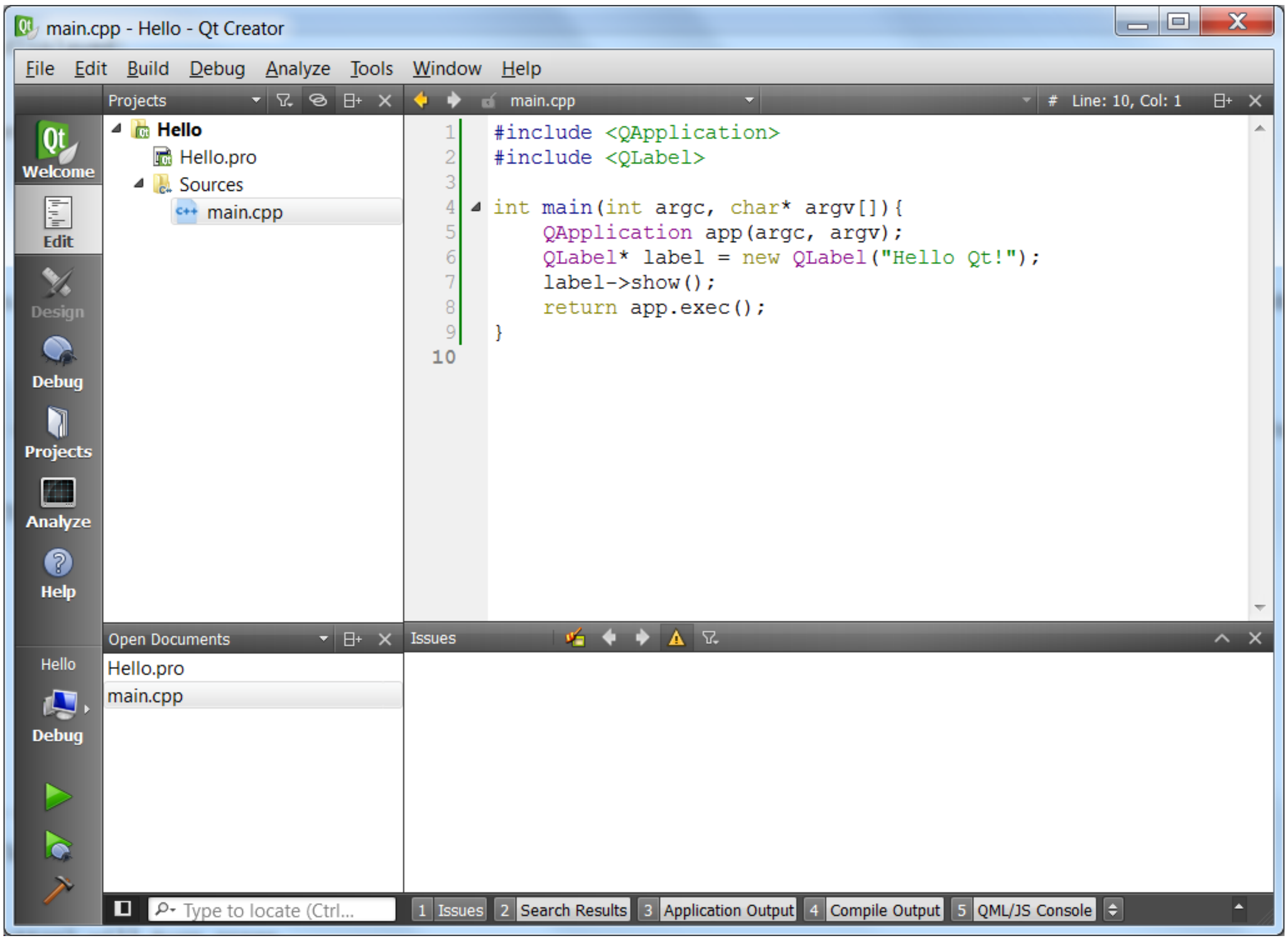
C:\unisa\cos3711\Hello:

main.cpp

Click on *Finish*.

An empty `main.cpp` file will appear in the Edit panel.

