# Компоновки виджетов (layouts)

* QLayout:
  + QBoxLayout:
    - QHBoxLayout;
    - QVBoxLayout;
  + QGridLayout;
  + QStackedLayout;

## Добавляем виджеты в компоновку

void QLayout::addWidget(QWidget \* w);

void QGridLayout::addWidget(QWidget \* widget, int fromRow, int fromColumn, int rowSpan, int columnSpan, Qt::Alignment alignment = 0)

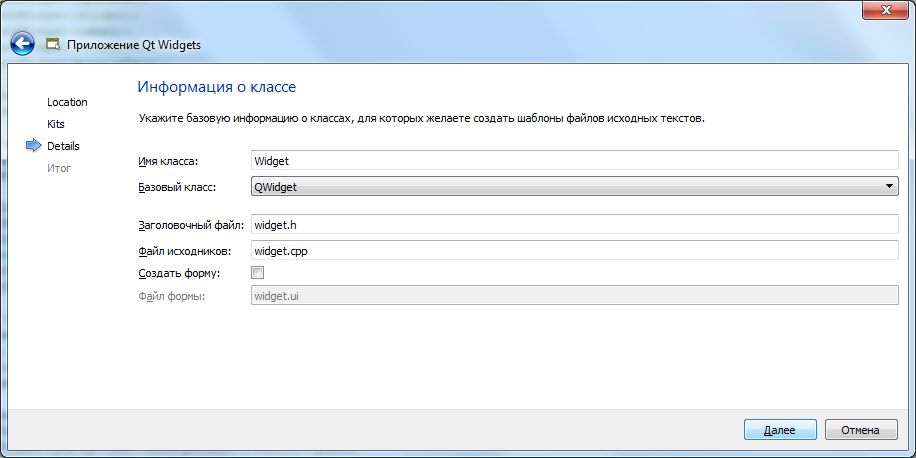
## Добавить одну компоновку в другую

void QBoxLayout::addLayout(QLayout \* layout, int stretch = 0)

void QGridLayout::addLayout(QLayout \* layout, int row, int column, int rowSpan, int columnSpan, Qt::Alignment alignment = 0)

# Создаем проект.

При создании проекта не забудьте:



## Наборный счетчик – QSpinBox

* задание диапазона – void QSpinBox::setRange ( int minimum, int maximum )
* начальное значение – void QSpinBox::setValue(int)

## **Ползунок** – QSlider

Для ползунка можно задать:

* горизонтальный/вертикальный - Qt::Horizontal или Qt::Vertical в конструкторе QSlider  
  QSlider::**QSlider**([Qt::Orientation](qthelp://org.qt-project.qtwidgets.502/qtcore/qt.html#Orientation-enum)*orientation*, [QWidget](qthelp://org.qt-project.qtwidgets.502/qtwidgets/qwidget.html) \**parent* = 0)
* диапазон значений, в которые будет пересчитываться позиция ползунка  
  void QAbstractSlider::SetRange (int min, int max)
* вид «тиков» ползунка (по умолчанию тики не рисуются)  
  QSlider:: SetTickPosition (TickPosition position)
* величину деления QSlider::SetTickInterval (int interval)

## Поле редактирования - QLineEdit

Для поля редактирования можно задать:

* множество разрешенных для ввода символов  
  QLineEdit::setValidator(const [QValidator](qthelp://org.qt-project.qtwidgets.532/qtgui/qvalidator.html) \**v*),

Перед вызовом метода требуется создать объект, производного от QValidator класса, в нашем случае – **QIntValidator** (так как задача требует ввода только целых значений). В классе QIntValidator есть удобный конструктор, который принимает допустимый диапазон вводимого значения (это означает: если при вводе очередной цифры результирующее значение выходит за заданный диапазон, эта цифра будет проигнорирована)

* QIntValidator::**QIntValidator**(int*minimum*, int*maximum*, [QObject](qthelp://org.qt-project.qtgui.502/qtcore/qobject.html) \**parent* = 0)

## Поле вывода – QLabel

Для QLabel можно задать:

* В конструкторе QLabel строчку (например «Not set»),
* Рамку (иначе рамки просто не будет)  
  QLabel::setFrameStyle(int style).
  + Для нашего примера вызовите функцию со следующими параметрами

label->setFrameStyle(QFrame::Box|QFrame::Plain);

* размер, так как «без содержимого» ширина QLabel будет нулевой  
  QLabel:: setMinimumWidth()
* выравнивать текст (по умолчанию текст «сдвинут» влево)  
  QLabel:: setAlignment(Qt::Alignment)

Для задания полей компоновки и расстояния между виджетами можно пользоваться свойствами QLayout посредством setMargin(int) и setSpacing(int)

# Сигналы и слоты

## Класс отправителя:

class Sender : public **QObject**

{

**Q\_OBJECT**

public:

void AnyFunc() { emit **changeValue(5)** }

…

**signals:**

**void changeValue(int); //только объявление!**

};

## Класс получателя:

class Reciever : public **QObject**

{

**Q\_OBJECT**

public:

…

**public slots:**

**void setValue (int);**

};

## Установка соединения

int main(int argc, char \*argv[])

{

X x; Y y;

QObject::connect(&x, SIGNAL(**changeValue** (int)), &y, SLOT(**setValue** (int)));

}

# Синхронизация элементов управления (сигналы/слоты)

|  |  |  |
| --- | --- | --- |
|  | Сигналы | Слоты |
| QSpinBox | void valueChanged(int) | void setValue(int) |
| QSlider | void QAbstractSlider::valueChanged(int) | void QAbstractSlider::setValue(int) |
| QLineEdit | void textChanged(QString) | void setText(QString) |
| QLabel |  | void setNum(int)  void setText(QString) |

LineEdit

сигнал - textChanged(QString)

Slider

слот - setValue(int)

mainWindow

void mainWindow::mySlot(QString s) {

//перевод s в целое

emit (mySignal(n));

}

mySignal(int)

### Фиксация размера виджетов в компоновке - QSizePolicy

Варианты задания размера виджетов в layout-е можно сформировать с помощью свойства   
QSizePolicy QWidget::sizePolicy. При создании виджетов работают следующие умолчания:

* Таким элементам управления, как QButton, QLineEdit, QSpinBox позволено изменять горизонтальный размер, а высота у них фиксирована
* Вертикальным слайдерам позволено изменять только высоту, горизонтальным – только ширину (это справедливо для всех полос прокрутки)
* Для остальных (большинства) виджетов - Preferred/Preferred, что означает, что оба размера виджета могут изменяться.
* …

Если умолчания не устраивают, то можно задать политику изменения размеров с помощью:  
QWidget:: setSizePolicy(QSizePolicy, QSizePolicy);