Примечание к лабораторным работам

Помимо написания кода в файле *.pro и соответствующего создания и вызова интерфейсов окна можно воспользоваться мастером создания проектов.

Для этого необходимо запустить QtCreator – интегрированную среду разработки (IDE – Integrated Development Environment) (главное окно представлено на рис. 1). Данная IDE включает в себя следующие компоненты: мастер проектов, текстовый редактор, компилятор, компоновщик, отладчик, проектный обозреватель, библиотеки Qt, Qt Designer, систему помощи.

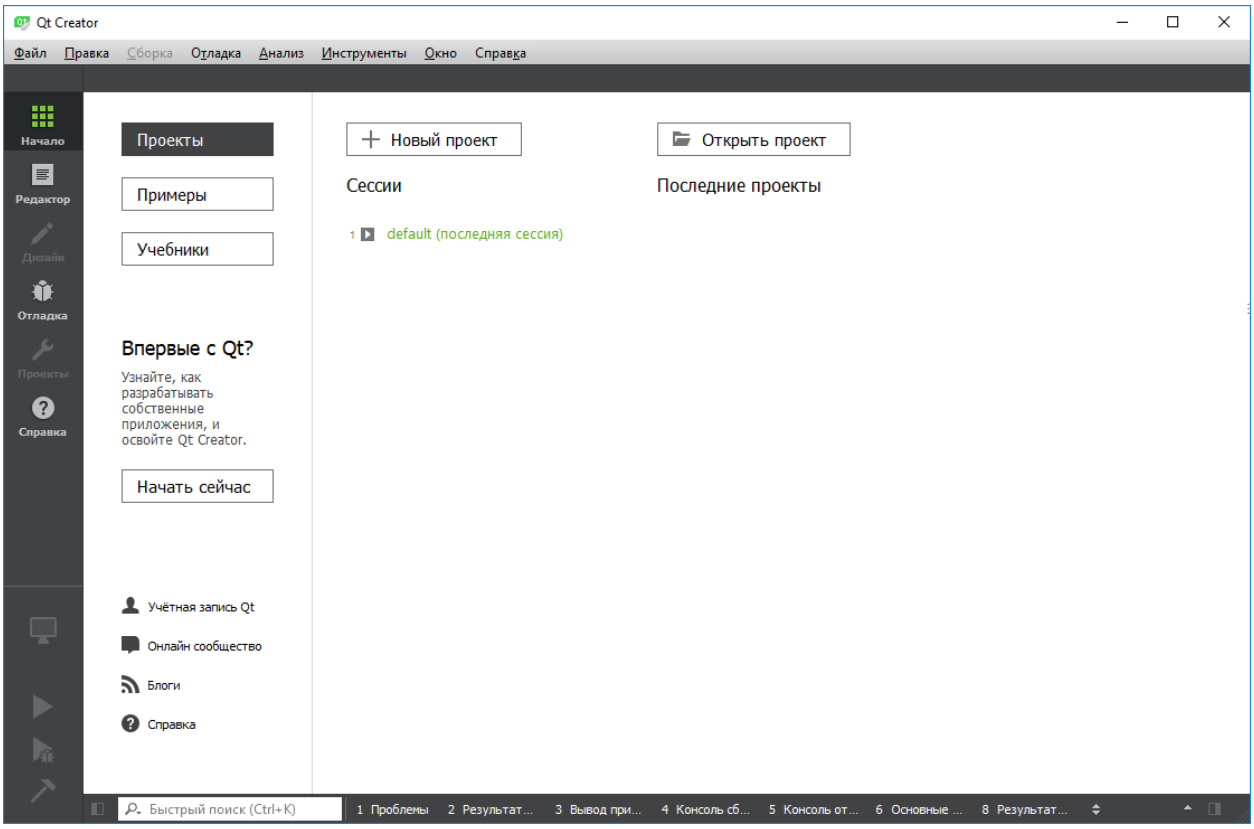

Рисунок 1. Главное окно интегрированной среды разработки QtCreator

Для создания проекта нужно сделать следующее:

1) В меню «Файл» выберите команду «Создать файл или проект» и в открывшемся диалоговом окне (рис. 2) выберите тип проекта «Приложение», а затем одну из следующих альтернатив:

