

Семинар 2 (часть 3)

Создание графичеких интерфейсов

Создание формы в QtDesigner и использование её в проекте

Qt Creator. Создание GUI.

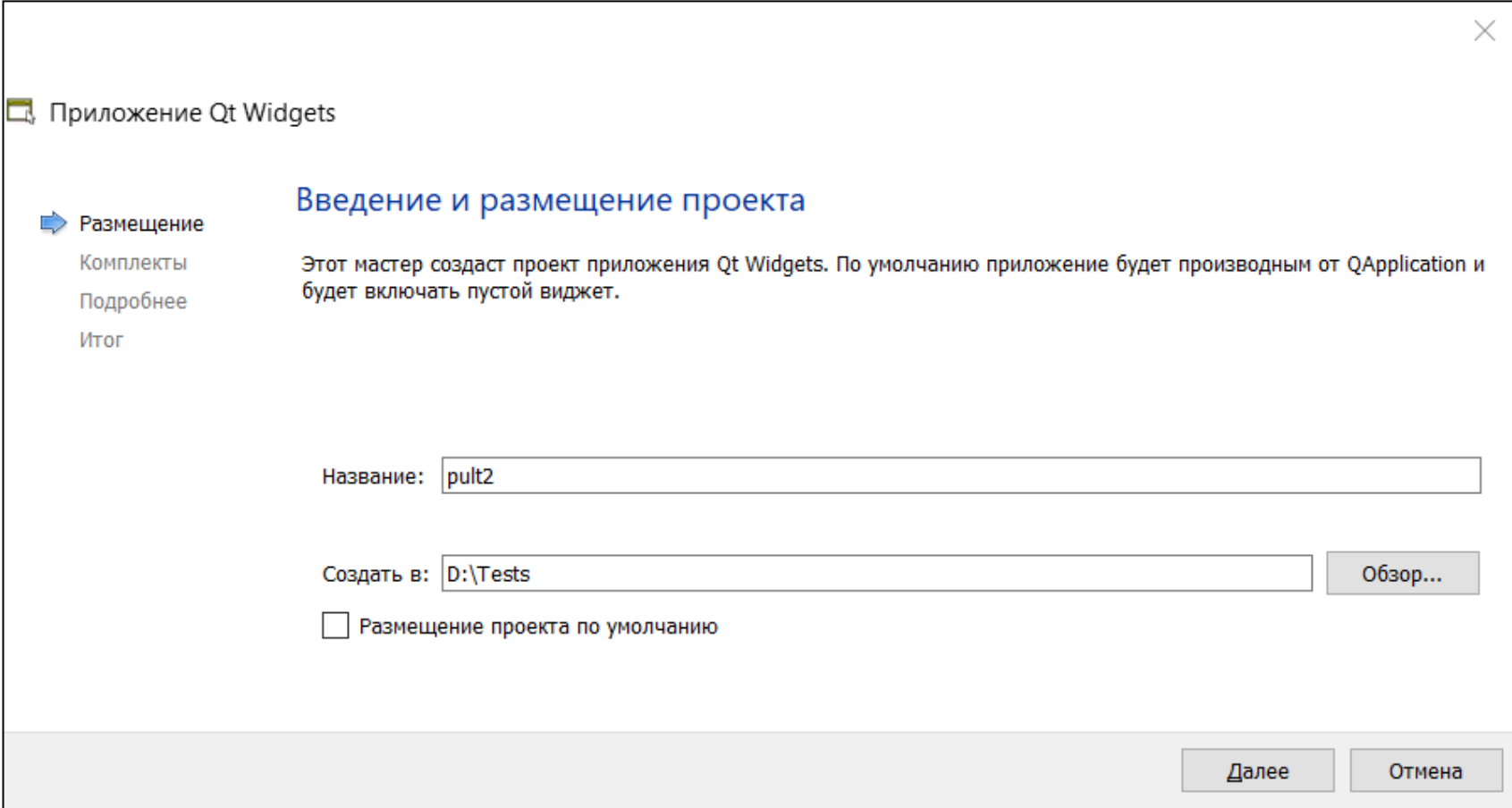
Используя второй подход можно создать «форму» в графическом редакторе Qt Designer.

Файлы формы имеют расширение .ui.

Qt Designer позволяет редактировать файлы форм, содержащих настройки вида виджетов.

Qt Designer можно использовать как отдельную программу или воспользоваться интеграцией с оболочкой Qt Creator — редактором форм.

Практическая часть. Создадим новый проект!



Приложение Qt Widgets

Размещение
Комплекты
Подробнее
Итог

Введение и размещение проекта

Этот мастер создаст проект приложения Qt Widgets. По умолчанию приложение будет производным от QApplication и будет включать пустой виджет.

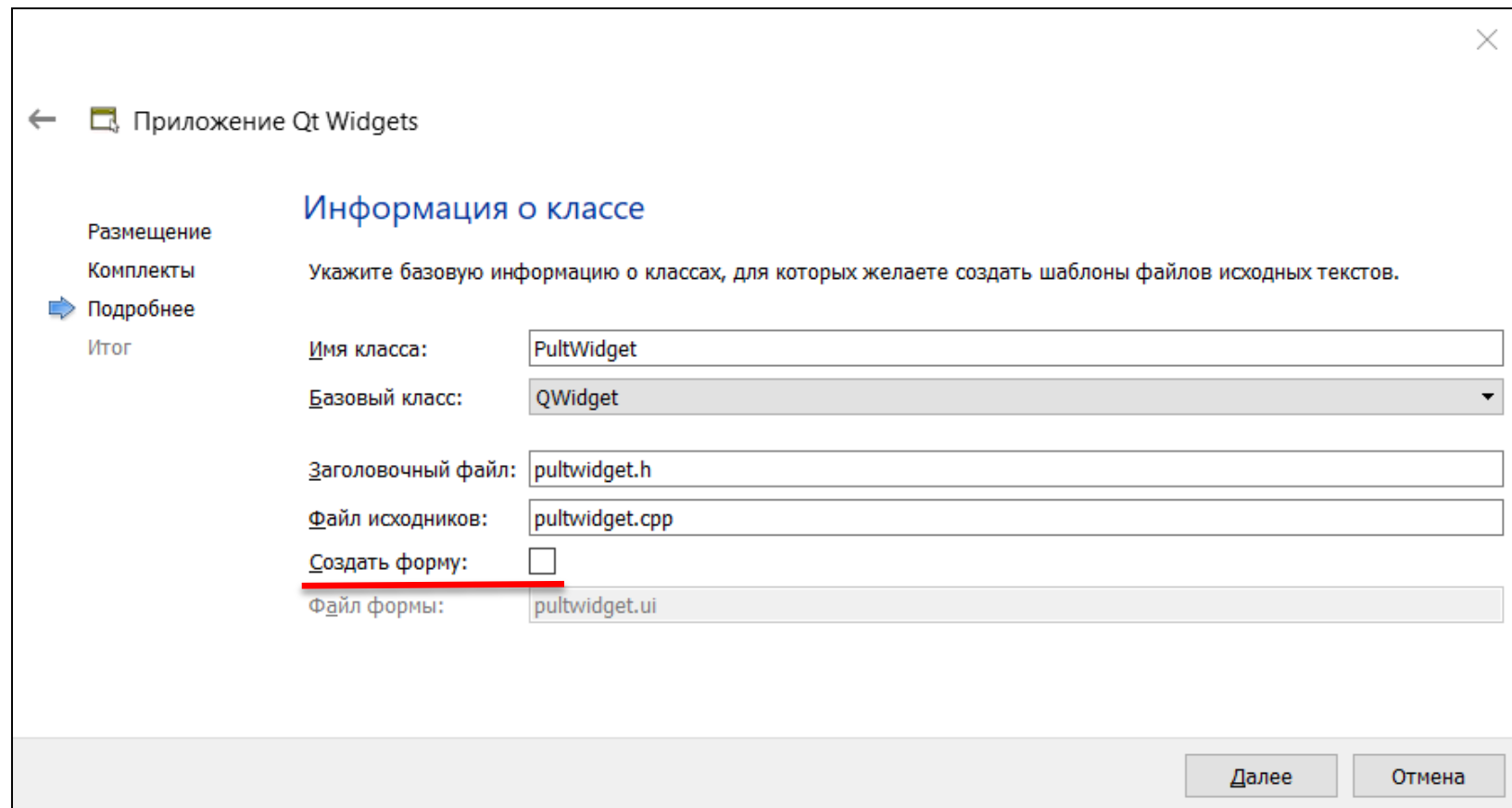
Название:

Создать в:

☐ Размещение проекта по умолчанию

Создание проекта. Шаг 2.

