

Практическая работа №4 – Строки в Qt

Цель работы

Знакомство с функционалом строк в Qt.

Теоретическое введение

Для работы со строками в Qt реализован класс QString, объекты которого позволяют хранить строки в формате Unicode. Практически вся функциональность в Qt, связанная со строками использует QString.

Класс QString по своей сути, аналогичен «вектору», т.е. QString – это контейнер, хранящий элементы символьного типа QChar. Таким образом, QString предоставляет большое количество методов и операторов для работы с символами, строками и подстроками.

Строки можно сравнивать друг с другом при помощи операторов сравнения ==, !=, <, >, <= и >=, например:

```
QString str = "Qt";  
bool b1 = (str == "Qt"); // b1 = true  
bool b2 = (str == "QT"); // b2 = false
```

Как видно, результат сравнения зависит от регистра символов.

При помощи метода isEmpty() можно узнать, не пуста ли строка:

```
QString str;  
bool b1 = str.isEmpty(); // true
```

Того же результата можно добиться, проверив длину строки методом length():

```
QString str;  
bool b1 = (str.length() == 0); // true
```

В классе QString имеются различия между пустыми и нулевыми строками, таким образом, строка, созданная при помощи конструктора по умолчанию, является нулевой строкой. Например:

```
QString str1 = "";
QString str2;
str1.isNull(); // false
str2.isNull(); // true
```

Объединение можно произвести разными способами, например, при помощи операторов += и + или вызовом метода append(). Например:

```
QString str1 = "Библиотека";
QString str2 = "Qt";
QString str3 = str1 + str2; // str3 = "Библиотека Qt"
str1.append(str2); //str1 = "Библиотека Qt"
```

Для замены определенной части строки можно использовать метод replace(). Например:

```
QString str = "Программирование";
str.replace("ирование", "а"); // str = "Программа"
```

Для преобразования строки в верхний или нижний регистр используются методы toLower() или toUpper(). Например:

```
QString str1 = "ПрОгРаМмА";
QString str2 = str1.toLower(); // str2 = "программа"
QString str3 = str1.toUpper(); // str3 = "ПРОГРАММА"
```

Строка может быть разбита на массив строк при помощи метода split(). Результатом будет объект QStringList, являющийся контейнером, хранящим объекты QString.

Таким образом можно разделить строку, содержащую предложение на отдельные слова. В качестве аргумента указывается символ, который является разделителем, в данном случае это пробел.

```
QString str = "Библиотека Qt";
QStringList list = str.split(" "); // "Библиотека", "Qt"
```

Рассмотрим также другой случай:

```
QString str = "Библиотека Qt ";
QStringList list = str.split(" "); // "Библиотека", "Qt", ""
```

В данном случае после "Библиотека Qt" был добавлен пробел, так что получилось выражение "Библиотека Qt ", поэтому после выполнения метода `split()`, было получено 3 элемента, последний из которых пустая строка.

Чтобы избежать этого, необходимо указать флаг, исключающий пустые строки:

```
QString str = "Библиотека Qt ";
QStringList list =
str.split(" ", Qt::SkipEmptyParts); // "Библиотека", "Qt"
```

Таким образом были получены все слова, без пустот.

Операция объединения списка строк в одну строку производится при помощи метода `join()`. Аргументом указывается символ или строка, который будет вставлен между каждым элементом, при создании единой строки. Например, в данном случае, между словами будет добавлен пробел:

```
QStringList list;
list.append("Библиотека");
list.append("Qt");
str = list.join(" "); // "Библиотека Qt"
```

В строке можно выполнить проверку, что она начинается или заканчивается на определённый символ или подстроку, для этого используются методы `startsWith()` и `endsWith()`:

```
QString str = "Библиотека Qt";
bool b1 = str.startsWith("Библ"); // true
bool b2 = str.endsWith("Qt"); // true
```

Если требуется сформировать строку из нескольких значений, можно использовать аргументы. Для этого используется метод `arg()` с указанием номера аргумента от 1 до 99, а также в самой строке необходимо указать место, куда будет вставлено значение, с помощью символа `%i`, где `i` – номер аргумента.

```
QString str = "Дата: %1.%2.%3. %1 %4 шёл дождь.";
str = str.arg(17).arg("09").arg(2024).arg("сентября");
// str = "Дата: 17.09.2024. 17 сентября шёл дождь."
```

Обратите внимание, что при вызове метода `arg()` значения будут форматироваться в том порядке, в котором они были вызваны, т.е., если в строке указаны аргументы `%1 %2 %3 %1`, то при первом вызове метода `arg()` будут заменены все значения `%1`, при втором – все `%2`, а при третьем – все `%3`.

