Лабораторная работа №4

Использование библиотеки элементов графического интерфейса Qt

План

- Работа с элементами графического интерфейса Qt
  - простейшее графическое приложение на Qt
  - работа с компоновщиками
  - создание приложения `ColorViewer`
  - использование `QFileDialog` — создание простейшего обозревателя текста

Необходимые знания и навыки

- Знакомство с материалом лаб. работ №№ 2, 3
- Базовое знание языка программирования C++
- Базовое знакомство с фреймворком Qt и механизмом сигналов и слотов (см. лаб. работу №3)
- Базовое знакомство с основными служебными программами Linux (`ls`, `rm`, `mkdir` и т. п.) и принципами работы систем управления пакетами

Необходимые программные и аппаратные средства

- ПК под ОС Linux (поддерживаются дистрибутивы Fedora 13, Ubuntu 10.04, openSUSE 11.3)

Работа с элементами графического интерфейса Qt

Простейшее GUI-приложение на Qt

- QApplication — движок GUI-приложения. Обязательно создаётся в единственном экземпляре.
- QWidget — базовый класс для всех элементов графического интерфейса. В конструкторе передаётся «родительский» элемент, в котором будет создан данный, либо 0, если элемент — корневое окно
- В настоящем примере будет создано пустое окно

```
#include <QApplication>
#include <QWidget>

int main (int argc,
          char **argv)
{
    QApplication app(argc,
                     argv);
    QWidget widget(0);

    widget.show();
    return app.exec();
    return 0;
}
```

Работа с компоновщиками (1/4)

- Попробуйте добавить в корневой виджет в предыдущем примере несколько элементов типов QPushButton, QLabel, QTextEdit
  - включите соответствующие заголовочные файлы, например,

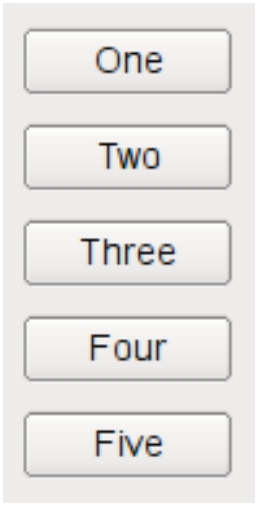
    #include <QPushButton>
  - создайте объекты, передав в конструкторе указатель на родительский widget

    QPushButton but1(&widget)
- Соберите и запустите приложение.
- Обратите внимание, что все элементы были помещены в левый верхний угол.
- Разместить элементы можно вручную, но это неудобно, поэтому для этой цели используются компоновщики

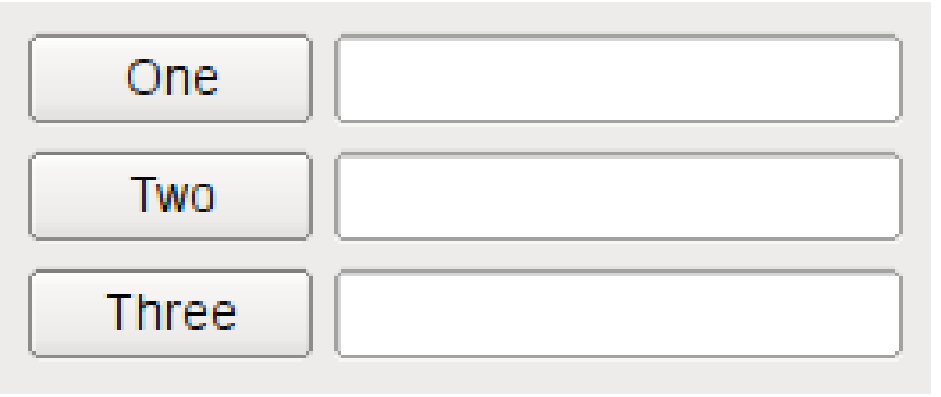
Работа с компоновщиками (2/4)

- Компоновщик (Layout manager) размещает элементы согласно определённому правилу, перекомпонует элементы после изменения размера области.
- Основные компоновщики в Qt:
  - QHBoxLayout, QVBoxLayout — размещает элементы в один ряд
  - QGridLayout — размещает элементы в ячейки таблицы
  - QFormLayout — размещает элементы сверху вниз в две колонки (используется в формах, где одна колонка — описание, а другая — поле ввода)