Убираем галочку «Создать форму». Так как форму будем добавлять вручную!



← Приложение Qt Widgets

Размещение
Комплекты
➔ Подробнее
Итог

Информация о классе

Укажите базовую информацию о классах, для которых желаете создать шаблоны файлов исходных текстов.

Имя класса: PultWidget

Базовый класс: QWidget

Заголовочный файл: pultwidget.h

Файл исходников: pultwidget.cpp

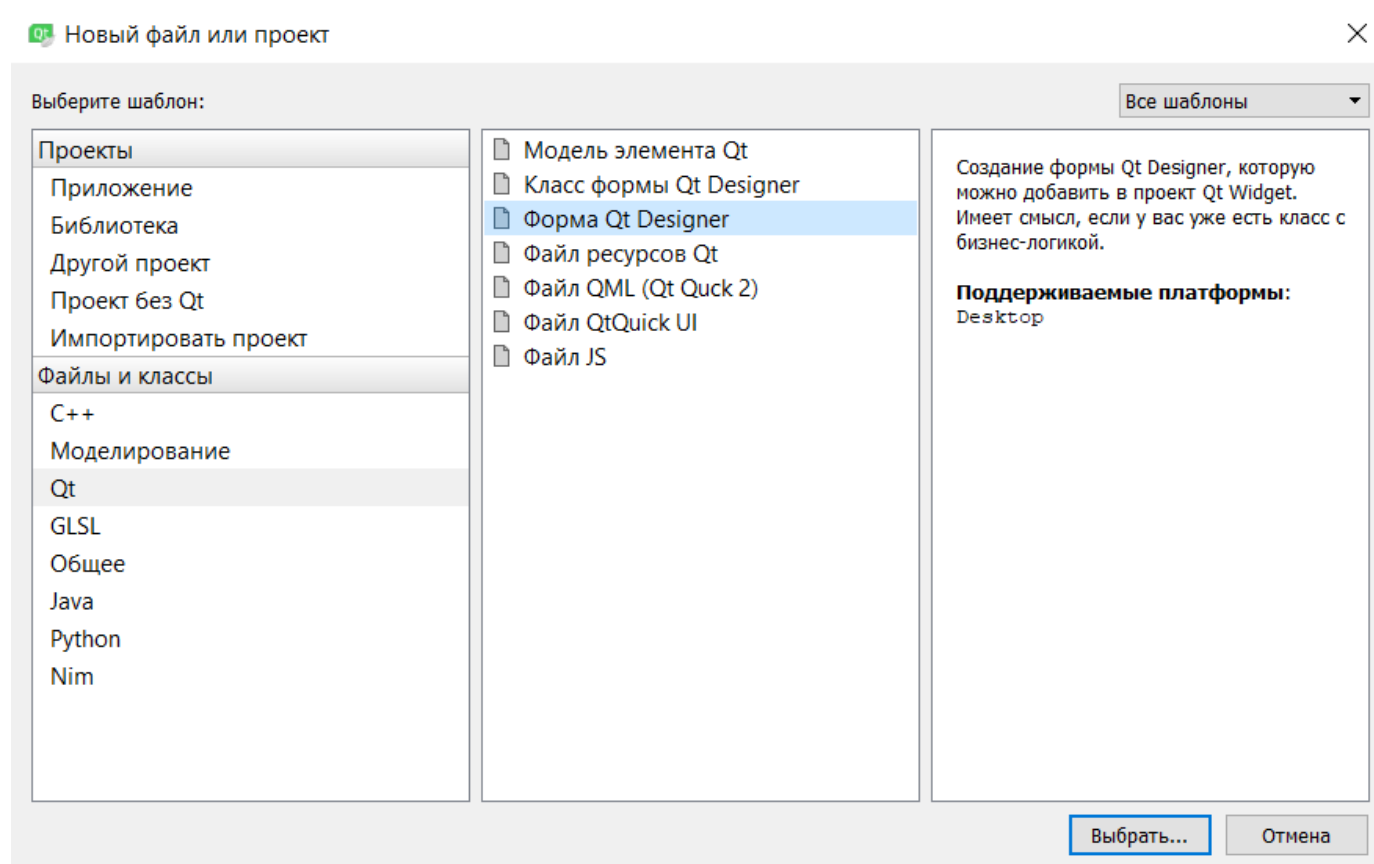