- Приложение Qt Widgets
- Консольное приложение Qt

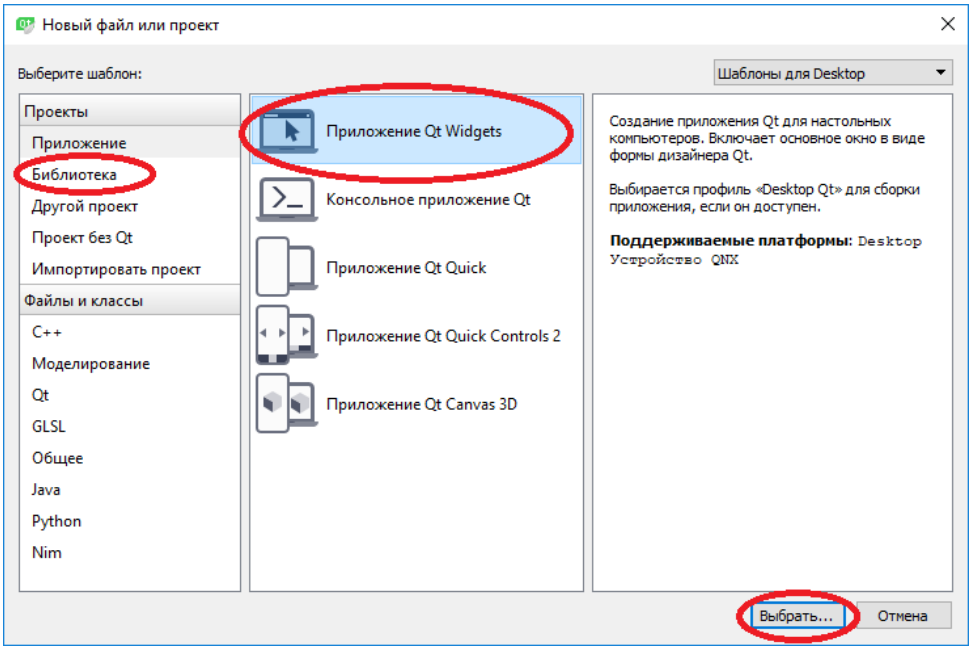

Рисунок 2. Окно создания нового проекта

Выберем вариант Qt Widgets для создания приложения с графическим интерфейсом.

В следующем окне, показанном на рис. 3, зададим в текстовом поле ввода «Название» имя для своего проекта, например MyApp, и Qt Creator присвоит это имя новому проекту.

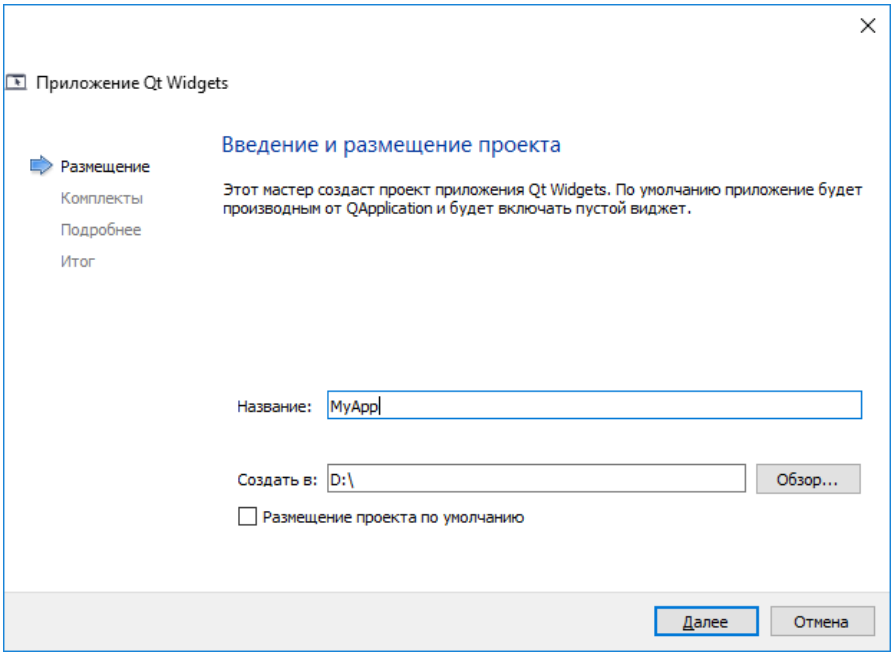

Рисунок 3. Окно ввода имени проекта

ВАЖНО!!! При создании проекта нельзя указывать в имени и пути русские буквы, а также рекомендуется чтобы не было пробелов в пути к проекту.

