Практическая работа №5 – Регулярные выражения

Цель работы

Знакомство с регулярными выражениями в Qt.

Теоретическое введение

Регулярное выражение – это последовательность символов, представляющая из себя шаблон выражения (текста), позволяющий находить подстроки в тексте. Обычно используется для поиска и замены подстроки в тексте.

Для использования регулярного выражения необходимо задать паттерн (шаблон), по которому будет производиться поиск. Шаблоны регулярных выражений представлены в таблице:

Символ	Описание	Пример
.	Любой символ	a.b
$	Должен быть конец строки	Abc$
[ ]	Любой символ из заданного набора	[abc]
-	Определяет диапазон символов в группе [ ]	[0-9A-Za-z]
^	В начале набора символов означает любой символ, не вошедший в набор	[^def]
*	Символ должен встретиться в строке ни разу или несколько раз	A*b
+	Символ должен встретиться в строке минимум 1 раз	A+b
?	Символ должен встретиться в строке 1 раз или не встретиться вообще	A?b
{n}	Символ должен встретиться в строке указанное число раз	A{3}b

{n,}	Допускается минимум n совпадений	a{3,}b
{,n}	Допускается до n совпадений	a{,3}b
{n,m}	Допускается от n до m совпадений	a{2,3}b
|	Ищет один из двух символов	ac|bc
\b	В этом месте присутствует граница слова	a\b
\B	Границы слова нет в этом месте	a\Bd
( )	Ищет и сохраняет в памяти группу найденных символов	(ab|ac)ad
\d	Любое число	
\D	Все, кроме числа	
\s	Любой тип пробелов	
\S	Все, кроме пробелов	
\w	Любая буква, цифра или знак подчеркивания	
\W	Все, кроме букв	
\A	Начало строки	
\b	Целое слово	
\B	Не слово	
\Z	Конец строки (совпадает с символом конца строки или перед символом перевода каретки)	
\z	Конец строки (совпадает только с концом строки)	

Для того чтобы найти конкретные символы, нужно поместить их в квадратные скобки, например, `[ab]` будет искать совпадение с `a` или `b`. Чтобы не писать все символы алфавита подряд, можно укаызать диапазон, например `[A-Z]` совпадает с любой буквой в верхнем регистре, `[a-z]` — с любой буквой в нижнем регистре, а `[0-9]` — с любой цифрой. Можно совмещать такие записи, например, `[a-z7]` будет совпадать с любой буквой в нижнем регистре и с цифрой `7`.

Также можно исключать символы, поставив перед ними знак ^. Например, [^0-9] будет соответствовать всем символам, кроме цифр.

В фигурных скобках указываются величины, называемые пределами. Они позволяют точно задать количество раз, которое символ должен повторяться в тексте. Например, a{4,5} будет совпадать с текстом, если буква a встретится в нем от 4 до 5 раз подряд. Например, в следующем отрывке задано регулярное выражение для ip-адреса, с помощью которого можно проверить строку на содержание в ней ip-адреса:

```
QRegularExpression reg("[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}");
QString str = "Эта строка содержит ip-адрес 123.222.63.1";
Bool b1 = str.contains(reg);    //true
```

Стоит отметить, что для указания символа точки в регулярном выражении перед ним стоит обратная косая черта (\), а в соответствии с правилами языка программирования C++ для ее задания в строке она должна удваиваться, тем самым «экранируя» спецсимвол. Если бы косой черты не было, то точка имела бы в соответствии с таблицей значение «любой символ», и регулярное выражение распознавало бы, например, строку “1z2y3x4”, как ip-адрес, что, разумеется, не правильно.

Шаблоны можно комбинировать при помощи символа |, задавая ветвления в регулярном выражении – также, как при использовании операторов if/else в коде. Регулярное выражение с двумя ветвями совпадает с подстрокой, если совпадает одна из ветвей. Например:

```
QRegularExpression rxp("(.com|.ru)");
bool b1 = rxp.match("www.bhv.ru").hasMatch();   // true
bool b2 = rxp.match("www.bhv.de").hasMatch();   // false
```

Указанные в таблице символы с обратной косой чертой (обратным слешем) позволяют значительно упростить регулярные выражения. Например, регулярное выражение [a-zA-Z0-9_] идентично выражению \w.

В Qt для работы с регулярными выражениями используется класс QRegularExpression.

Для выполнения простого сопоставления, можно вызвать метод match():

```
// сопоставление двух цифр, пробела и слово
QRegularExpression re("\\d\\d \\w+");
QRegularExpressionMatch match = re.match("abc123 def");
bool hasMatch = match.hasMatch(); // true
```

Если сопоставление успешно, с помощью метода `.captured(0)` можно извлечь выделенную подстроку:

```
QRegularExpression re("\\d\\d \\w+");
QRegularExpressionMatch match = re.match("abc123 def");
if (match.hasMatch()) {
    QString matched = match.captured(0); // matched == "23 def"
    // ...
}
```

В данном случае аргумент 0 означает, что мы хотим получить подстроку, соответствующую всему шаблону.

Также возможно извлечение конкретных подстрок, начиная с 1, если в шаблоне используются группы:

```
QRegularExpression re("^(\\d\\d)/(\\d\\d)/(\\d\\d\\d\\d)$");
QRegularExpressionMatch match = re.match("08/12/1985");
if (match.hasMatch()) {
    QString day = match.captured(1); // day == "08"
    QString month = match.captured(2); // month == "12"
    QString year = match.captured(3); // year == "1985"
    // ...
}
```

Обратите внимание, что группы нумеруются начиная с 1.

Также в Qt имеется перегрузка этой функции, требующая QString как параметр для извлечения именованных подстрок:

```
QRegularExpression
re("^(?<date>\\d\\d)/(?<month>\\d\\d)/(?<year>\\d\\d\\d\\d)$");
QRegularExpressionMatch match = re.match("08/12/1985");
if (match.hasMatch()) {
    QString date = match.captured("date"); // date == "08"
    QString month = match.captured("month"); // month == "12"
    QString year = match.captured("year"); // year == 1985
}
```

Таким образом возможно использовать строковые обозначения при извлечении подстрок.

Задание:

Доработать программу из предыдущей практической работы, заменив проверку вводимых данных с помощью регулярных выражений:

1. Имя пользователя, не более 15 символов латиницы, допускается использование цифр.
2. Строка для ввода фамилии, имени и отчества в одну строку, разделённую пробелами. ФИО должны начинается с заглавной буквы, содержит только буквы. Длина слова (фамилии, имени и отчества) не должна превышать 15 символов.
3. Строка паспорта в формате «серия <пробел> номер», например, 4211 324521.
4. Строка даты рождения в виде строки в формате «день.месяц.год», например, 01.08.2005 – наличие 0 в числах до 10 обязательно.
5. Строка номера телефона в формате +7-XXX-XXX-XX-XX.
6. Строка e-mail, в правильном формате, длина e-mail не более 20 символов.

Контрольные вопросы для защиты работы

1. Реализация дополнительного поля в форме.