Создать форму: ☐

Файл формы: pultwidget.ui

Далее Отмена

Добавление в проект формы

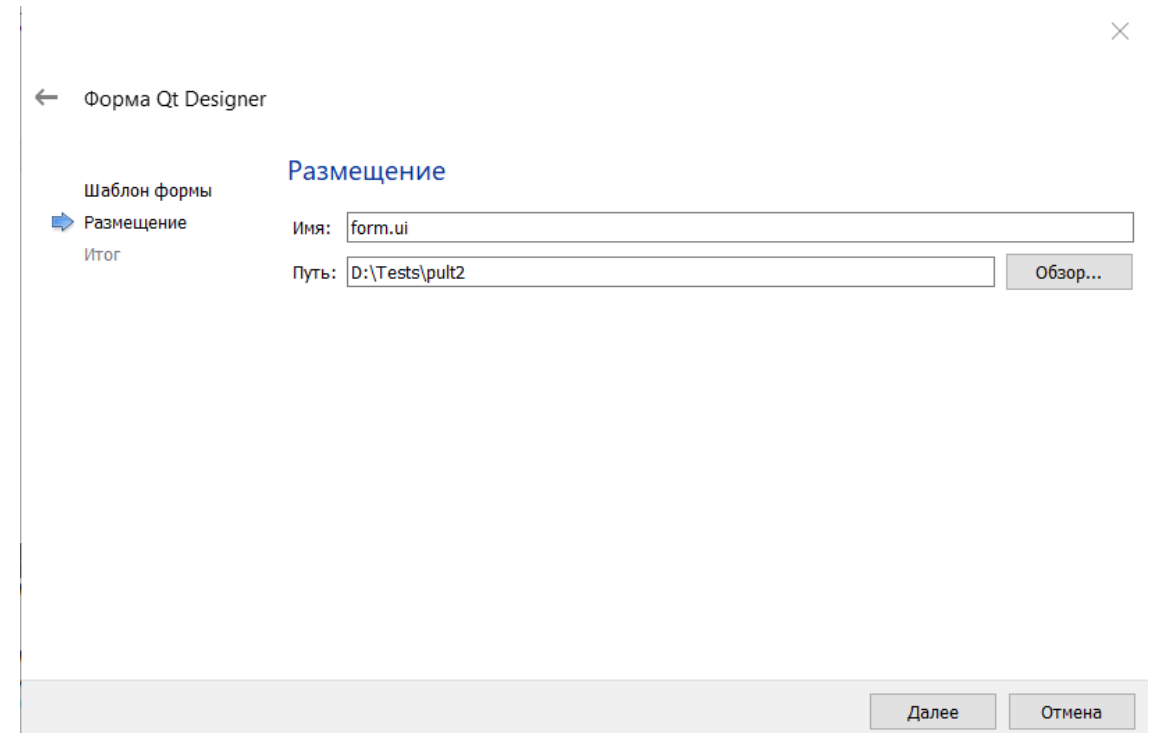
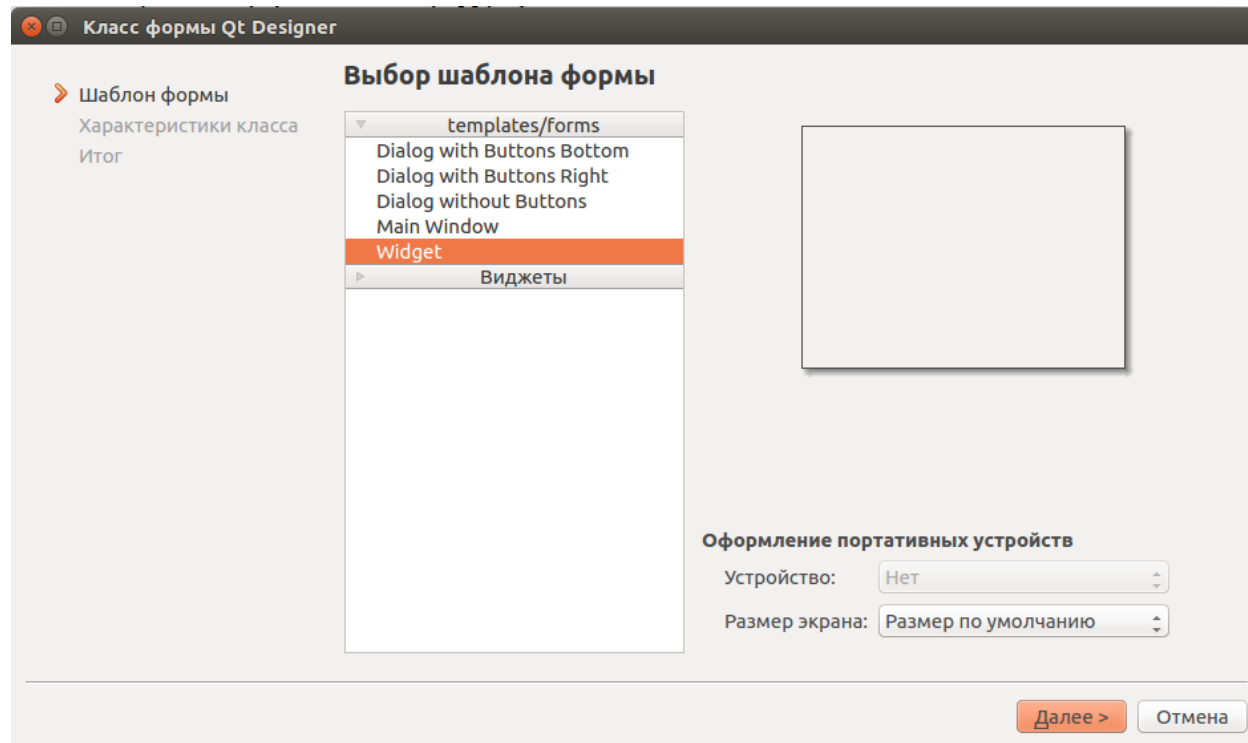
Создадим форму главного окна. Файл->Новый файл или проект. Выбираем «Файлы и классы Qt»-> Класс формы Qt Designer.



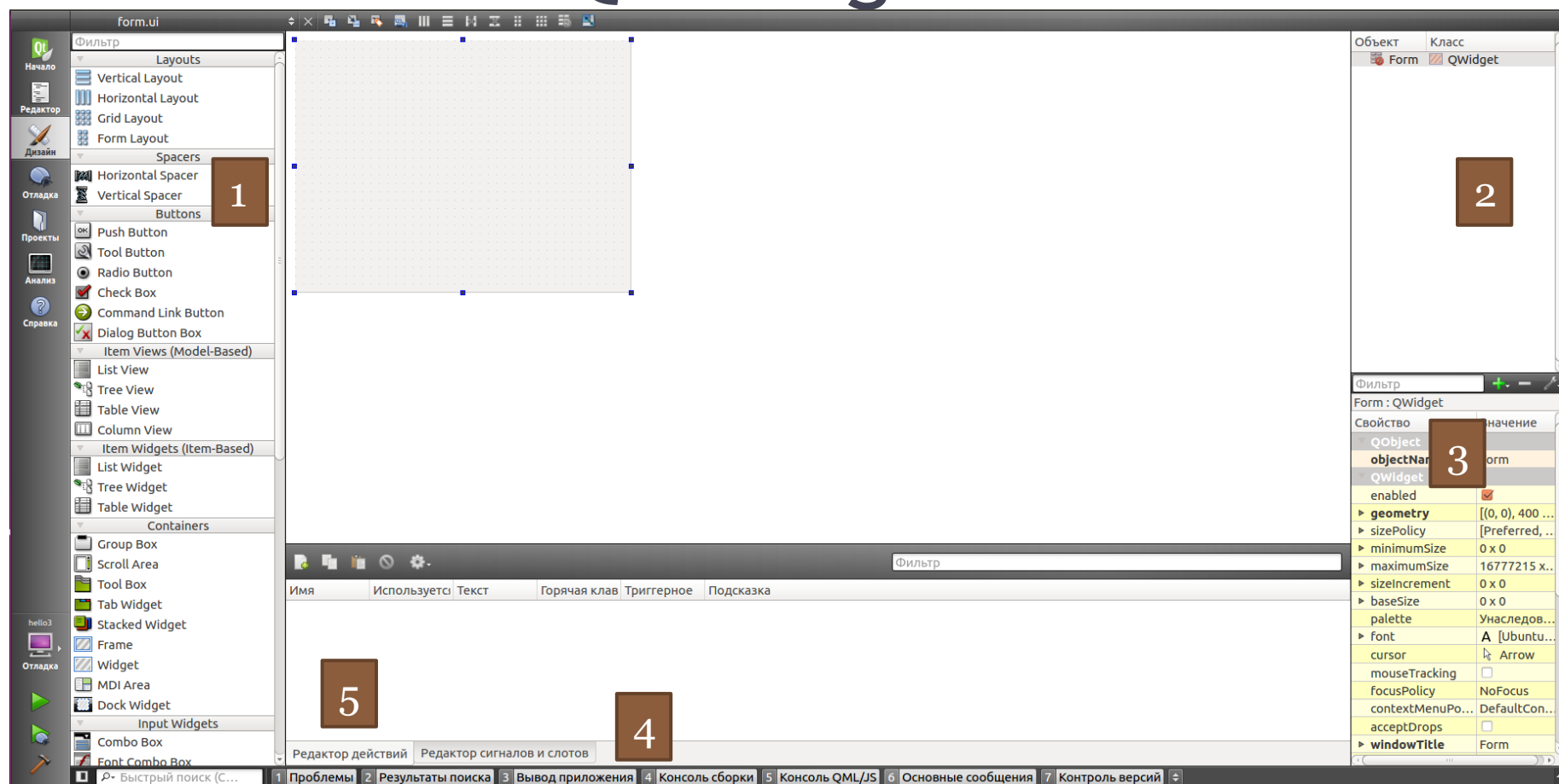
Qt Creator. Создание GUI.

Qt Designer предлагает на выбор различные шаблоны формы. Выберем Widget.