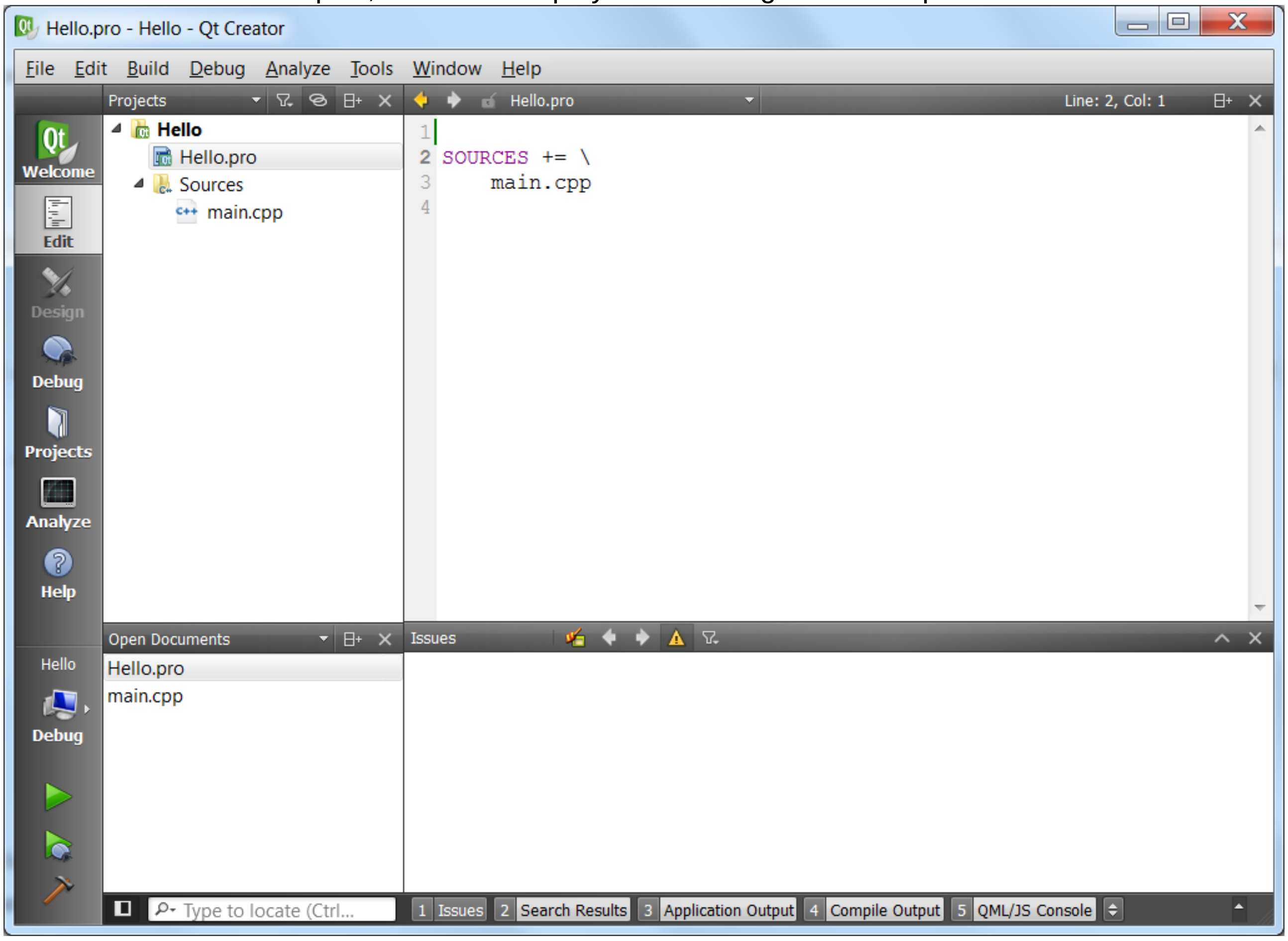
Type in the following code:


main.cpp - Hello - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Projects
Hello
Hello.pro
Sources
main.cpp
Welcome
Edit
Design
Debug
Projects
Analyze
Help
Open Documents
Hello.pro
main.cpp
Issues
Line: 10, Col: 1
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console

```
#include <QApplication>
#include <QLabel>

int main(int argc, char* argv[]){
    QApplication app(argc, argv);
    QLabel* label = new QLabel("Hello Qt!");
    label->show();
    return app.exec();
}
```

