

Програмування GUI

На основі мови C++ та фреймворку Qt

Лекція 4

Віджети, які реалізують введення даних



До найпростіших віджетів, що дозволяють ввести інформацію в програму, відносяться **однорядковий текстовий редактор і віджети лічильників**. Текстовий редактор дозволяє вводити будь-яку текстову інформацію, а лічильники - тільки числа.



Клас QLineEdit



Призначений для створення об'єкта однострочного текстового редактора. Можна створювати та редагувати текст, розташований в одному рядку.



Конструктори:

```
QLineEdit (QWidget *parent=0);
```



```
QLineEdit (const QString &contents, QWidget  
*parent=0),
```

де `parent` – покажчик на батьківський віджет, `contents` – задає «початковий» текст в редакторі при його створенні.



```
QLineEdit *edit=new QLineEdit ();
```

```
QLineEdit *edit1=new QLineEdit ("Hello");
```

Hello!



Методи та слоти класу QLineEdit



1. Метод `setText(const QString &str)` - програмно змінює текст в редакторі

```
edit->setText("Bye!!");
```



2. Метод `text()` – повертає текст, що знаходиться в редакторі

```
QString text=edit->text();\
```



3. Метод `setReadOnly(bool)` встановлює режим «тільки для читання».





```
edit->setReadOnly(true); // текст не можна редагувати  
edit->setReadOnly(false); // текст можна редагувати
```



Методи та слоти класу QLineEdit




 4. Метод `setEchoMode`
(`QLineEdit::EchoMode`) визначає - як
текст, введений в редактор, відображається
користувачеві

 `QLineEdit::Normal` – звичайний режим
відображення даних

 `QLineEdit::NoEcho` – текст, що вводитьься,
не відображається в редакторі

`QLineEdit::Password` – режим «пароля».

 Замість символів, що вводять відображаються
зірочки «****» або кружечки «•••» в залежності від
ОС і її налаштувань



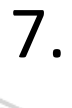
Методи та слоти класу QLineEdit



5. Метод `copy()` копіює виділений текст в буфер обміну



6. Метод `cut()` копіює виділений текст в буфер обміну і видаляє його з редактора



7. Метод `paste()` вставляє текст з буфера обміну в редактор



Сигнали класу QLineEdit



- `void returnPressed()` – висилається при натисканні на клавішу <Enter> після редагування даних в редакторі



- `void textEdited(const QString &text)` висилається при редагуванні **користувачем** тексту в редактора, `text` - посилання на рядок «нового» тексту



- `void textChanged(const QString &text)` – висилається при редагуванні **користувачем або програмній зміні** тексту в редакторі, `text` - Посилання на новий текст.



Клас QTextEdit

- Клас QTextEdit дозволяє переглядати та редагувати як простий текст, так і текст в форматі HTML. Він успадкований від класу QAbstractScrollArea, що дає можливість автоматично відображати смуги прокрутки, якщо текст не може бути повністю відображений у відведєній для нього області.
- Також можна використовувати клас QPlainTextEdit.

Методи класу QTextEdit



- `setReadOnly ()` - встановлює або знімає режим блокування зміни тексту;
- `text ()` - повертає поточний текст.



Слоти:

`setPlainText ()` - установка звичайного тексту;

`setHtml ()` - установка тексту в форматі HTML;



`selectAll ()` або `deselect ()` - виділення або зняття виділення всього тексту;

`clear ()` - очищення поля введення.



Сигнали:

`selectionChanged ()` - відправляється при змінах виділення тексту.



Віджети лічильників



В основі віджетів лічильників лежить абстрактний клас `QAbstractSpinBox`, що представляє однорядкове текстове поле для введення цифрових значень і дві кнопки зі стрілками, що дозволяють збільшувати / зменшувати число в лічильнику на задану величину.



На базі даного класу реалізовано класи:

1. `QSpinBox` – реалізує введення цілих чисел



2. `QDoubleSpinBox` – реалізує введення дійсних чисел



Class QSpinBox




 Конструктор:

```
QSpinBox (QWidget *parent=0).
```

```
QSpinBox *spin=new QSpinBox ();
```

Методи та слоти QSpinBox:

 1. Методи `void setMaximum (int max)` и `void setMinimum (int min)` Встановлюють діапазон чисел, що вводяться.

 За замовчуванням використовують діапазон [0;99].

```
spin->setMinimum(-10); //діапазон [-10;10]
```

```
spin->setMaximum(10);
```

Клас QSpinBox



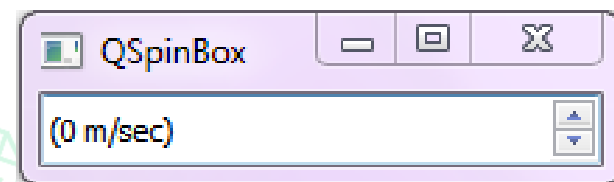
2. Методи `int maximum()` та `int minimum()` зчитують діапазон чисел, що вводяться