Назовем форму «form.ui»

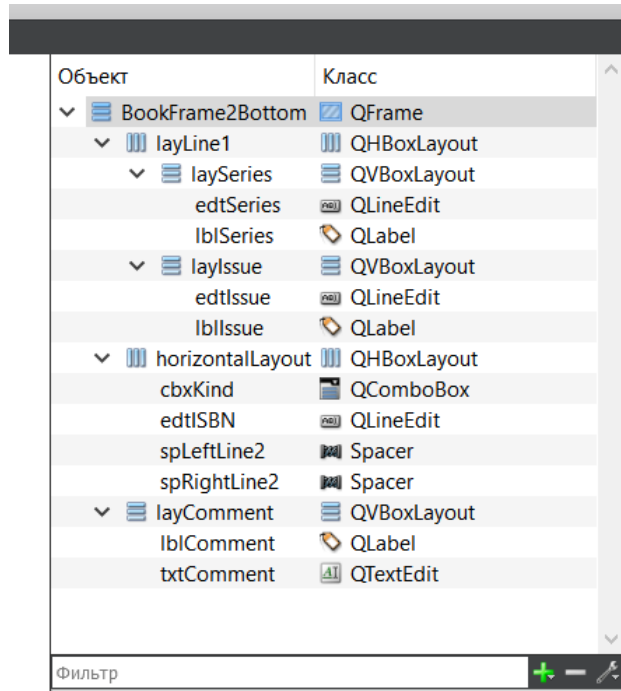


Qt Designer

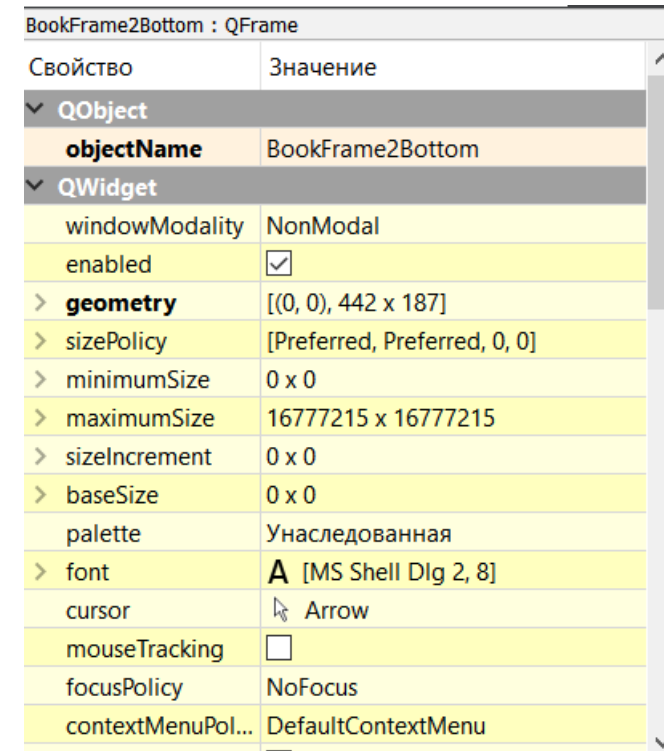


(1) – Окно «Виджеты», в котором можно найти объекты компоновки и сами виджеты, сгруппированные в отдельные категории. Из этого окна посредством «перетаскивания» виджеты добавляются на форму.

Qt Designer

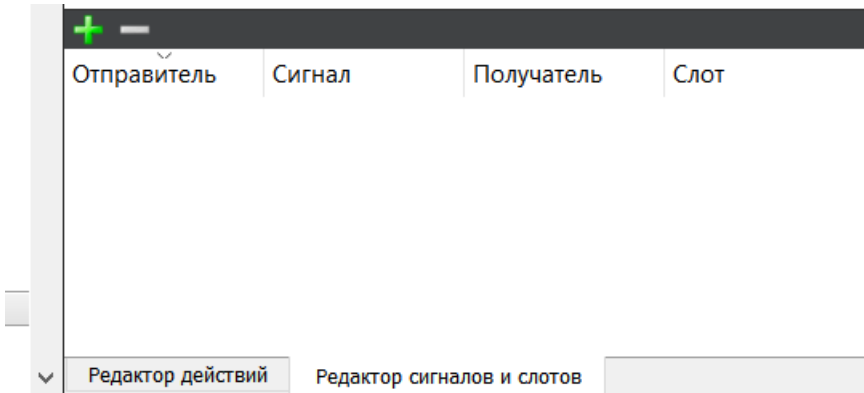


(2) – Окно «Объекты» - содержит список используемых виджетов. В этом окне их можно выбирать для последующего изменения. Например, при помощи «Редактора свойств» (3) изменять их свойства;

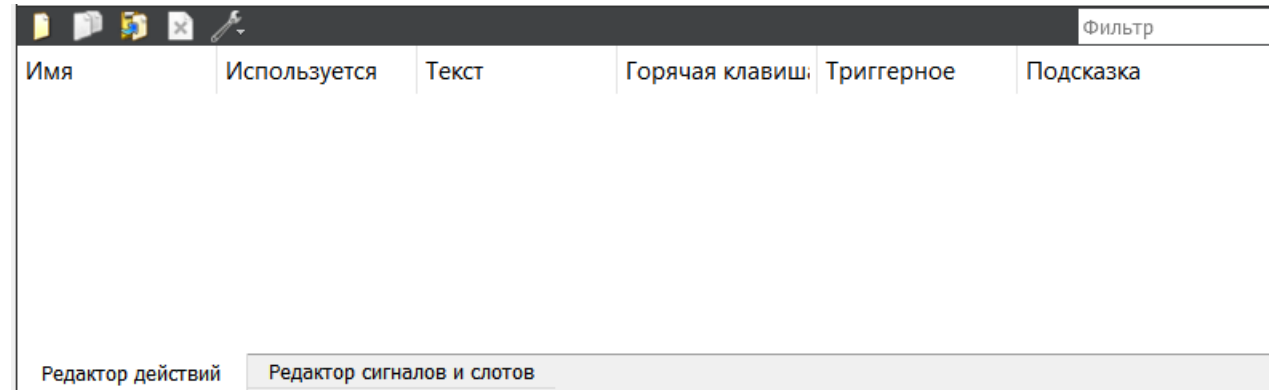


(3) – Окно «Редактор свойств» - определяет ряд свойств выбранного виджета. Это могут быть как цвет фона, шрифт, максимальный минимальный размер и т.п.

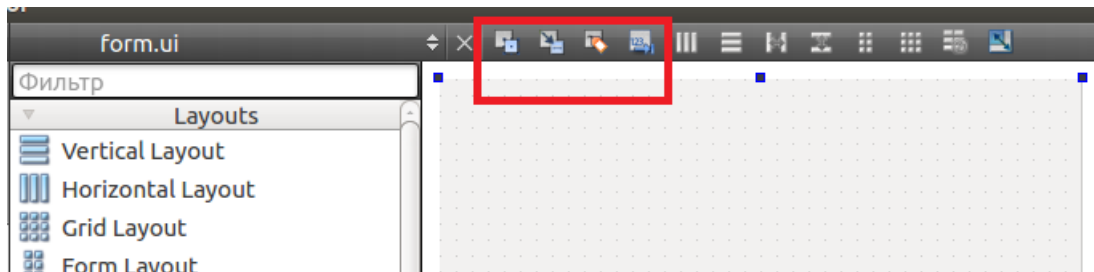
Qt Designer



(4) - Окно «Редактор сигналов и слотов» - окно редактирования соединений сигналов со слотами.



(5) – Окно «Редактор действий» – предназначено для создания, удаления и управления действиями команд создаваемой формы.



По умолчанию в Qt Designer установлен режим редактирования виджетов формы.

В Qt Designer 4 режима редактирования:

- 1) Изменение виджетов (F3);
- 2) Изменение сигналов/слотов (F4);
- 3) Изменение партнеров;
- 4) Изменение порядка обхода.