После нажатия на кнопку «Далее» нужно указать для каких комплектов будет создаваться данное приложение (рис. 4). Комплект – набор библиотек, настроек компилятора и других специализированных программ, которые позволяют создать приложение для разных платформ (Desktop, Android, встраиваемые системы и т.д.). Выберем комплект для создания обыкновенного 32bit приложения для Windows с нужным нам компилятором.

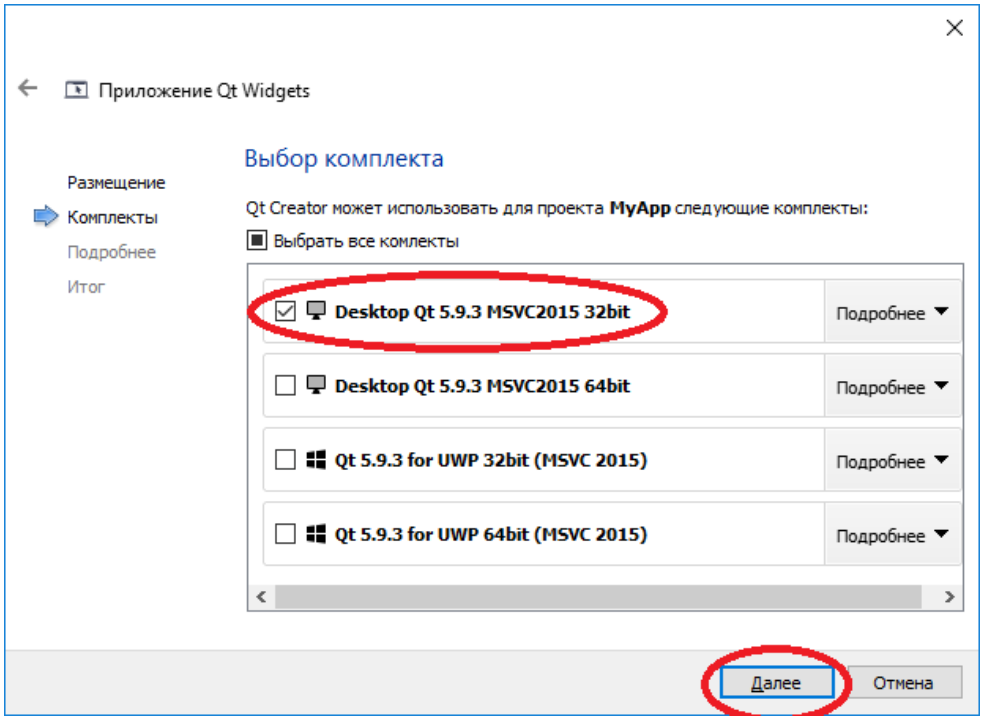

Рисунок 4. Окно выбора комплекта

В следующем окне укажем основные названия наших классов (рис. 5). Обычно класс с главной формой называют MainWindow (который наследуется от QMainWindow). Для продолжения необходимо нажать «Далее».