Double click on `Hello.pro`, which will display the following in the Edit panel:


Hello.pro - Hello - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Projects
Hello
Hello.pro
Sources
main.cpp
Open Documents
Hello.pro
main.cpp
Issues
Line: 2, Col: 1
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console

```

SOURCES += \
    main.cpp
```

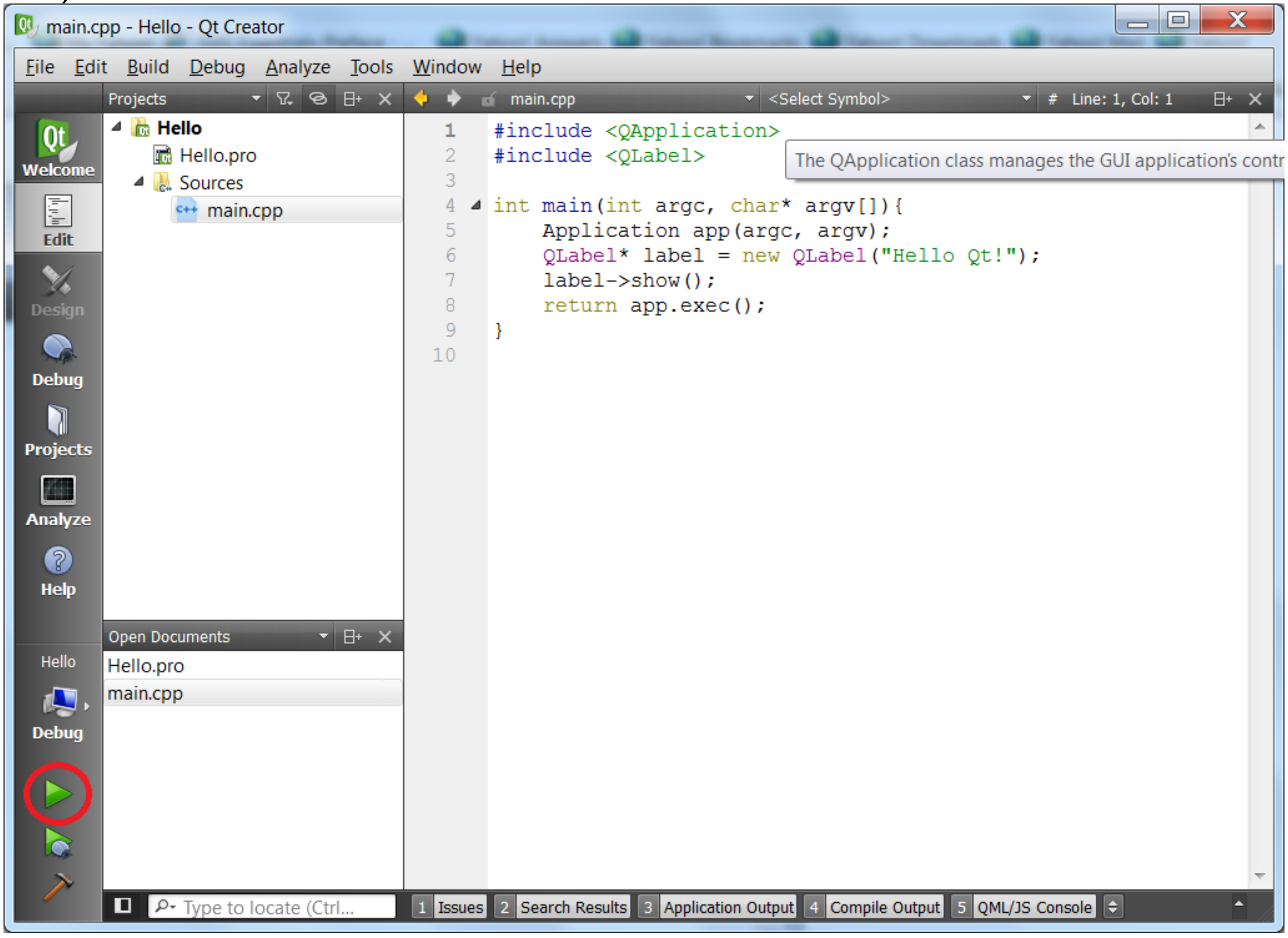
Edit `Hello.pro` to add `QT += widgets`

Select *File* and *Save All* to save all the changes you have made.