Практическая часть

Создание формы в Qt Designer

Управление АНПА 1

Курс	
+	-

Марш	Крен
+	+
-	-
0.0	0.0

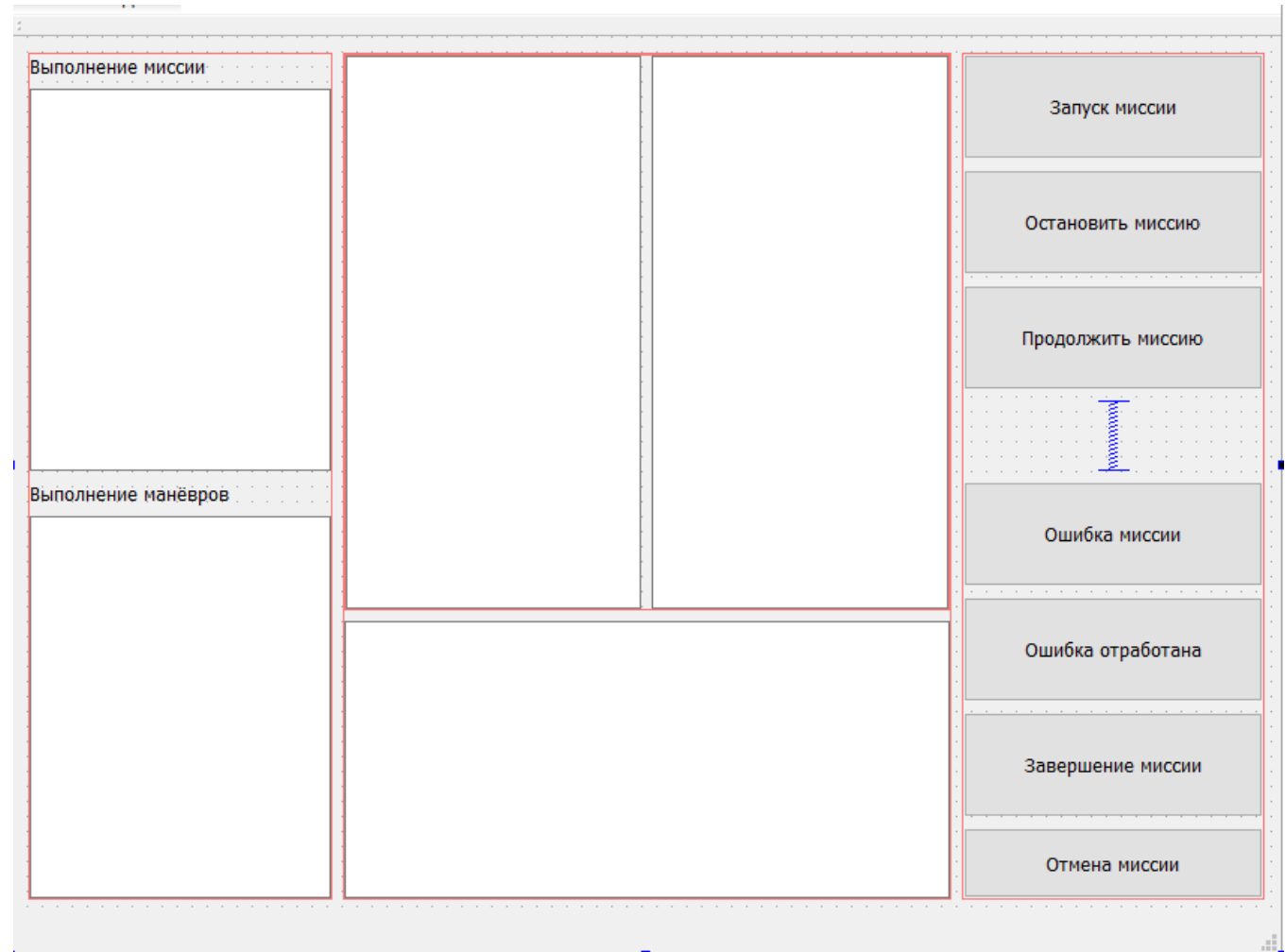
Верт. скор.	Дифферент
+	+
-	-
0.0	0.6

Объект	Класс
Form	QWidget
horizontalLayout	QHBoxLayout
btnMinus	QPushButton
btnPlus	QPushButton
label	QLabel
btnReset	QPushButton

Ui - файл

- Представляет собой XML- описание дерева объектов, размещенных на форме

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <ui version="4.0">
3   <class>MainWindow</class>
4   <widget class="QMainWindow" name="MainWindow">
5     <property name="geometry">
6       <rect>
7         <x>0</x>
8         <y>0</y>
9         <width>1039</width>
10        <height>688</height>
11      </rect>
12    </property>
13    <property name="windowTitle">
14      <string>MainWindow</string>
15    </property>
16    <widget class="QWidget" name="centralWidget">
17      <layout class="QHBoxLayout" name="horizontalLayout_2" stretch="2,4,1">
18        <item>
19          <layout class="QVBoxLayout" name="verticalLayout_2">
20            <item>
21              <widget class="QLabel" name="label">
22                <property name="text">
23                  <string>Выполнение миссии</string>
24                </property>
25              </widget>
26            </item>
27            <item>
28              <widget class="QTextBrowser" name="textBrowser">
29                <property name="sizePolicy">
30                  <sizepolicy hstyp="Expanding" vstyp="Expanding">
31                    <horstretch>0</horstretch>
32                    <verstretch>0</verstretch>
33                  </sizepolicy>
34                </property>
35                <property name="minimumSize">
36                  <size>
37                    <width>200</width>
38                    <height>0</height>
39                  </size>
```



Как использовать ui - файлы в проектах?

- Способы подключения к проекту:
 - Подключение на этапе компиляции
 - Подключение в процессе выполнения

Подключение на этапе компиляции

1. Создается форма в QtDesigner (имя_формы.ui)
 2. Запускается UIC (User Interface Compiler) и преобразует .ui – файл в класс на C++:
 1. Название файла: «ui_имя_формы.h»
 2. Класс формы создается в namespace Ui
 3. В классе есть три компоненты:
 1. Указатели на переменные виджетов
 2. Метод setupUi () – позволяет разместить форму на виджете*
 3. Метод retranslateUi() –п позволяет выполнять перевод
- *Тип виджета строго определён!

Работа с Ui-файлами

Форма.ui (XML-файл)

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>MainWindow</class>
  <widget class="QMainWindow" name="MainWindow">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>1039</width>
        <height>688</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>MainWindow</string>
    </property>
    <widget class="QWidget" name="centralWidget">
      <layout class="QHBoxLayout" name="horizontalLayout_2" stretch="2,4,1">
        <item>
          <layout class="QVBoxLayout" name="verticalLayout_2">
            <item>
              <widget class="QLabel" name="label">
                <property name="text">
                  <string>Выполнение миссии</string>
                </property>
              </widget>
            </item>
            <item>
              <widget class="QTextBrowser" name="textBrowser">
                <property name="sizePolicy">
                  <sizepolicy hsize="Expanding" vsize="Expanding">
                    <horstretch>0</horstretch>
                    <verstretch>0</verstretch>
                  </sizepolicy>
                </property>
                <property name="minimumSize">
                  <size>
                    <width>200</width>
                    <height>0</height>
                  </size>
                </property>
              </widget>
            </item>
          </layout>
        </item>
      </layout>
    </widget>
  </widget>
</ui>
```



UIC



ui_форма.h (класс на C++)

```
Namespace Ui {
  class Форма {
```

1. указатели на виджеты формы, менеджеры компоновки, группы кнопок и т.п.

