第2章-调试输出

author：岳石磊 copyright：科林明伦 内部资料禁止外泄

1. qDebug

需要 #include<qDebug>
在程序的输出窗口上输出信息：
兼容了 C 的 printf 和 C++ cout 输出。
用法一：void debug(const char *msg, ...)const 像 printf一样输出。
用法二：QDebug debug() const; 像 cout 一样输出，需要#include。

2. QString

1. 初始化

可以用 字符、常量字符串、字符指针、字符数组 等类型进行初始化QString。
QString 的构造并不支持 std::string 类型初始化，需要将 std::string 转化为 const char*，适用于初始化和赋值。
QString 的 operator= 也并不支持std::string 类型，但可以用其提供的静态函数
QString::fromStdString(const std::string &str)
注意此函数并不直接作用于QString本身，而是通过返回值返回。

2. 拼接

它提供了 二元操作符 +、+= 可进行拼接。
QString 提供了一个类似于+= 功能的 append()函数，它是直接作用于QString本身的，并返回**拼接后**的字符串（QString&），以便继续操作。
prepend 在字符串前面拼接，用法同append()。

3. sprintf 格式化

QString 还提供了类似于 printf 格式化一样，整合字符串的函数，sprintf。
sprintf 它是直接作用于调用的str对象的，相当于对str重新赋值（是 = 而不是+=），返回值为格式化后的完整字符串。

4. arg 灵活的格式化

我们还可以使用 arg 函数来灵活的构建字符串，它与sprintf 不同，不是直接作用于原来QString的，所以我们需要接一下返回值。
arg的功能是**按小到大顺序依次替换**字符串中的 %n，n 范围1~99，如果n 大于99，取%后两位进行替

换，其余保留。arg 支持 int ,double ,char,long,QString...等多种变量类型。平时在使用的过程中%n n 最好从1按照顺序开始。
注意：如果替换的字符串中包含%n的情况，后续的替换顺序、位置有可能会受到影响。

```
    QString str("%1 %2");
    str.arg("%1").arg("str"); //"str %2"
```

下面也是一种常见的出错情况，输出结果并不是"113322"

```
    QString str("%1%3%2");
    QString strr = str
            .arg("11")    //"11%3%2"
            .arg("22")    //"11%322"
            .arg("33");   //"11332"
```

5. QString与数字转换

还可以将数字类型的转换为字符串 number
number()： 重载了多个函数，支持 int、long、double 等转换为字符串。并不直接作用于本身，通过返回值返回。
setNum()： 直接作用于本身，当然也可以通过返回值返回。
将字符串转换为数字类型使用 toInt、toDouble、toLong 等。

```
int   //返回值 为转换成功后的int 类型
QString::toInt(
bool *ok = Q_NULLPTR,  //转换成功失败，如果发生错误 ok 为false,否则为true
int base = 10          //转换基数，标识了转换前的字符串是多少进制数
) const
```

待转换的数字并不一定只在 0~9 之间，对于 QString("10a").toInt 转换的成功失败，取决于转换基数。

6. 定位

contains() 函数用于判断是否包含一个字符串，如果存在返回true，否则返回 false。
bool QString::contains(const QString &str, Qt::CaseSensitivity cs = ...) const
默认大小写敏感的。
判断是否包含外，还想确定子串的位置，使用 indexOf 函数
int QString::indexOf(const QString &str, int from = 0, Qt::CaseSensitivity cs = Qt::CaseSensitive) const
返回 字串 在 原串 中的位置下标，如果不存在子串，则返回 -1，从字符串的头开始定位，如果存在多

个，返回出现的第一个下标。
`lastIndexOf` 则反向定位从字符串的尾开始定位，如果不存在子串，则返回 -1。

7. 截取

从哪个位置开始(下标从0开始) 截取多长，如果位置越界了则返回空，如果长度越界了则截取有效字符
```
QString QString::mid(int position, int n = -1) const
```
//从左边截取n个字符(有效字符)
```
QString QString::left(int n) const
```
//从右边截取n个字符(有效字符)
```
QString QString::right(int n) const
```