3.4 Building and running the program

It is possible to build and run a program separately, but it's easier to do it in one go.

Click on the *Run* button (the button with a green triangle in the bottom left-hand corner - circled in red here).

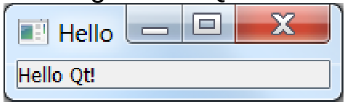


```
#include <QApplication>
#include <QLabel>

int main(int argc, char* argv[]){
     Application app(argc, argv);
     QLabel* label = new QLabel("Hello Qt!");
     label->show();
     return app.exec();
}
```

The QApplication class manages the GUI application's contr

If the build is successful, a small window should appear somewhere on the screen displaying the message Hello Qt!. You can resize the window.



If the build was unsuccessful, make sure you have entered the program correctly and re-run the project.

When you have finished admiring your first Qt program, close the little window.

4. FURTHER USE OF QT CREATOR

Note: If you successfully completed COS2144/COS2614 after 2010 you should be familiar with Qt Creator. If so, you may read through this section briefly.

Our prescribed book, Ezust, assumes that you are using some Linux platform, whereas we assume that the majority of COS3711 students will be using an MS Windows platform. We therefore decided to prescribe a Qt programming environment which is available on both platforms, namely Qt Creator.

When you start up Qt Creator, you will notice buttons down the left hand side of the screen labelled Welcome, Edit, Design, Debug, Projects, Analyze, and Help. Qt Creator starts in Welcome mode.

4.1 Creating projects

In Qt Creator, it is impossible to compile a C++ source code file (to get an executable file) without having it part of a project. The Welcome mode of Qt Creator therefore allows you to create a new project or open a recent project. You can also open an existing project not listed as a recent project by means of the File drop-down menu.

To create a new project in Qt Creator, follow the instructions given above.

4.2 Adding and removing files from projects

The instructions above explain how to create a new source code file and add it to an empty project. However, sometimes you have source code in another file that you want to use instead of **main.cpp**. Do the following:

- In Edit mode, right-click on the project name in the Projects panel, and choose **Add Existing Files ...** Browse to the source code file (**.cpp**) that you want to add, select it and click on **Open**. It will be added to the list of source files under the current project in the Projects panel. (In Edit mode, double-click on any file in the Projects panel to get it displayed in the editor on the right.)

Sometimes you need to remove a source code file from your project. Do the following:

- In Edit mode, right-click on the name of the file you want to remove (e.g. **unwanted.cpp**) in the Projects panel and choose **Remove File ...** (You can click the **Delete file permanently** checkbox if you want to, but this is not recommended as you will effectively be throwing code away.) Click on OK to remove the file from the project. This means that when you compile your project again, this file will not be compiled and its object code will not be linked with the other code in your project to form the executable.

4.3 Entering command line arguments

In order to enter command line argument, select Projects mode, choose the Build & Run tab, and then the Run tab, and type in the command line arguments in the Arguments textbox.

4.4 Getting context-sensitive help

You can get information on any Qt class or method in your source code by moving the cursor onto the class or method name in the editor and pressing the F1 key.

Note: At this level of programming, you are expected to use the API documentation extensively. You can browse the documentation through the *Index* option via the *Help* menu in Qt Creator.

4.5 Compiling Ezust code

All the code for the examples in Ezust is available in the **src.tar.gz** file on Disk 2017. You will need to unzip this file (you may use 7-zip which is also on Disk 2017) to the hard-drive of your computer (or to a memory stick) to be able to access any of the code in it. We recommend that you unzip and keep all the Ezust code together. Do this:

- Double-click on **src.tar.gz** and extract all the files to the directory where you want them (e.g. **C:\unisa\cos3711**). Although you can work on this code (editing and changing it) where you have unzipped it, we suggest that you copy the code as you need it to a separate project directory. Then you can edit the code to your heart's content and still leave the original code intact. The following instructions explain how to do this:

- Create a new project (as explained in **Creating projects** above). This will create a directory with the name of the project. Next, copy the Ezust code files from where you unzipped them to this directory. Finally, add the necessary files to the project (as explained in **Adding and removing files from a project** above). Then build and run the project.