2. void setupUi () – метод, который располагает форму на базовом виджете

3. void retranslateUi () – метод, который выполняет перевод параметров формы

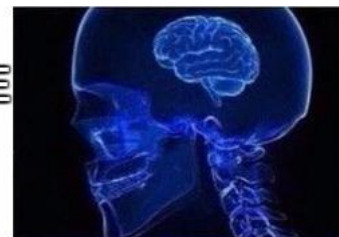
```
};
}
```

Использование формы в проекте

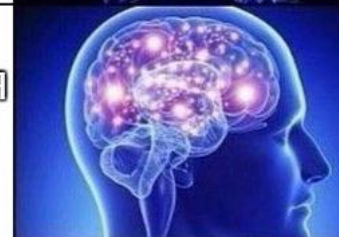
Для того, чтобы использовать файл формы в проекте существуют 4 способа:

- 1). Прямой способ (direct approach);
- 2). С использованием наследования (inheritance approach);
- 3). С использованием множественного наследования (multiple inheritance approach);
- 4). Динамическая загрузка формы;

MULTIPLE
INHERITANCE
APPROACH



ДИНАМИЧЕСКАЯ
ЗАГРУЗКА
ФОРМЫ



INHERITANCE
APPROACH



DIRECT
APPROACH



Прямой способ. Direct approach

1. В проекте создается форма;
2. Заголовочный файл формы «ui_форма.h» подключается к проекту;
3. Создается объект формы;
4. Создается объект виджета;
5. Для объекта формы вызывается `setupUi(ук-ль_на_виджет);`

Практическая часть 1. Прямой способ

main.cpp

```
1  #include <QApplication>
2
3
4  //1. Подключаем заголовочный файл формы.
5  //Этот файл автоматически создает UIC (User Interface Compiler)
6  //файл называется по следующим правилам: "ui_название_файла_формы.h"
7  #include "ui_form.h"
8
9  int main(int argc, char *argv[])
10 {
11     QApplication a(argc, argv);
12     //2. Создание виджета, на котором будем размещать форму
13     QWidget *w = new QWidget();
14     //3. Создание объекта формы
15     //При автоматической генерации UIC кода формы (т.е. ui_form.h), действует следующее правило:
16     //Для всех форм действует namespace Ui, т.е. для создания формы пишем следующее:
17     Ui::Form ui;
18     //4. Для того, чтобы разместить форму на виджете в классе формы предусмотрен следующий метод:
19     //setupUi(указатель_на_виджет)
20     ui.setupUi(w);
21     //5. чтобы отобразить виджет вызываем метод show().
22     w->show();
23     return a.exec();
24 }
```

Недостатки подхода:

- Сложно настраивать сигналы и слоты, программировать взаимодействие с интерфейсом.

Использование формы в проекте. Способ с использованием наследования.

1. Класс основного виджета создается как класс-наследник Qwidget, Qframe и т.п.
2. В классе виджета создается объект формы:
 1. Создается объект класса формы
 2. Создается указатель на объект формы
3. В конструкторе класса происходит установка формы на виджет.

Практическая часть 2. Способ с использованием наследования.

pultwidget.h

```
1  #ifndef PULTWIDGET_H
2  #define PULTWIDGET_H
3
4  #include <QWidget>
5  //1. Подключаем заголовочный файл формы.
6  //Этот файл автоматически создает UIC (User Interface Compiler)
7  //файл называется по следующим правилам: "ui_название_файла_формы.h"
8  #include "ui_form.h"
9
10 class PultWidget::public QWidget
11 {
12     ... Q_OBJECT
13
14     public:
15         ... PultWidget(QWidget *parent = 0);
16         ... ~PultWidget();
17     private:
18         ... //2. Создаем объект класса формы. Форма создается UIC по определенным правилам.
19         ... //Все формы находятся в namespace Ui
20         ... Ui::Form ui;
21 };
22
23 #endif // PULTWIDGET_H
24 ♦
```

Практическая часть 2. Способ с использованием наследования.

pultwidget.cpp

```
< > pult3/pultwidget.cpp PultWidget::~PultWidget()
1  #include "pultwidget.h"
2
3  PultWidget::PultWidget(QWidget *parent)
4  {
5      //Напомню: для того, чтобы разместить форму на виджете
6      //в классе формы предусмотрен следующий метод:
7      //setupUi(указатель_на_виджет)
8      //3. Размещаем форму на виджете,
9      //в качестве указателя на виджет используется this
10     ui.setupUi(this);
11 }
12
13
14 PultWidget::~PultWidget()
15 {
16
17 }
18
```

main.cpp

```
1  #include "pultwidget.h"
2  #include <QApplication>
3
4  int main(int argc, char *argv[])
5  {
6      QApplication a(argc, argv);
7      PultWidget w;
8      w.show();
9
10     return a.exec();
11 }
12
```

Достоинства подхода

- Возможность соединения сигналов и слотов;
- Данные формы инкапсулированы в ui-объекте;
- Можно определить множество интерфейсов в одном объекте.

Практическая часть 3

```
pult3/pultwidget.h  ui: Ui::Form *
1  #ifndef PULTWIDGET_H
2  #define PULTWIDGET_H
3
4  #include <QWidget>
5
6  //1. Не подключаем файл формы в заголовочном файле!!!
7  //2. Объявляем класс формы перед объявлением класса виджета -- PultWidget
8  namespace Ui {
9      class Form;
10 }
11
12 class PultWidget::public QWidget
13 {
14     Q_OBJECT
15
16 public:
17     PultWidget(QWidget *parent = 0);
18     ~PultWidget();
19 private:
20     //3. Создаем указатель на объект класса формы:
21     Ui::Form *ui;
22 };
23
24 #endif // PULTWIDGET_H
```

Практическая часть 3

```
1  #include "pultwidget.h"
2
3  //4. Подключаем заголовочный файл формы.
4  //Этот файл автоматически создает UIC (User Interface Compiler)
5  //файл называется по следующим правилам: "ui_название_файла_формы.h"
6  #include "ui_form.h"
7
8  PultWidget::PultWidget(QWidget *parent)
9  {
10     //5. Создаем объект формы
11     ui = new Ui::Form();
12     //6. Размещаем форму на виджете,
13     //в качестве указателя на виджет используется this
14     ui->setupUi(this);
15 }
16
17
18 PultWidget::~PultWidget()
19 {
20     //7. Добавляем удаление динамически созданного объекта ui
21     delete ui;
22 }
23
```


Достоинства подхода

- Класс UI может быть подключен перед определением класса виджета, т.е. «ui_форма.h» не подключается в .h – файле
- Т.е. форма может быть изменена, а перекомпиляция зависимых исходников не требуется:
 - Важно при наличии требования бинарной совместимости версий
 - Уменьшается время компиляции

Подход рекомендуется при разработке больших приложений и библиотек!

Способ с использованием множественного наследования

- Создается класс, который наследуется от выбранного типа виджета (QWidget, QFrame..) и от класса формы.
- Таким образом, все объекты и методы формы становятся доступны в новом классе!

Практическая часть 4. Способ с использованием множественного наследования.

По сути этот способ очень похож на предыдущий, но его достоинство в том, то мы можем напрямую обращаться ко всем виджетам формы.

Pultwidget.h

```
1  #ifndef PULTWIDGET_H
2  #define PULTWIDGET_H
3
4  #include <QWidget>
5
6  //1. Подключаем заголовочный файл формы.
7  //Этот файл автоматически создает UIC (User Interface Compiler)
8  //файл называется по следующим правилам: "ui_название_файла_формы.h"
9  #include "ui_form.h"
10
11 //2. Класс виджета "PultWidget" должен наследоваться не только от класса QWidget,
12 //но и от класса формы:
13 class PultWidget : public QWidget, public Ui::Form
14 {
15     ... Q_OBJECT
16
17 public:
18     ... PultWidget(QWidget *parent = 0);
19     ... ~PultWidget();
20 };
21
22 #endif // PULTWIDGET_H
23
```

Практическая часть 4. Способ с использованием множественного наследования.

pultwidget.cpp

```
1  #include "pultwidget.h"
2
3  PultWidget::PultWidget(QWidget *parent)
4  {
5      //3. Устанавливаем виджет на форме.
6      //Так как мы унаследовались от класса формы, то теперь можем
7      //вызвать метод setupUi() напрямую:
8      setupUi(this);
9  }
10
11
12  PultWidget::~PultWidget()
13  {
14
15  }
16
```