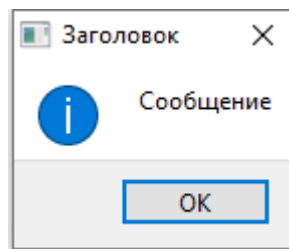
Применять форматирование строк можно в различных случаях, например, для вывода строки сообщения пользователю, можно воспользоваться классом `QMessageBox`.

Подключить его можно директивой `#include <QMessageBox>`

А для вывода сообщений доступны несколько вариантов:

Информативное сообщение:

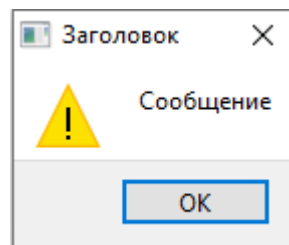
```
QMessageBox::information(this, "Заголовок", "Сообщение");
```



Информационное сообщение

Сообщение с предупреждением:

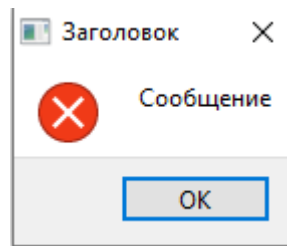
```
QMessageBox::warning(this, "Заголовок", "Сообщение");
```



Сообщение с предупреждением

Сообщение о критической ошибке:

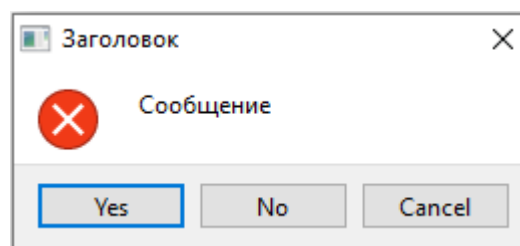
```
QMessageBox::critical(this, "Заголовок", "Сообщение");
```



Сообщение об ошибке

Также в диалоговое окно можно добавить кнопки и проверить, какая из них была нажата пользователем:

```
QMessageBox::StandardButton chose =  
QMessageBox::critical(this, "Заголовок", "Сообщение",  
QMessageBox::Yes|QMessageBox::No|QMessageBox::Cancel);  
if(chose == QMessageBox::Yes)  
{  
    // выбрана кнопка "Да"  
}
```



Сообщение с кнопками действий.

Таким образом можно выводить информацию для пользователя с помощью диалоговых окон.

Задание

Создать программу с формой регистрации, в которой должны присутствовать следующие параметры:

1. Имя пользователя, не более 15 символов латиницы, допускается использование цифр.
2. Строка для ввода фамилии, имени и отчества в одну строку, разделённую пробелами. Проверить, что ФИО начинается с заглавной буквы, содержит только буквы. Длина слова (фамилии, имени и отчества) не должна превышать 15 символов.

3. Переключатель выбора пола человека.
4. Строка паспорта в формате «серия <пробел> номер», например, 4211 324521.
5. Строка даты рождения в виде строки в формате «день.месяц.год», например, 01.08.2005 – наличие 0 в числах до 10 обязательно.
6. Строка номера телефона в формате +7-XXX-XXX-XX-XX.
7. Строка e-mail, в правильном формате, длина e-mail не более 20 символов.

Для каждого поля выполнить проверку корректности данных, используя методы работы со строками, описанные в теоретическом материале данной работы.

В случае корректности заполнения всех полей вывести пользователю информационное сообщение следующего формата:

«Вы успешно зарегистрировали аккаунт <имя пользователя>. Ваше имя: <имя>, ваша фамилия: <фамилия>, ваше отчество: <отчество>. Ваш пол: <пол текстом>. Серия Вашего паспорта: <серия паспорта>, номер: <номер паспорта>. Вы родились <дата, месяц текстом, год>. Ваш номер телефона: <номер телефона начиная с 8, без пробелов>. Ваш e-mail: <e-mail>. Спасибо за регистрацию!».

Вместо <...> необходимо вставить корректные данные с формы.

В случае, если введенные данные некорректны – отобразить сообщение об ошибке, с указанием проблемы.

Пример формы:

MainWindow

Форма регистрации

Имя пользователя:

ФИО:

Пол: ☒ Мужской ☐ Женский

Паспорт:

Дата рождения:


Телефон:

E-mail:

Пример формы

Пример сообщения:


Регистрация

 Вы успешно зарегистрировали аккаунт ivan24. Ваше имя: Иван, ваша фамилия: Иванов, ваше отчество: Иванович. Ваш пол: Мужской. Серия Вашего паспорта: 4211, номер: 732418. Вы родились 04 июля 1992 года. Ваш номер телефона: 89997513216. Ваш e-mail: ivan24@mail.ru. Спасибо за регистрацию!

Пример сообщения при корректных данных

Пример ошибки:

Ошибка

 Вы не указали дату рождения!

Пример сообщения об ошибке

Контрольные вопросы для защиты работы

1. Реализация дополнительного поля в форме.