3. `void setSingleStep (int val)` – задає величину, на яку зміниться значення лічильника при натисканні на одну з його кнопок

```
spin->setSingleStep(5);
```


4. Методи `void setPrefix(const QString &p)` та `void setSuffix(const QString &p)` задають текст, який буде відображатися перед (prefix) або після (suffix) значення лічильника

```
spin->setPrefix("(");  
spin->setSuffix(" m/sec)");
```



Клас QSpinBox



 5. void setButtonSymbols (ButtonSymbols bs) – задає зображення на іконках лічильника

QAbstractSpinBox::UpDownArrows




 QAbstractSpinBox::PlusMinus



QAbstractSpinBox::NoButtons

 6. int value() – зчитує значення лічильника

 7. void setValue (int) – встановлює нове значення лічильника

```
spin->setValue(15);
```

```
int v=spin->value(); //v=15
```



Сигнали класу QSpinBox



Висилаються при зміні значення лічильника

1. `void valueChanged (int i)`



2. `void valueChanged (const QString
&text)`



Параметри `i` і `text` зберігають нове значення відповідно в числовий або текстовій формі.



Клас QDoubleSpinBox



Аналогічний класу QSpinBox, тобто має такі ж методи, слоти і сигнали, тільки числові параметри в них мають тип `double`, а не `int`. За замовчуванням діапазон чисел, що вводяться `[0.00; 99.00]`.



Клас QValidator



Призначений для перевірки текстової інформації, що зберігається в поле текстового редактора або в об'єкті класу `string`, заданим програмістом вимогам.

Наприклад, чи є текстова інформація числом з наперед заданого діапазону.

Об'єкти класу `QValidator` і його спадкоємців можна використовувати для контролю введення тексту в однорядковий текстовий редактор `QLineEdit`.

Конструктор:

`QValidator (QObject *parent)`, де `parent` – покажчик на батьківський об'єкт.

Клас QValidator



Як батьківський об'єкт можна використовувати головне вікно програми, текстовий редактор, з яким буде пов'язаний об'єкт QValidator тощо. Для практичного використання даного класу необхідно створити свій клас валідатора, успадкувавши клас QValidator і перевизначити його віртуальний метод:

```
QValidator::State validate(QString  
&input, int &pos) const.
```

Параметр `input` вказує на перевіряється текст, `pos` дозволяє задати першу позицію курсора в текстовому редакторі, наприклад, для вказівки позиції неприпустимого символу.

Клас QValidator



Приклад.

Розробимо валідатор, який дозволяє вводити в редактор не більше 5 символів.



```
class MyValidator: public QValidator
{public:
    MyValidator (QObject *parent=0):QValidator
    (parent) {}
    State validate (QString &str, int &pos) const
    {pos=0;
    if (str.length()>5) return Invalid;
    return Acceptable;
    }
};
```




Клас QValidator



 Значення, що повертається, методу `validate`:

`QValidator::Invalid` – некоректне введення

 `QValidator::Intermediate` – частково некоректне введення

`QValidator::Acceptable` – коректне введення

 Для зв'язку валідатора і однострочного текстового редактора використовується методу класу `QLineEdit`:

 `void setValidator (const QValidator *v),`

параметр `v` - покажчик на об'єкт валідатора, що підключається до текстового редактора.



Клас QValidator



 Приклад.

```
QLineEdit *edit=new QLineEdit ("",this);  
MyValidator *validator =new MyValidator  
    (edit);  
edit->setValidator(validator);
```

Тепер валідатор буде перевіряти довжину введеного тексту в символах, і, якщо користувач захоче ввести більш 5 символів, валідатор це дію заблокує.



Класи QIntValidator, QDoubleValidator



В Qt є «готові» класи, створені на базі класу QValidator, які дозволяють виконати перевірку на коректність введення цілих і дійсних чисел. Клас QIntValidator призначений для перевірки введення цілих чисел.

Конструктор:

```
QIntValidator (int bottom, int top,  
QObject *parent),
```

де bottom та top задають допустимий діапазон чисел, що вводяться.

Також задати діапазон можна слотами:

```
void setBottom (int) та void setTop (int) -  
встановлюють відповідно нижню і верхню межу
```

Клас QIntValidator



Приклад.

```
QString str;  
int pos = 0;  
QIntValidator v(100, 900, this);  
QValidator::State state;  
str = "1"; state=v.validate(str, pos); // state= Intermediate  
str = "012"; state= v.validate(str, pos); // state= Intermediate  
str = "123"; state=v.validate(str, pos); // state= Acceptable  
str = "678"; state=v.validate(str, pos); // state= Acceptable  
str = "999"; state=v.validate(str, pos); // state= Intermediate  
str = "1234"; state=v.validate(str, pos); // state= Invalid  
str = "-123"; state=v.validate(str, pos); // state= Invalid  
str = "abc"; state=v.validate(str, pos); // state= Invalid  
str = "12cm"; state=v.validate(str, pos); // state= Invalid
```



Клас QDoubleValidator



Клас QDoubleValidator аналогічний класу QIntValidator, тільки призначений для перевірки дійсних чисел.

