

2005 字符串类应用与常用基本数据类型--零声教育 vico 老师

一、Qt 字符串类应用

1、操作字符串

1.1 QString 提供一个二元的“+”操作符，主要用于组合两个字符串。QString str1="Hello world"传递给 QString 一个 const char* 类型的 ASCII 字符串“Hello world”，它被解释为一个典型的以“\0”结尾的 C 类型字符串。

```
1  #include <QCoreApplication> // Qt提供一个事件循环
2
3  #include <QDebug> // 输出流
4
5  int main(int argc, char *argv[])
6  {
7      QCoreApplication a(argc, argv);
8
9      // 1:QString 提供二元"+"操作符应用，功能一样"+="
10     QString str1="Ling";
11     str1=str1+"Sheng EDU!";
12     qDebug()<<str1; // 打印信息
13     qDebug()<<qPrintable(str1); // 去掉双引号
14
15     QString str2="12345";
16     str2+="ABCDE";
17     qDebug()<<qPrintable(str2); // 去掉双引号
18
19     return a.exec();
20 }
21
```

1.2 QString::append() 函数具备与 “+=” 操作符同样的功能，直接在一个字符串末尾添加另一个字符串。

```
19 // 2:QString::append()函数
20 QString str1="Good";
21 QString str2="bye";
22 str1.append(str2); // str1="Good bye"
23 qDebug()<<qPrintable(str1);
24 str1.append(" Hello world!");
25 qDebug()<<qPrintable(str1); // str1="Goodbye Hello world!"
26
```

1.3 组合字符串：QString::sprintf()。其实它跟 C++ 库当中 sprintf() 函数一样。

```
27 // 3:QString::sprintf()函数
28 QString strtemp;
29 strtemp.sprintf("%s","Hello ");
30 qDebug()<<qPrintable(strtemp);
31 strtemp.sprintf("%s","Hello world.");
32 qDebug()<<qPrintable(strtemp);
33
34 strtemp.sprintf("%s %s","Welcome","to you.");
35 qDebug()<<qPrintable(strtemp);
36
```

1.4 字符串组合方式 QString::arg() 函数，该函数的重载可以处理多种数据类型。因为它类型安全，同时支持 Unicode，可以改变 %n 参数顺序。

```
37 // 4:QString::arg()函数
38 QString strTemp;
39 strTemp=QString("%1 was born in %2.").arg("Sunny").arg(2000);
40 qDebug()<<strTemp;
41
```

1.5 各位朋友们 QString 还有其它组合字符串方法：insert()/prepend()/replace() 等等，大家可以自己测试。

2、查询字符串

2.1 函数 `QString::startsWith()` 判断一个字符串是否以某个字符串开头。`Qt::CaseInsensitive` 代表大小写不敏感；`Qt::CaseSensitive` 表示大小写敏感。对应关系函数：`QString::endsWith()`。

```
42 // 4:QString::startsWith()函数
43 QString strTemp="How are you";
44 qDebug()<<strTemp.startsWith("How",Qt::CaseSensitive); // true
45 qDebug()<<strTemp.startsWith("are",Qt::CaseSensitive); // false
46
```

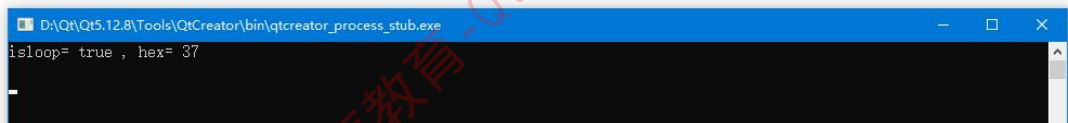
2.2 函数 `QString::contains()` 判断一个指定的字符串是否出现过。

```
47 // 5:QString::contains()函数
48 QString strTemp="How are you";
49 qDebug()<<strTemp.contains("How",Qt::CaseSensitive); // true
50
51
```

2.3 `QString::toInt()` 函数将字符串转换为整型数值。


`toDouble()/toFloat()/toLong()` 等等。

```
int hex=str.toInt(&isloop,16); // isloop=true hex=37
qDebug()<<"isloop="<<isloop<<","<<"hex="<<hex<<endl;
```



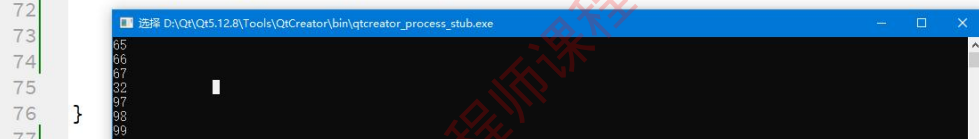
2.4 `QString::compare()` 函数对两个字符串进行比较。

```
61 // 7:QString::compare()函数
62 int a1 = QString::compare("abcd","ABCD",Qt::CaseInsensitive); // 大小写不敏感
63 int b1 = QString::compare("about","Cat",Qt::CaseSensitive);
64 int c1 = QString::compare("abcd","Cat",Qt::CaseInsensitive);
65 cout<<"a1="<<a1<<","<<"b1="<<b1<<","<<"c1="<<c1<<endl;
```