Note:
(1) `src.tar.gz` will also be uploaded on *my*Unisa under Additional Resources.
(2) If a program provided by Ezust does not work, please refer to *my*Unisa under Additional Resources to obtain a working copy of the code. Lecturers will be uploading corrected code as necessary on *my*Unisa.

5. WITHOUT USING QT CREATOR

This section discusses how one can build and execute Qt programs using the command prompt. Refer to Section 1.6 and Appendix C, Development Tools in Ezust.

Note: It is beneficial to learn to build and execute programs from the command prompt. However, this skill will not be tested in your assignments or in the exam.

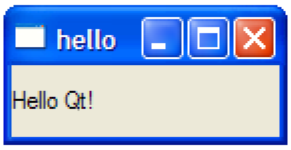
The typical cycle of developing an application is:
1. Enter the program.
2. Create a platform-independent project file.
3. Create a platform-specific makefile from the project file.
4. Make the file.
5. Run the file.

In order to understand how Qt operates let's go through the steps by creating and running an application without using **Qt Creator**.

Step	**Implementation**
Enter the program	• Use Notepad to create a source file `hello.cpp` and save it in the directory `c:\cos3711\hello`. `#include <QApplication>` `#include <QLabel>` `int main(int argc, char *argv[])` `{` `QApplication app(argc, argv);` `QLabel *label = new QLabel("Hello Qt!");` `label->show();` `return app.exec();` `}`
Set the paths	• Open the command prompt. • If you followed the default installation instructions of Qt Creator, typing the following will set the path correctly: `path=C:\Qt\Qt5.3.0\5.3\mingw482_32\bin;` `C:\Qt\Qt5.3.0\Tools\mingw482_32\bin` Note: If you installed Qt in a folder other than the default folder the paths should be set to the folders where `qmake` and `mingw32-make` are placed.
Create a platform-independent project file	• From the command prompt, change the directory to `hello` • Type in `qmake -project` • A platform-independent project file (`hello.pro`) is created. (Enter `dir` to see files in directory.)
Create a platform-specific makefile from the project file.	• Type in `qmake hello.pro` • A Makefile is created as well as other files and two directories. • A `dir` will display the following: `…` `<DIR>          debug` `201 hello.cpp`

	`312 hello.pro`
	`5,005 Makefile`
	`4,360 Makefile.Debug`
	`4,414 Makefile.Release`
	`<DIR>         release`
	…

Make the file.	• To make the file type `mingw32-make` at the command prompt.

Run the file.	• As a debug version is created by default, you just change directories to `c:\cos3711\hello\debug`.
	• A `dir` will reveal the following:
	…
	`1,355,316 hello.exe`
	`366,848 hello.o`
	…
	• Type in `hello` and this window will be displayed. (It can be resized)
	
	hello
	Hello Qt!

6. INSTALLING EZUST LIBRARIES

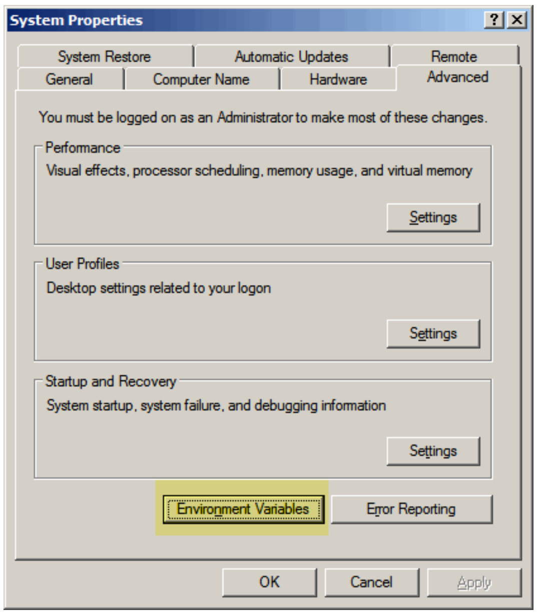
6.1 Building the libraries

Note: It is essential to compile the Ezust libraries to use many of the programs in the textbook. Please follow the instructions in this section carefully.

To get the code to work in examples that are part of the Ezust libraries, it is necessary to create a compiled version of these libraries. Compiling libraries is covered in Chapter 7.