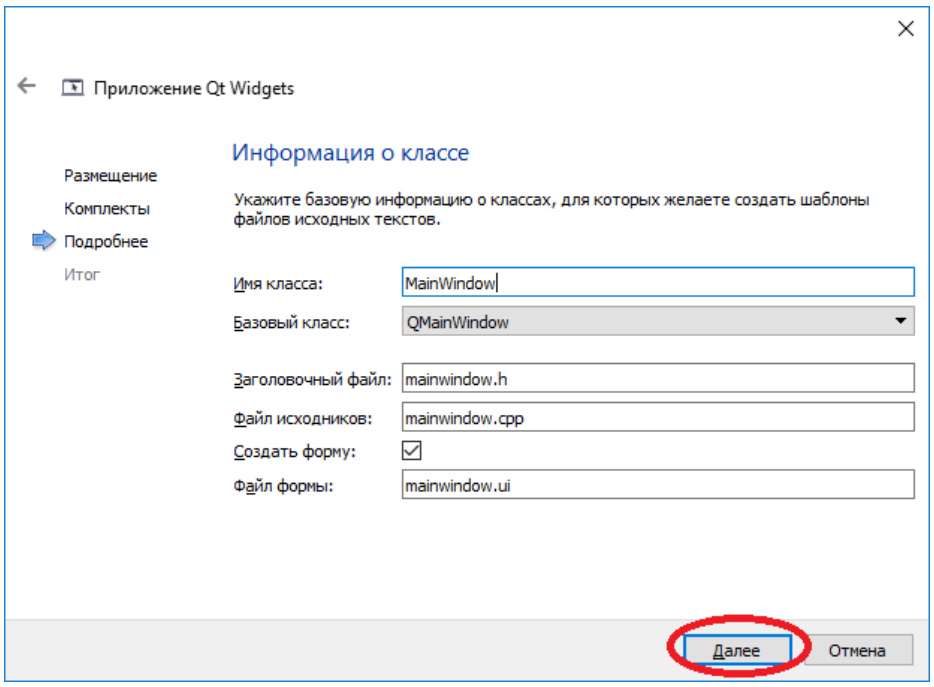

Рисунок 5. Окно именования класса

В следующем окне можно просмотреть список создаваемых файлов и можно указать систему контроля версий (например, Git). Для окончания нажмем «Завершить» (рис. 6).

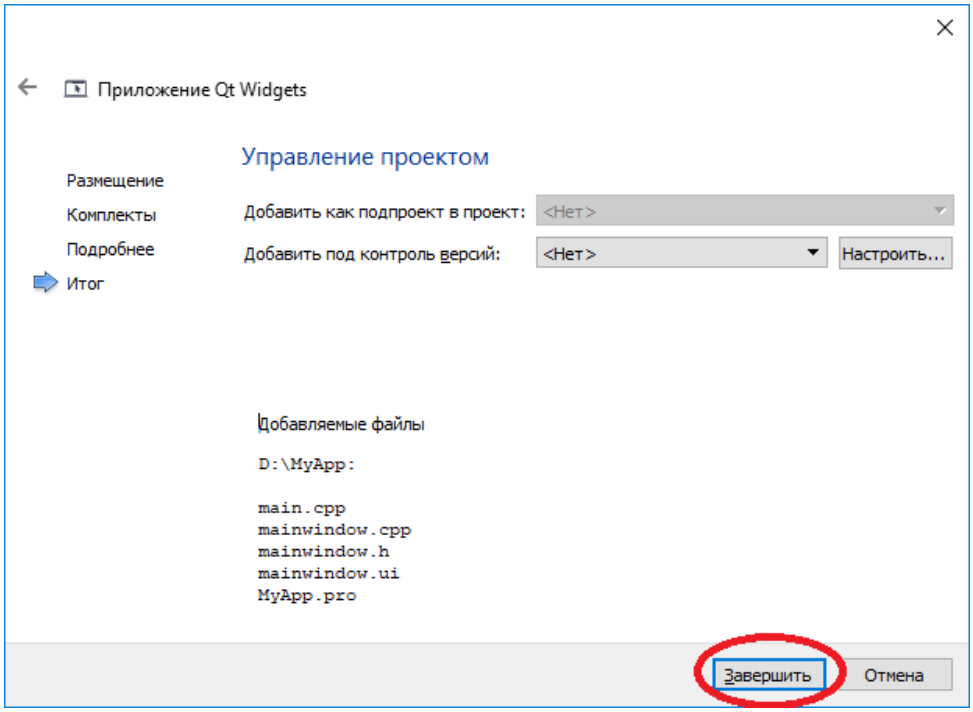

Рисунок 6. Окно для управления проектом

В результате мы создали простое приложение с одной главной диалоговой формой. Наше проект содержит следующие файлы:

- MyApp.pro – файл настроек проекта (в случае ручного создания его надо описать);
- main.cpp – файл, содержащий функцию main (обычно его редактировать не требуется);
- mainwindow.ui – файл формы (там будут располагаться другие визуальные элементы – кнопки, поля вводы и т.д.);
- mainwindow.h – содержит объявление класса MainWindow;
- mainwindow.cpp – содержит реализацию класса MainWindow.