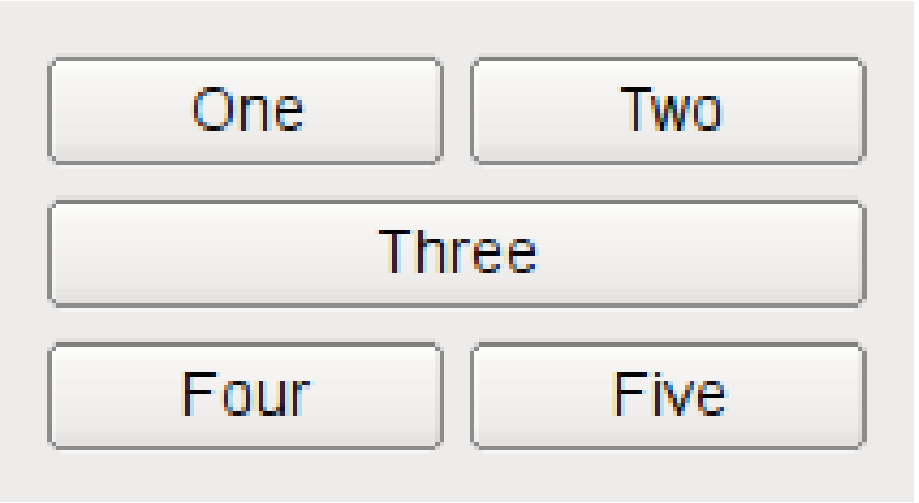
Работа с компоновщиками (3/4)



QVBoxLayout



QFormLayout



QGridLayout

SPRINT SPbSU-Intel Laboratory

Работа с компоновщиками (4/4)

- Задание: пользуясь примером в директории 03, создайте интерфейс, аналогичный представленному на картинке справа
- Используйте элементы QSpinBox, QSlider, QPlainTextEdit

Знакомство с элементами интерфейса

- Добавим функциональность созданному на предыдущем этапе приложению:
  - спин-боксы и слайдеры будут перемещаться синхронизировано в диапазоне значений от 0 до 255.
  - Цвет фона `QPlainTextEdit` будет меняться соответственно
- Выполнение:
  - Выставляем диапазон допустимых значений для `QSpinBox` и `QSlider` при помощи методов `setMinimum()` и `setMaximum()`
  - запрещаем ввод в текстовое поле: `setEnabled(false)`
  - Реализуем метод `setColor()` и слоты `setRed(int)`, `setGreen(int)`, `setBlue(int)`
  - к слотам подключаем сигналы `QSlider::sliderMoved()` и `QSpinBox::valueChanged()`
  - в реализации слотов синхронизируем значения слайдера и спин-бокса и вызываем `setColor()`

Знакомство с элементами интерфейса – stylesheets

- Для изменения цвета фона текстового поля воспользуемся таблицами стилей для описания стиля элементов.
  - таблицы стилей используют синтаксис CSS
  - будем задавать цвет в виде строки #rrggbb
  - таким образом, надо задать `QPlainTextEdit` следующий стиль:

    ```
    QPlainTextEdit { background: #rrggbb; }
    ```
  - задаём стиль при помощи метода `setStyleSheet()` (таблица стиля передаётся в виде строки)

Работа с диалогами на примере QFileDialog

- Создадим простейший обозреватель текстовых файлов
  - Создайте новый виджет и поместите на него элемент `QTextEdit`.
  - Добавьте кнопку `QPushButton` и подключите её сигнал `clicked()` к слоту `openFile()`
  - Реализуйте в слоте выбор имени файла пользователем: `QFileDialog::getOpenFileName()`
  - Откройте `QFile` в соответствии с выбранным названием
  - Прочитайте его содержимое и поместите в виде текста в элемент `QTextEdit`

Для дополнительного чтения

1) Документация по qt
   - http://doc.qt.nokia.com/4.7/index.html
   - http://doc.qt.nokia.com/4.7/widgets-and-layouts.html
   - http://doc.qt.nokia.com/4.7/layout.html

Вопросы?

sergeyle@gmail.com