2.5 Qt 将 `QString` 转换成 ASCII 码

```
67 // 8:Qt将QString转换成ASCII码
68 QString str="ABC abc";
69 QByteArray bytes=str.toUtf8();
70 for(int i=0;i<str.size();i++)
71     qDebug()<<int(bytes.at(i));
72
73
74
75
76
77 }
```

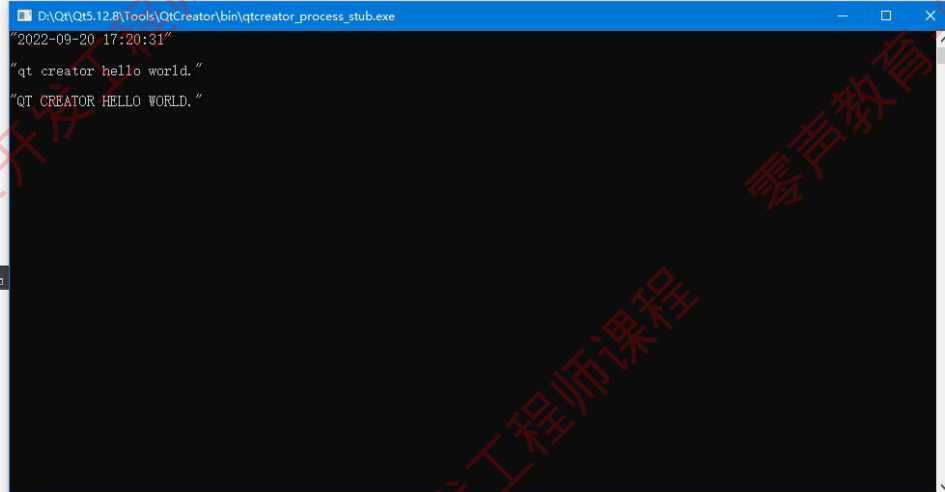


二、Qt 常见基本数据类型（注意：定义在#include <QtGlobal>）

类型名称	注释	备注
qint8	signed char	有符号 8 位数据
qint16	signed short	16 位数据类型
qint32	signed int	32 位有符号数据类型
qint64	long long int 或(__int64)	64 位有符号数据类型，Windows 中定义为 __int64
qintptr	qint32 或 qint64	指针类型 根据系统类型不同而不同，32 位系统为 qint32、64 位系统为 qint64
qlonglong	long long int 或(__int64)	Windows 中定义为 __int64
qptrdiff	qint32 或 qint64	根据系统类型不同而不同，32 位系统为 qint32、64 位系统为 qint64
qreal	double 或 float	除非配置了-qreal float 选项，否则默认为 double
quint8	unsigned char	无符号 8 位数据类型
quint16	unsigned short	无符号 16 位数据类型
quint32	unsigned int	无符号 32 位数据类型
quint64	unsigned long long int 或 (unsigned __int64)	无符号 64 比特数据类型，Windows 中定义为 unsigned __int64
quintptr	quint32 或 quint64	根据系统类型不同而不同，32 位系统为 quint32、64 位系统为 quint64
qulonglong	unsigned long long int 或 (unsigned __int64)	Windows 中定义为 __int64
uchar	unsigned char	无符号字符类型
uint	unsigned int	无符号整型
ulong	unsigned long	无符号长整型
ushort	unsigned short	无符号短整型

大家可以提前研究（后继会讲）：QByteArray、QVector、QVariant 等等。

```
75
76 // 8:QDateTime QByteArray
77 QDateTime dt;
78 QString strDT=dt.currentDateTime().toString("yyyy-MM-dd hh:mm:ss");
79 qDebug()<<strDT<<endl;
80
81 QByteArray a1("Qt Creator Hello World.");
82 QByteArray b1=a1.toLower(); // 将字符串大写字母转小写，小写不变
83 qDebug()<<b1<<endl;
84 QByteArray c1=a1.toUpper();
85 qDebug()<<c1<<endl;
86
87 return a.exec();
88 }
```



The screenshot shows a console window titled "D:\Qt\Qt5.12.8\Tools\QtCreator\bin\qtcreator_process_stub.exe". The output of the program is as follows:

```
"2022-09-20 17:20:31"
"qt creator hello world."
"QT CREATOR HELLO WORLD."
```