8.统计

//返回字符串中字符数量
str.size()： Equivalent to length()
str.length()： Equivalent to size()
str.count()：统计某个字符或字符串出现的次数

9. 去除空白符

空白符 包括 空格（' '|32），制表符（\t| 9），和回车换行符（\r\n| 13,10）。
trimmed()：去除字符串两边的空字符
simplified()：将一个或多个内部空白符替换为**单个空格**，同时去掉两端的空白符。
注意：字符串的 空格、\n、\r 等不管多少个都会被替换为一个空格。
simplified 和 trimmed 函数并不是直接作用于字符串本身，所以我们一般需要返回值。

10. 分割

`split` 将字符串按照指定的 **子串**进行拆分，返回字符串数组`QStringList`。
常用 按照 空格进行拆分或路径拆分等。

3. MessageBox

1 简单提示对话框

消息对话框QMessageBox 用于显示 提示、警告、错误等信息，或者提供确认选择等。
最简单的提示框 information（静态函数），需要 `#include<QMessageBox>`。

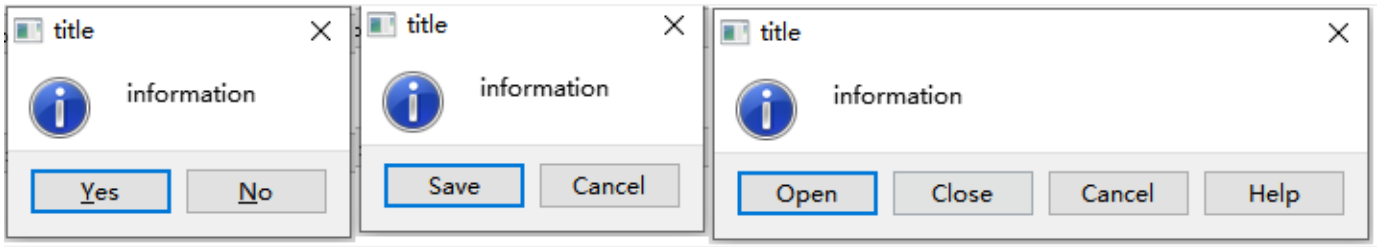
```
static StandardButton information(
    QWidget *parent,   //对话框得父窗口，可传空，如果指定了对话框自动显示到父窗口上方居中位置
    const QString &title,  //对话框标题字符串
```

```
    const QString &text,   //显示得信息字符串
    StandardButtons buttons = Ok, // 按钮，默认OK
    StandardButton defaultButton = NoButton //选择的按钮，默认不选择任何按钮
);
```

当QMessageBox 对话框显示时，其他的窗口不可操作，它是一个阻塞函数，就意味着，关闭它之前后面的代码不能被执行。

StandardButtons 可指定的枚举值包含在QMessageBox类中，如 QMessageBox::Yes。
不同的 StandardButton 改变是button上的文字而已，其他的样式都是一样的。对于以上 StandardButton 我们可以任意多个组合。

```
    QMessageBox::information(this,"title","information",QMessageBox::Yes|QMessageBox::No);
    QMessageBox::information(this,"title","information",QMessageBox::Save|QMessageBox::Cancel);
    QMessageBox::information(this,"title","information",QMessageBox::Open|QMessageBox::Close|QMessageBox::Cancel|QMessageBox::Help);
```



对于指定了多个button，如果我们想改变选中的焦点，就需要指定第四个参数了（defaultButton），如果要判断点击了哪一个按钮，需要返回值(StandardButton)。

常用的除了 information 还有 warning,critical,about， question等。
warning 和 critical 与 information 用法基本差不多。
但是about 与上面的三个不同，他不能指定按钮，只有一个默认的 Ok，所以也没有返回值。函数原型如下：

```
static void about(QWidget *parent, const QString &title, const QString &text);
```

question 确认选择对话框，默认提供了 Yes 和 No 按钮，这是与其他对话框不同的地方，原型如下：

```
static StandardButton question(QWidget *parent, const QString &title,const QString &text,
StandardButtons buttons = StandardButtons(Yes | No),StandardButton defaultButton = NoButton);
```