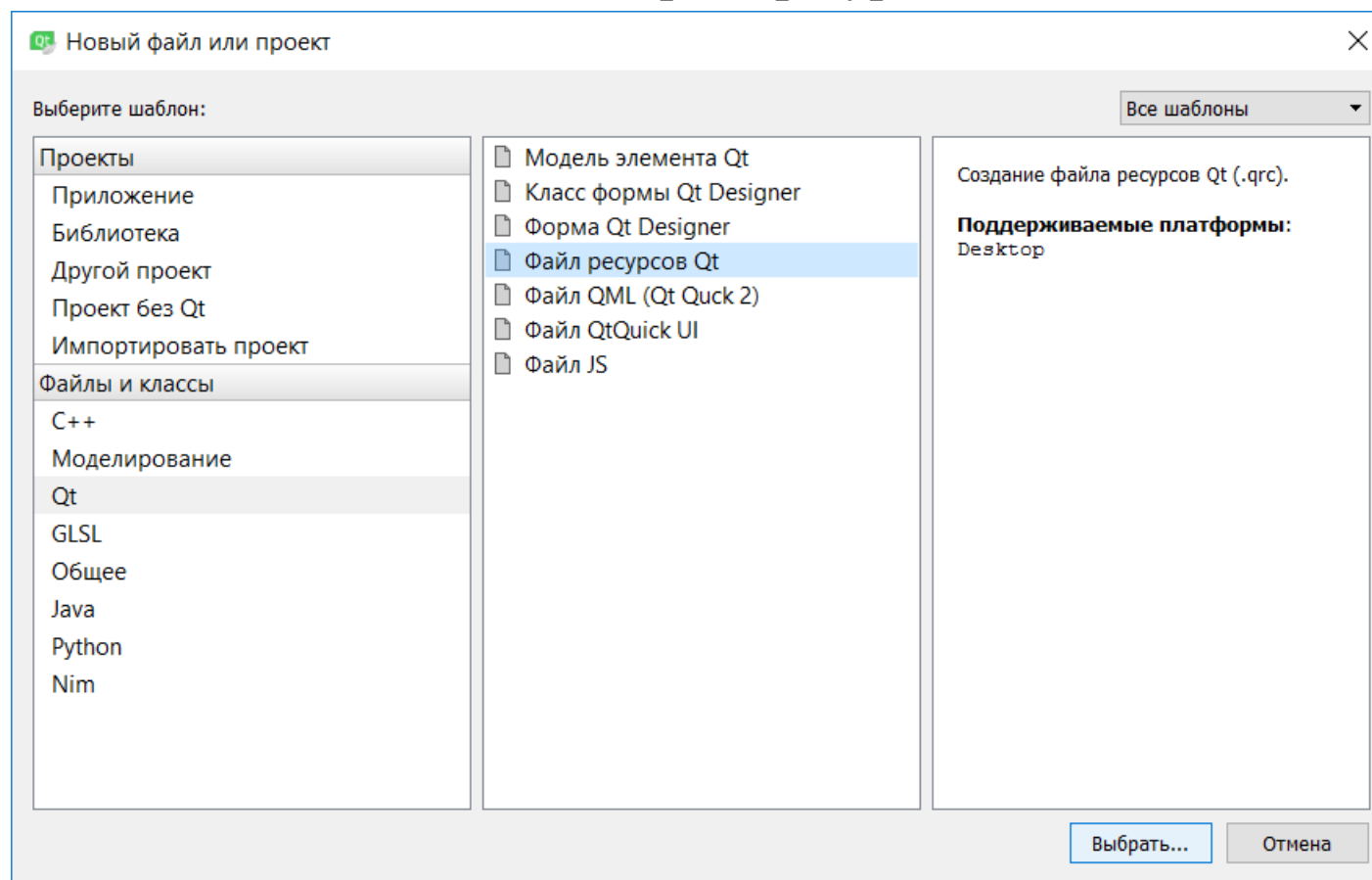
Использование формы в проекте. Динамическая загрузка формы.

Данный способ коренным образом отличается от изложенных ранее. Суть в том, что XML-репрезентация формы (ui-файл) используется в программе как есть без дополнительных преобразований. При помощи специального класса `QUiLoader`, содержащегося в модуле `QtUiTools` данные репрезентации формы могут быть загружены, и в результате их интерпретации этот класс создаст соответствующий виджет.

```
Pult.pro*
1 #-----
2 #
3 # Project created by QtCreator 2017-02-08T15:59:22
4 #
5 #-----
6 #В нашем проектном файле должен быть подключен модуль QtUiTools - это мы делаем в секции QT
7 # добавляем модуль uitools
8 QT      += core gui widgets uitools
9
10 TARGET = Pult
11 TEMPLATE = app
12
13 SOURCES += application.cpp \
14           main.cpp\
15           mainwindow.cpp
16
17 HEADERS += application.h \
18           mainwindow.h
19 #Саму форму мы располагаем в ресурсе, а ресурс подключаем к проекту в секции RESOURCES
20 #Заметьте, что секция FORMS в нашем проектном файле отсутствует, так как мы будем использовать
21 # XML-данные формы напрямую.
22 RESOURCES += \
23             resources.qrc
24
```

Использование формы в проекте. Динамическая загрузка формы.

Создадим файл ресурсов:



Создание файла ресурсов

×

Файл ресурсов Qt

➡ Размещение

Итог

Размещение

Имя:

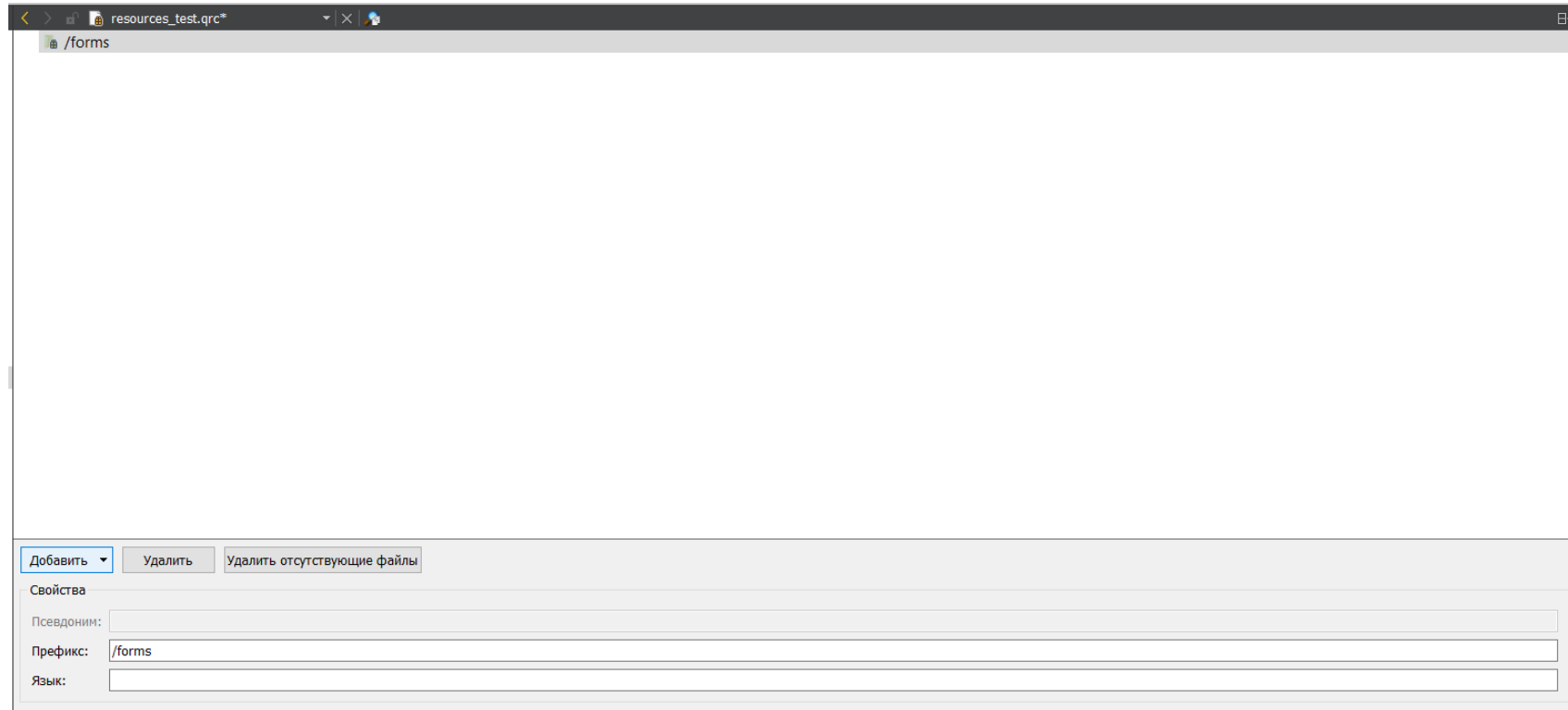
Путь:

Далее

Отмена

Использование формы в проекте. Динамическая загрузка формы.