Firstly, an environment variable needs to be set. To set an environment variable:

1. Go to “Control Panel” (Left-click on the “Start” and then left-click on “Control Panel” or right-click the "Start" icon followed by selecting “Settings” and “Control Panel”).
2. Select the "System" icon.
3. Click the "Advanced" tab
4. Click the "Environment Variables" button. Under the "User Variables" section, click “New”.



System Properties

System Restore | Automatic Updates | Remote
General | Computer Name | Hardware | Advanced

You must be logged on as an Administrator to make most of these changes.

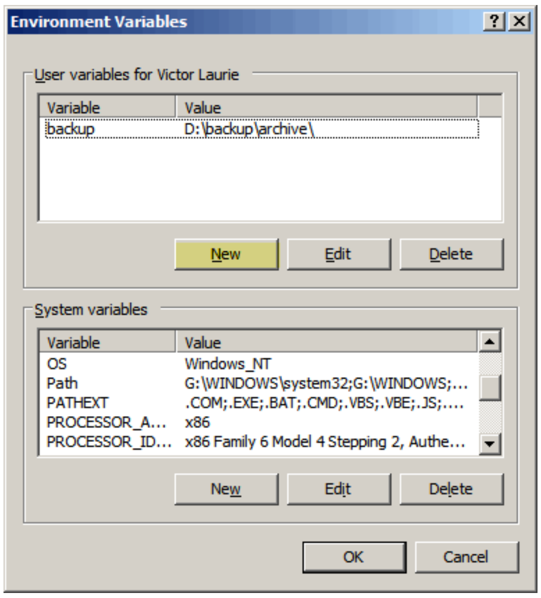
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings

User Profiles
Desktop settings related to your logon
Settings

Startup and Recovery
System startup, system failure, and debugging information
Settings

Environment Variables | Error Reporting

OK | Cancel | Apply



Environment Variables

User variables for Victor Laurie

Variable	Value
backup	D:\backup\archive\

New | Edit | Delete

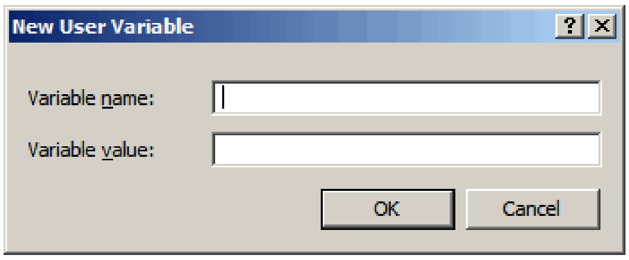
System variables

Variable	Value
OS	Windows_NT
Path	G:\WINDOWS\system32;G:\WINDOWS;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	x86
PROCESSOR_ID...	x86 Family 6 Model 4 Stepping 2, Authe...

New | Edit | Delete

OK | Cancel

5. Enter **CPPLIBS** in the "Variable name" field and the path (**c:\projects\libs**)in the "Variable value" field.



6. Click OK

If the creation of the environment variable was successful, it will be displayed in the "Environment Variables" window with the "Variable" name "CPPLIBS" and "Value" c:\projects\libs.

Now follow the steps below to prepare the Ezust libraries.

☞ Download and extract the updated libs folder from Additional Resources on *my*Unisa to the c:\projects folder.

☞ Open the libs.pro (It should be available in c:\projects\libs) using Qt Creator. Click Configure Project in the Configure Project setup dialog box.

☞ Build `libs.pro` (Build – Build All; do not run it). Note that this may take a while, and that there may be some warnings produced.

☞ If you managed to build the libraries correctly, verify that several `.dll` files (specifically `dataobjects.dll`) are created in the C:\projects\libs folder.

☞ You can now close the libs project in Qt Creator.

6.2 Using the libraries

If you are creating a project of your own that needs the libraries supplied with the book, you need to add a few additional lines in the `.pro` file.

For example if you are using classes from the `dataobjects` library, then you need to add the following two lines in the `.pro` file.

```
LIBS += -L$$(CPPLIBS) -ldataobjects
INCLUDEPATH += $$(CPPLIBS)/dataobjects
```

If you are using classes from different libraries then you need to add the above statements for each library in the `.pro` file. Once you have included the above statements in the `.pro` file, there is no need to explicitly add a header or `.cpp` file from the dataobjects library into your project.