На два последних файла обратим дополнительное внимание.

mainwindow.h содержит следующее:

```
#ifndef MAINWINDOW_H // Данная препроцессорная обёртка предотвращает попытку
#define MAINWINDOW_H // многократного включения заголовочных файлов

#include <QMainWindow> //Подключаем класс родителя

namespace Ui {          //Подключаем форму, которую будем использовать
class MainWindow;
}

//Объявление класса
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0); //Конструктор
    ~MainWindow(); //Деструктор

private:
    Ui::MainWindow *ui; //Форма
};

#endif // MAINWINDOW_H
```

Здесь можно добавлять любые атрибуты класса MainWindow, любые методы, которые затем можно реализовать в файле *.cpp

mainwindow.cpp содержит следующее:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

//Реализация конструктора
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
```

```
    ui(new Ui::MainWindow)
{
    //ui - форма, которая будет содержать другие элементы
    ui->setupUi(this);
}

//Деструктор
MainWindow::~MainWindow()
{
    delete ui;
}
```

Здесь можно добавить реализацию собственных методов класса MainWindow.

Пример.

Создадим на форме кнопку и поле ввода текста. При нажатии на кнопку заголовок окна должен меняться на тот текст, который написан в поле ввода.

Для этого перейдем в файл формы (рис. 7).

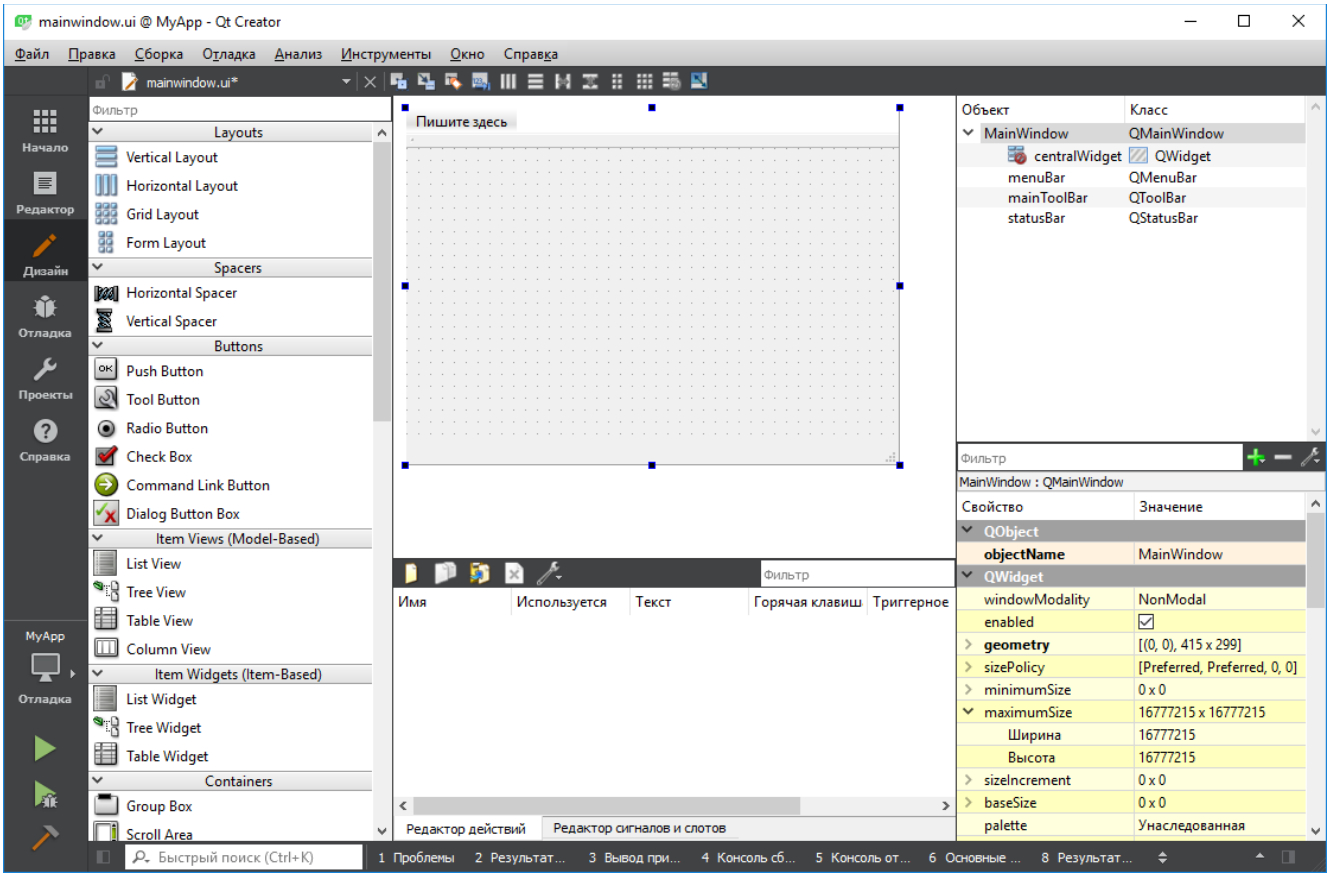

Рисунок 7. Окно дизайнера формы

Добавим на форму два элемента: QPushButton – кнопку и QLineEdit – поле ввода. Разместим их на форме и должно получиться что-то подобное (рис. 8).

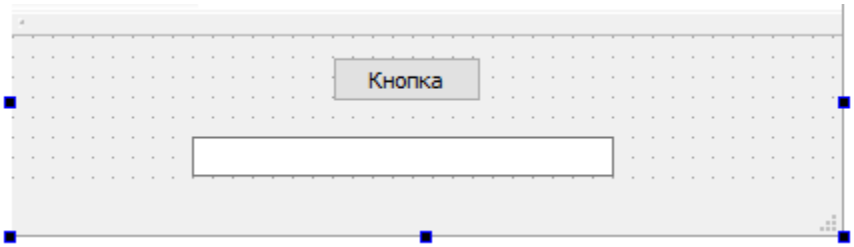

Рисунок 8. Шаблон формы

Для того, чтобы обработать событие нажатия на кнопку нажмем правой кнопкой мыши на «Кнопку» и выберем «Перейти к слоту…». В появившемся окне выберем пункт «clicked()» (рис. 9).

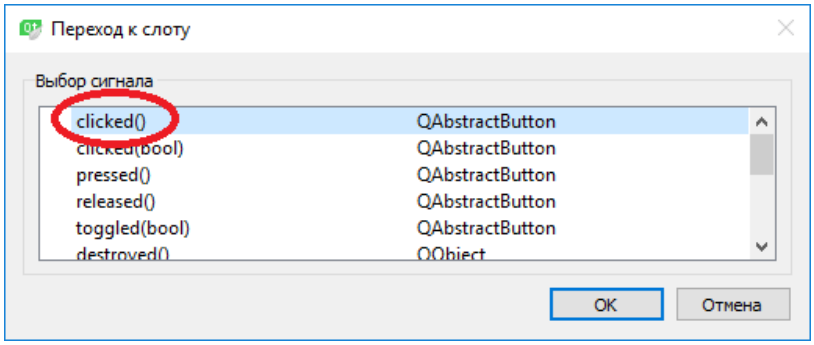

Рисунок 9. Окно перехода к слоту

После нажатия на «Ок» нас перенесет в файл mainwindow.cpp в метод void MainWindow::on_pushButton_clicked()

Внутри мы и напишем наш код:

