Лабораторная работа 1.
Qt. Установка и настройка библиотеки Qt и IDE"Qt Creator"

Литература:

1. Шлее, М. Qt 5.10. Профессиональное программирование на C++/ М. Шлее. – СПб.: БХВ-Петербург, 2018. – 1072 с.
2. https://doc.qt.io/qt-5.15/gettingstarted.html

Qt – кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++.

Qt – это полноценная среда разработки с инструментами, предназначенными для упрощения создания приложений и пользовательских интерфейсов для настольных, встроенных и мобильных платформ.

1.1 Установка

Наряду с коммерческими вариантами Qt на официальном сайте библиотеки http://www.qt.io) предлагается также и открытая лицензия (LGPL). Скачать программу установки для использования открытой лицензии IDE “Qt Creator” можно на странице https://www.qt.io/download-open-source. Здесь доступны как несколько вариантов установщиков. Предусмотрены варианты установки среды Qt Creator, которая будет использовать компилятор IDE "Visual Studio". Однако рекомендуем обратить внимание на версию, использующую компилятор MinGW.

Процесс установки не должен вызвать особых вопросов. Сначала необходимо создать учетную запись и зайти, используя уже существующую. Далее выбрать компоненты установки. На этом этапе необходимо убедиться, что в качестве одного из элементов установки указан компилятор MinGW. После необходимо ознакомиться с лицензионным соглашением и завершить установку.

1.2 Структура проекта Qt. Общие рекомендации по разработке проектов с использованием библиотеки Qt

В проект Qt входят файл конфигурации проекта (*.pro), исходные файлы (*.cpp) и заголовочные файлы (*.h). В начале файла конфигурации производиться подключение необходимых модулей. Для подключения модуля используется конструкция `QT +=` (например, `QT += core gui widgets`).

Существует много различных по своему предназначению модулей Qt: https://doc.qt.io/qt-5.15/qtmodules.html

Например:

- ***QtCore** – ядро Qt. Базовый модуль, который содержит класс QObject, контейнерные классы, классы для ввода и вывода, классы моделей интервью, классы для работы с датой и временем и др., но не содержит классов относящихся к интерфейсу пользователя.*
- ***QtGUI** – модуль базовых классов для программирования пользовательского интерфейса. Содержит класс QGuiApplication, который поддерживает механизм цикла событий, позволяет получить доступ к буферу обмена, позволяет изменять форму курсора мыши и т.д.*
- ***QtWidgets** – модуль содержащий классы, которые являются “деталями” при реализации пользовательского интерфейса. В его состав входит класс QWidget – базовый класс всех элементов управления библиотеки Qt, классы автоматического размещения*

элементов, классы меню, классы диалоговых окон и окон сообщений, классы для рисования, а также классы всех стандартных элементов управления.

- ***QtSql*** *– модуль отвечающий за поддержку работы с базами данных.*

Кроме перечисленных выше модулей библиотека Qt поддерживает также: QtNetwork для работы с сетями, QtOpenGL для работы с графикой OpenGL, QtTest содержащий вспомогательные классы для тестирования кода, QtXML отвечающий за поддержку XML и многие другие.

Далее в файле конфигурации проекта можно задать следующие параметры (в виде пары: **ПАРАМЕТР = ЗНАЧЕНИЕ**) https://doc.qt.io/qt-5/qmake-variable-reference.html:

TARGET – имя приложения. Если данное поле не заполнено, то название программы будет соответствовать имени проектного файла;

TEMPLATE – указывает тип проекта. Например: app – приложение, lib – библиотека.

FORMS – список файлов с расширением ui. Эти файлы создаются программой Qt Designer и содержат описание интерфейса пользователя в формате XML.

Кроме того, в файле проекта указывается список заголовочных файлов (ключевое слово HEADERS) и файлов исходного кода (ключевое слово SOURCES) в виде КЛЮЧЕВОЕ_СЛОВО += ИМЯ_ФАЙЛА1 ИМЯ_ФАЙЛА2 и т.д.

Важно отметить, что при добавлении в проект файлов, QtCreator автоматически вносит изменения в файл проекта. Например, при добавлении в пустой проект HelloWorld файла `main.cpp`, в файле HelloWorld.pro появится строка `SOURCES += main.cpp`.