Откроем файл ресурсов:



Использование формы в проекте. Динамическая загрузка формы.

form.h

```
1  #ifndef FORM4_H
2  #define FORM4_H
3
4  #include <QWidget>
5  #include <QtUiTools>
6
7  //Суть данного метода заключается в том, что XML-репрезентация формы (ui-файл)
8  //используется в программе как есть, без дополнительных преобразований.
9  //При помощи специального класса QUiLoader, содержащегося в модуле QtUiTools,
10 //данные репрезентации формы могут быть загружены, и в результате их интерпретации
11 //этот класс создаст соответствующий виджет.
12
13 namespace PULT {
14
15 class Form4 : public QWidget
16 {
17     Q_OBJECT
18 public:
19     explicit Form4(QWidget *parent = 0);
20     virtual ~Form4();
21
22 signals:
23
24 public slots:
25 };
26
27 } //namespace PULT
28
29 #endif // FORM4_H
30
```

form.cpp

```
1  #include "form4.h"
2  #include "QDebug"
3  namespace PULT {
4
5  Form4::Form4(QWidget *parent) : QWidget(parent) {
6
7      QUiLoader *puil = new QUiLoader(this);
8      QFile file(":/frame.ui");
9
10     QWidget *pwgtForm = puil->load(&file, this);
11
12     if (pwgtForm){
13         resize (pwgtForm->size());
14     }
15 }
16
17 Form4::~Form4(){
18 }
19
20 } //namespace PULT
21
22 }
23
```

Домашнее задание

1. Создать форму управления движением вашего ПА

На форме должны быть:

- 1). виджеты для управления движением ПА (по курсу, крену, дифференту, маршу...и тп. в зависимости от особенностей ДРК)
- 2). режимы работы (Ручной, Автоматический) (остальная функциональность зависит от вашей амбициозности)))
- 3). форма должна быть скомпонована и подгружена в проект (выбирайте любой способ из 4 описанных)

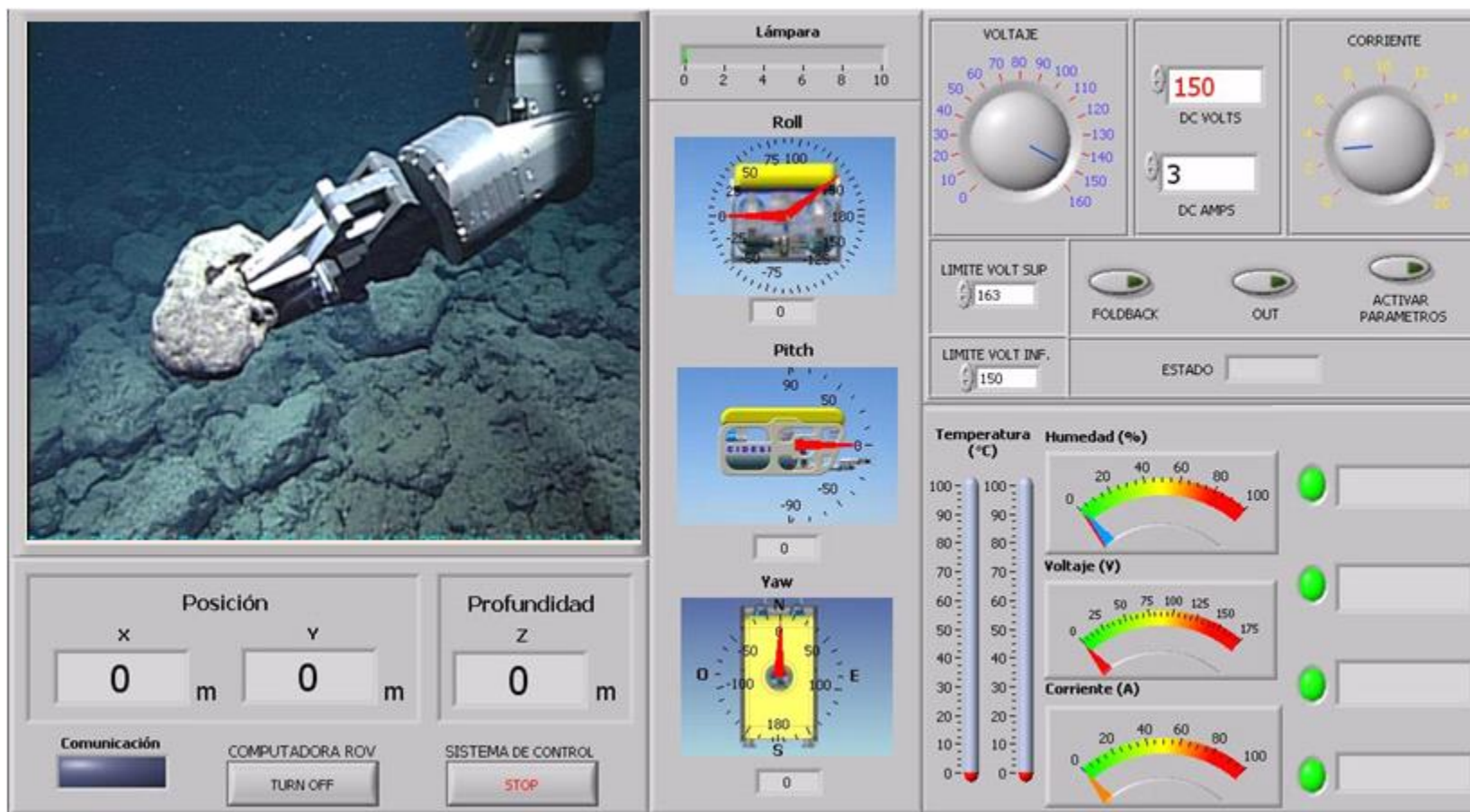
The screenshot shows a software interface titled "Frame" with standard window controls (minimize, maximize, close). The interface is divided into two main sections:

- Управление ПА (Ship Movement Controls):** This section contains six control groups arranged in a 2x3 grid:
 - Марш (Maneuver):** Includes minus and plus buttons, and a numeric input field set to 0.
 - Курс (Course):** Includes minus and plus buttons, and a numeric input field set to 0.
 - Крен (Roll):** Includes minus and plus buttons, and a numeric input field set to 0.
 - Лег (Pitch):** Includes minus and plus buttons, and a numeric input field set to 0.
 - Глубина (Depth):** Includes minus and plus buttons, and a numeric input field set to 0.
 - Дифферент (List):** Includes minus and plus buttons, and a numeric input field set to 0.
- Режимы (Modes):** This section on the right contains three buttons for different operating modes:
 - РУЧНОЙ (Manual)
 - АВТОМАТИЧЕСКИЙ (Automatic)
 - ТЕХНОЛОГИЧЕСКИЙ (Technological)

Примеры GUI для управления ПА



Примеры GUI для управления ПА



Примеры GUI для управления ПА



Примеры GUI для управления ПА