```
QString text = ui->lineEdit->text();
this->setWindowTitle(text);
```

В первой строке мы создали переменную text, которая хранит текст, введенный пользователем. Для того, чтобы получить его мы обратились к атрибуту lineEdit формы ui (ui->lineEdit), у которого затем вызвали метод text().

Во второй строке мы обратились к главному окну приложения (this->) и вызвали метод setWindowTitle и передали ему в качестве параметра переменную text.

Для запуска нажмем комбинацию Ctrl+R или F5. В результате получим окно (рис. 10).

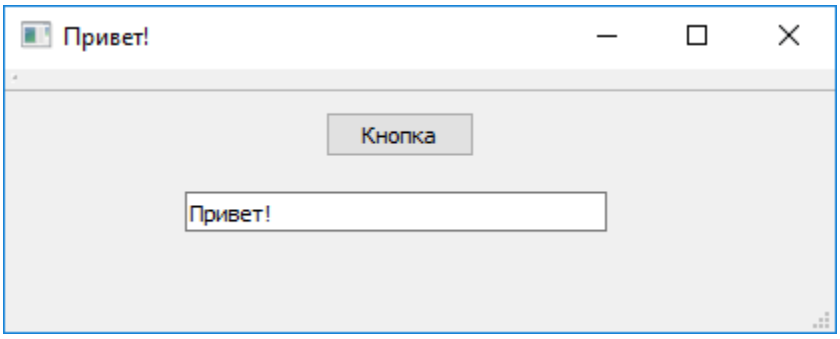

Рисунок 10. Окно нашего приложения

Таким образом, важно отметить что существует два способа создания проектов: ручной с самостоятельным описанием всех файлов и классов и автоматический с помощью мастера проектов. В дальнейшем Вы можете использовать любой.