

П. В. Матренин

**Введение в
кроссплатформенное
программирование на C++ в
Qt**

Методические указания

**Москва
Издательство Нобель Пресс**

УДК 62-63
ББК 30.6
П11

- П11 **П. В. Матренин**
Введение в кроссплатформенное программирование на C++ в Qt: Методические указания / П. В. Матренин – М.: Lennex Corp, — Подготовка макета: Издательство Нобель Пресс, 2021. – 55 с.

ISBN 978-5-518-40395-6

Рассмотренные в методическом пособии теоретические положения и учебные примеры позволяют студентам быстро освоить технологию создания программных продуктов в среде Qt 4 на языке C++. Qt является кроссплатформенным инструментом разработки программного обеспечения любой сложности, которое будет работать во всех основных операционных системах (Windows, Mac OS X, Unix, Linux, системы для мобильных устройств и встраиваемые системы) без изменения исходного кода. Qt используется в AMD, Disney, Google, HP, KDE, Oracle, Siemens, Skype, Sumsung и др. В пособии рассмотрены особенности Qt, дан обзор IDE Qt Creator и приведены два учебных примера, в которых подробно, по шагам описан процесс создания приложений с графическим интерфейсом пользователя, механизм сигналов и слотов, работа с текстовыми файлами, создание интерфейсов в визуальном дизайнера и процесс интернационализации приложений. Приведены исходные коды с комментариями и пояснениями.

ISBN 978-5-518-40395-6

© Издательство Нобель Пресс, 2021
© П. В. Матренин, 2021

Оглавление

1. Общие положения.....	5
1.1. Цель лабораторных работ.....	5
1.2. Темы лабораторных работ.....	5
1.3. Система программирования Qt.....	6
2. Обзор среды разработки.....	7
2.1. Знакомство с Qt, сигналы и слоты.....	7
2.2. Запуск системы Qt Creator.....	10
2.3. Краткий обзор Qt Creator.....	11
2.3.1 Панели вывода	12
2.3.2 Управление сессиями в Qt Creator	13
2.3.3. Интеграция справки по Qt	13
2.3.4. Интеграция с Qt Designer	14
2.4. Текстовый редактор.....	15
2.5. Создание проекта в Qt Creator	19
2.5.1. Создание нового проекта	19
2.5.2. Установка имени проекта и его размещения	20
2.5.3. Указание информации о классе.....	20
2.6. Отладка с помощью Qt Creator	21
2.6.1. Введение	21
2.6.2. Взаимодействие с отладчиком	21
2.6.3. Точки останова.....	21
2.6.4. Выполнение.....	22
2.6.5. Стек.....	23
2.6.6. Переменные и выражения.....	23
2.6.7. Дизассемблер и Регистры	25
2.6.8. Потоки.....	25
2.7. Настройки среды	25
3. Лабораторная работа № 1	26
3.1. Основы на примере создания текстового редактора	26
3.1.1. Начало разработки нового приложения	26
3.1.2. Проектные файлы .pro.....	28
3.1.3. Описание класса главного окна.....	29
3.1.4. Конструктор главного окна	31
3.1.5. Обработка событий.....	31
3.1.6. Реализация слотов.....	32
3.1.7. Иконка и ресурсы	33

3.1.8. Действия	35
3.1.9. Создание меню.....	37
3.1.10. Создание панели инструментов	37
3.1.11. Строка состояния	38
3.1.12. Сохранение настроек приложения.....	38
3.1.13. Функции для работы с файлами	39
3.1.14. Запуск приложения.....	41
3.2. Дополнительные возможности	44
3.2.1. Добавление объектов на форму, изменение их свойств	44
3.2.2. Компоновка объектов.....	45
3.2.3. Порядок обхода элементов	47
3.2.4. Сигналы и слоты. Предпросмотр	48
3.2.5. Связь класса главного окна и его формы	50
3.2.6. Интернационализация приложения	51
4. Заключение.....	54
Список литературы.....	55
ПРИЛОЖЕНИЕ.....	56

1. Общие положения

1.1. Цель лабораторных работ

Изучение инструментальной системы программирования Qt (Qt SDK, Qt Creator) и разработка на ее основе учебных и демонстрационных приложений с графическим интерфейсом пользователя, посвященных изучению ключевых вопросов технологии объектно-ориентированного анализа, проектирования и программирования.

1.2. Темы лабораторных работ

1. Изучение инструментальной системы объектно-ориентированного программирования Qt (Qt 4.x с использованием редактора QtCreator 2.x) и ее применение для визуальной разработки приложения с дружественным графическим интерфейсом пользователя.

2. Разработка класса, представляющего заданную предметную область, и интерактивного приложения, демонстрирующего работу методов класса.