Также в файле проекта допустимо устанавливать следующие параметры:

LIBS – список библиотек, которые должны быть подключены для создания исполняемого модуля;

CONFIG – настройки компилятора;

DESTDIR – директория, в которую будет помещен исполняемый файл;

INCLUDEPATH – каталог, содержащий заголовочные файлы;

и другие.

При разработке проекта файлы классов лучше всего разбивать на две отдельные части. Часть определения класса помещается в заголовочный файл *.h, а реализация класса – в файл с расширением .cpp.

В заголовочном файле с определением класса необходимо указывать директиву препроцессора `#ifndef` или `#pragma once`. Такой подход применяется во избежание конфликтов в случае, когда заголовочный файл включается в исходные файлы более одного раза. По традиции заголовочный файл, как правило, носит имя содержащегося в нем класса.

Для ускорения компиляции, при использовании в заголовочных файлах указателей на типы данных рекомендуется предварительно объявлять эти типы, а не напрямую включать их определения посредством директивы `#include`.

В случае, когда предполагается использование механизма сигналов и слотов или приведение типов при помощи функции `qobject_cast<T>()`, рекомендуется в начале определения класса указать макрос `Q_OBJECT`.

Например,

```
Class MyClass : public QObject {
      Q_OBJECT
   public:
     MyClass();
     ...
};
```

Основную программу рекомендуется размещать в отдельном файле. В этом файле должна быть реализована функция `main()`. В связи с этим, в качестве имени для такого файла зачастую выбирают `main.cpp`.

1.3 Задание 1. Первая программа на Qt – «Hello World»

Создадим первую программу на Qt (см. [1, стр.26]).

Создайте новый проект («Other project» –> «Empty qmake project»). Проект назовите HelloWorld, выберите компилятор и завершите создание проекта.

В файл HelloWorld.pro запишите следующий код:

```
QT += core gui widgets
TARGET = HelloWorld
TEMPLATE = app
```

В контекстном меню проекта нажмите «Add New…» и выберите C++ Source File. Назовите его `main`. В проекте должен появиться каталог исходных файлов, содержащий `main.cpp`. Обратите внимание на то, что в файл `HelloWorld.pro` автоматически добавились следующие строки:

```
SOURCES += \
    main.cpp
```

Это означается, что файл `main.cpp`, находящийся в корне каталога проекта, включен в проект. Теперь наполните файл main.cpp следующим содержимым:

```
#include<QtWidgets>
int main(int argc, char** argv){
    QApplication app(argc, argv);
    QLabel* lbl = new QLabel("Hello World!");
    lbl->show();
    return app.exec();
    }
```

В первой строке функции `main` создается объект класса `QApplication`, который осуществляет контроль и управление приложением. Любая использующая Qt-программа с графическим интерфейсом должна создавать только один объект этого класса, и он должен быть создан до использования операций, связанных с пользовательским интерфейсом.

Затем создается объект класса `QLabel`. После создания элементы управления Qt по умолчанию невидимы, и для их отображения необходимо вызвать метод `show()`. Объект класса `QLabel` является основным управляющим элементом приложения, что позволяет завершить работу приложения при закрытии окна элемента.

Наконец, в последней строке программы приложение запускается вызовом `QApplication::exec()`. С его запуском приводится в действие цикл обработки событий, который передает получаемые от системы события на обработку соответствующим объектам. Он продолжается до тех пор, пока либо не будет вызван статический метод `QCoreApplication::exit()`, либо не закроется окно последнего элемента управления. По завершению работы приложения метод `QApplication::exec()` возвращает значение целого типа, содержащее код, информирующий о его завершении.

1.4 Задание 2. Пример «Getting Started Programming with Qt Widgets»

Выполнить инструкции https://doc.qt.io/qt-5/qtwidgets-tutorials-notepad-example.html

Задание предусматривает создание приложения, которое представляет собой небольшой текстовый редактор, который позволяет вам создать текстовый файл, сохранить его, распечатать или повторно открыть и отредактировать снова. Вы также можете установить шрифт, который будет использоваться.

1.5 Задание 3. Прочитать!

1) Прочитать указанные главы книги 1:
   a) Главы 1 – 3
   b) Глава 47.
2) В текстовом виде подготовить вопросы (20-25 шт) по выполненной лабораторной работе.