3. Разработка иерархии абстрактных и конкретных классов, представляющих заданную предметную область, и интерактивного приложения, демонстрирующего работу механизмов наследования, полиморфизма и перегруженных операторов в разработанной иерархии классов.

4. Вывод объектов и списка объектов класса в файл и их ввод из файла с применением классов файловых потоков. Использование шаблонов классов. Обработка исключений.

Лабораторная работа № 1 подробно описана ниже. При ее выполнении рекомендуется использовать дополнительную литературу [1–5]. Индивидуальные задания к лабораторным работам № 2, 3 и 4 и детальные указания по их выполнению выдаются студентам во время занятий. При выполнении этих лабораторных работ следует использовать учебную литературу [6–8] и конспект лекций.

1.3. Система программирования Qt

Qt – это инструментальная система визуального объектно-ориентированного программирования для быстрой разработки приложений любой сложности. Qt предоставляет набор библиотек, средства разработки, тестирования и отладки приложений, включая большую библиотеку компонентов, комплект инструментов проектирования, шаблоны приложений и форм, документацию. Эти средства упрощают создание прототипа и сокращают время разработки программы. Qt позволяет запускать написанное ПО в большинстве современных операционных систем (ОС) путем компиляции проектов для каждой ОС без изменения исходного кода. Включает в себя все основные средства, которые могут потребоваться при разработке прикладного программного обеспечения.

2. Обзор среды разработки

2.1. Знакомство с Qt, сигналы и слоты

Процесс установки Qt SDK (открытой бесплатной версии) на конкретную ОС можно найти в литературе [1-5].

Прежде чем переходить к запуску Qt Creator, рекомендуется ознакомиться с основами программирования в Qt без использования графического редактора ([2], главы 1-2), или ([3], глава 1) или на вводной лекции. Здесь приведены наиболее важные сведения.

Отличительная особенность Qt состоит в использовании предварительной системы обработки исходного кода (Meta Object Compiler, МОС). МОС является посредником между некоторыми особыми возможностями библиотеки Qt и стандартными компиляторами C++. МОС позволяет вводить в язык C++ новые понятия и принципы, например, слоты и сигналы. Препроцессор МОС ищет в заголовочных файлах на C++ описания классов, содержащие макрос `Q_OBJECT`, и создает дополнительный исходный файл на C++, содержащий метаобъектный код, который доступен для компиляторов.

В Qt используется объектная модель, для этого вводится базовый для многих классов класс *QObject*. Важной особенностью класса *QObject* является возможность использования механизма сигналов и слотов. Принципы ООП выдаются на лекциях, поэтому здесь не приводятся. Однако, механизм сигналов и слотов является отличием Qt, которое делает язык C++ еще более подходящим под ООП. Причины возникновения этого механизма и его преимущества обстоятельно описаны в [2, глава 2]. Возможно, этот механизм лучше соответствует требованиям ООП, чем механизм функций обратного вызова, используемый в Windows API.

Пример: есть экранная форма с кнопками и полями вывода и т.д. и есть некоторые действия (логика работы), вызываемые формой (расчеты, соединения с БД, чтение файлов и т.п.). Очевидно, что по принципам ООП форма должна быть в одном классе, объекты, реализующие логику, в других.

При подходе без сигналов/слотов пришлось бы, например, включить рабочий класс как член класса формы, чтобы в коде обработчика кнопки вызывать его методы. Но если и рабочий класс должен иметь доступ к форме, например, к полям вывода? В его методах можно было бы использовать закрытые члены класса формы. В любом случае два класса оказались бы

слишком тесно переплетены, то есть связь между ними была бы намного сильнее, чем это следовало бы из логики их взаимосвязи. А если код рабочего класса и класса формы пишут разные программисты? Если нужно заменить эту форму на другую или внести в нее другую функциональность? Все это было бы проблематично сделать из-за сильной связи классов. Также сигналы и слоты полезны в случае использования для какого-либо объекта отдельного потока.

Поэтому и введен механизм сигналов и слотов, который базируется на принципе ООП «Объекты взаимодействуют, посылая и получая сообщения». Сигнал – это посылка сообщения, а слот – это приемник сигналов. Звонок телефона – сигнал, способность человека услышать звонок – это его слот. Сигнал можно подключить к различному числу слотов разных объектов, причем объект, высылающий сигнал, ничего не знает о том, кто его получит – об этом знает только разработчик, соединяющий сигналы одних объектов со слотами других. Объект, слот которого получил сигнал, может узнать отправителя через метод *sender()*, возвращающий указатель на объект-отправителя.

Такое соединение будет работать немного медленнее, чем вызов функций, но для большинства некритичных к времени выполнения приложений эта разница не имеет значения. В остальном слоты очень похожи на обычные методы, кроме того, что они не могут содержать параметров по умолчанию и являться статическими, последнее связано с необходимостью соединять сигнал со слотом конкретного созданного объекта.

Основные свойства соединений сигналов и слотов:

- к сигналу можно подключать несколько слотов;
- слот можно подключать к нескольким сигналам;
- сигнал можно подключить к другому сигналу (в этом случае при генерации первого сигнала будет сгенерирован и второй, но не наоборот);
- связь можно установить и разорвать в любое время;
- параметры сигнала и слота должны задаваться в одинаковом порядке и иметь одинаковый тип, если у сигнала больше параметров, чем у слота, дополнительные параметры будут игнорироваться;
- при уничтожении объекта все связанные с ним соединения будут уничтожены автоматически.

Полностью избавиться от лишних связей механизм позволяет не во всех случаях (иногда, например, все же приходится использовать указатель на форму), но значительно их

сокращает. В ходе выполнения первой лабораторной работы этот механизм будет разобран и применен.

Рассмотрим простой пример: есть объект, посылающий сигналы, несущие некоторую информацию в виде целого числа, и объект, которые эти сигналы получает. Необходимо реализовать классы для отправки и приема сигналов, так чтобы реализация одного класса не зависела бы от другого, а объекты классов не были бы связаны ничем, кроме соединения сигналов и слотов.

Код класса источника сигналов:

```
class MySender : public QObject {
    Q_OBJECT//макрос Q_OBJECT необходим для классов, содержащих
    сигналы и слоты
public:
    void sendSignal(int x);
signals:
    void signal1(const int);
};
void MySender:: sendSignal(int x) {
    emit signal1(x);//emit означает "выслать сигнал"
```

код класс приемника:

```
class MyReceiver : public QObject { Q_OBJECT
public slots:
    void slot1(int x);
};
void MyReceiver::slot1(int x) {
    qDebug()<<x<<sender()->objectName();
    //qDebug() позволяет выполнять вывод сообщений в
    терминал при отладке
}
```

последним приведен код функции для демонстрации связи.

```
int main()
{
    MySender *sender = new MySender();
    MyReceiver *receiver = new MyReceiver();
    sender->setObjectName("sender");
    sender->sendSignal(10);
    //сигнал не будет получен, так как еще нет соединения
    QObject::connect(sender,SIGNAL(signal1(int)),
                     receiver,SLOT(slot1(int)));
    sender->sendSignal(100);
    //в терминале будет выведено 100 "sender1"
    QObject::disconnect(sender,SIGNAL(signal1(int)),
                        receiver,SLOT(slot1(int)));
    sender->sendSignal(100);
    // сигнал не будет получен, так как уже нет соединения
    return 0;
}
```

В коде классов источника и приемника нет никаких взаимных связей, связь между объектами класса устанавливается так же не внутри их реализаций. К одному объекту источнику можно подключить несколько объектов приемников, один приемник может быть соединен с несколькими источниками. Реализация сигналов никогда не прописывается в коде, ее берет на себя МОС.

2.2. Запуск системы Qt Creator

Вы можете запустить Qt Creator одним из следующих способов.

- на рабочем столе дважды щелкните значок Qt Creator;
- выберите пункт меню *Пуск* → *Программы* → *Qt Creator by Nokia* → *Qt Creator*;
- выберите пункт меню *Пуск* → *Выполнить*, а затем введите qt creator;
- в папке Qt\qtcreator-x.x.x\bin или QtSDK\QtCreator\bin дважды щелкните файл qtcreeator.exe.

В следующих разделах приводится руководство по системе Qt Creator, упрощающей написание приложений. Руководство основано на материалах ресурса Crossplatform.ru.

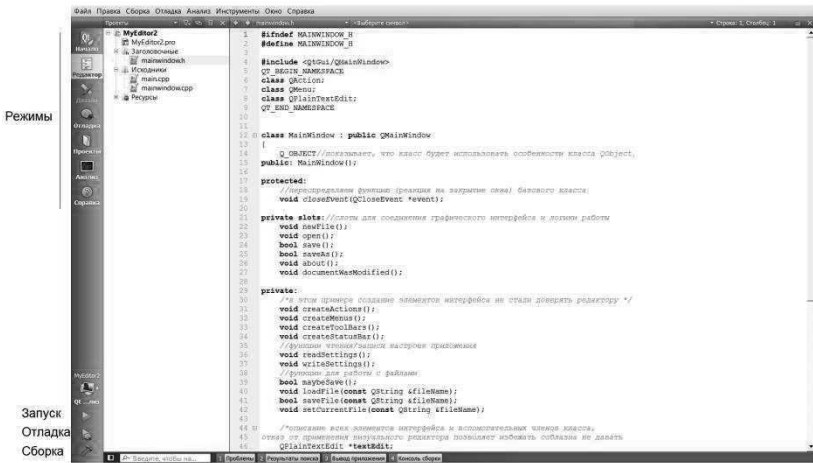
Qt Creator включает большое число полезных особенностей:

- текстовый редактор предоставляет навигацию, поддержку синтаксиса и дополнения кода;
- мастер генерации проектов позволяет создавать различные виды проектов, используя шаблоны;
- документация интегрирована в Qt Creator, в процесс работы можно быстро перейти к справке;
- формы графического интерфейса пользователя можно быстро проектировать благодаря инструменту Qt Designer;
- приложения можно отлаживать с использованием графического интерфейса к GNU symbolic debugger (GDB) и Microsoft Console Debugger (CDB);
- Qt SDK включает в себя утилиты для быстрой локализации приложений.

2.3. Краткий обзор Qt Creator

В данном обзоре использованы материалы [1]. На рисунке 2.1 показан снимок экрана Qt Creator в режиме *Правка*.

Замечание. В различных версиях *Qt Creator* могут быть изменения в расположении и названии элементов, структуре меню и параметрах. Поэтому приведенные скриншоты и описания могут не во всем соответствовать вашей версии, хотя принципиальных отличий в версиях с одинаковым старшим числом нет (в данном руководстве используются *Qt 4* и *Qt Creator 2*).



Панели Вывода
Рисунок 2.1 – Окно Qt Creator

В *Qt Creator* есть несколько режимов работы: *Начало*, *Редактор (Правка)*, *Дизайн*, *Отладка*, *Проекты*, *Анализ (Вывод)* и *Справка*.

Начало. Отображает экран приветствия, позволяя быстро загружать недавние сессии или отдельные проекты. Здесь же можно перейти к демонстрационным программам и примерам.

Редактор. Позволяет редактировать файлы проекта и исходных кодов. Боковая панель слева предоставляет различные виды для перемещения в проекте: файловая система, иерархия файлов, входящих в проект, обзор классов и т.д.

Дизайн. Режим для визуальной работы с формами графического пользовательского интерфейса.

Отладка. Инструменты и настройки для отладки приложений.

Проекты. Позволяет настроить этапы сборки и запуска проектов, включая выбор компилятора, типа сборки и прочее.

Анализ. Инструмент для отладки.

Справка. Показывает всю документацию, по библиотекам Qt и Qt Creator.

2.3.1 Панели вывода

Панель задач в Qt Creator может отображать одну из четырех различных панелей: **Вывод сборки (Проблемы)**, **Результаты поиска**, **Вывод приложения** и **Вывод компилятора (Консоль сборки)**. Эти панели доступны во всех режимах.

Панель **Вывод сборки** (рисунок 2.2) предоставляет сообщения об ошибках или предупреждениях. Незначительные сообщения компилятора могут быть пропущены, кроме того панель упорядочивает вывод сообщений.

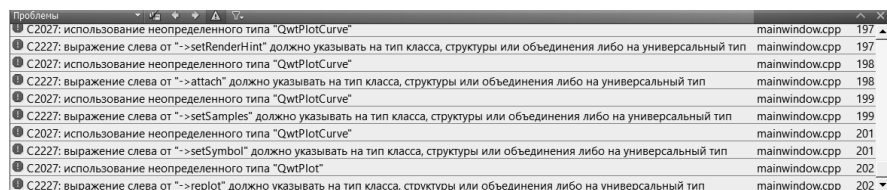


Рисунок 2.2 – Вывод сборки

Панель **Результаты поиска** (рисунок 2.3) отображает результаты поиска в файле, каталоге или проекте.

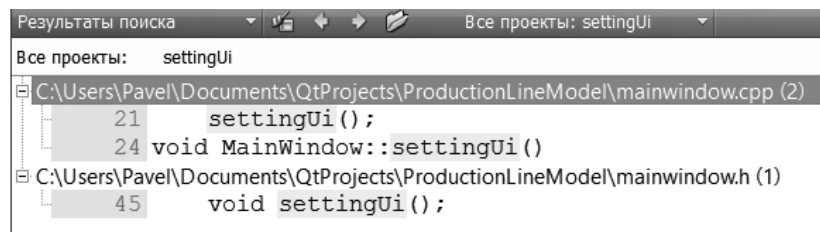


Рисунок 2.3 – Результаты поиска

Панель **Вывод приложения** (рисунок 2.4) показывает статус программы при ее выполнении и отладочную информацию.