11/14/13

(Case Insensitive) String Comparison in C++

Adrian Winterstein

dkfz.
DEUTSCHES KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

Compare Strings for Equality

```
std::string string1 = "Foo";
std::string string2 = "Bar";
```

```
if ( string1.compare(string2) )
  MITK_INFO << "Strings are equal.";
```

```
if ( string1.compare(string2) == 0 )
  MITK_INFO << "Strings are equal.";
```

compare() returns 0 if strings are equal.
Greater / lower relation for non-equal strings.

```
if ( string1 == string2 )
  MITK_INFO << "Strings are equal.";
```

Use ==operator if only testing for equality.

Substring Comparison

```
std::string str1 ("Something is hidden here.");
std::string str2 ("hidden");
```

Find first occurence in the string:

get last position instead with rfind()

```
size_t pos = str1.find(str2)
if ( pos != std::string::npos )
  MITK_INFO << "Substring found at position " << pos << ".";
```

Test for a substring at a specific position:

position & length

```
if ( str2 == str1.substring(13, 6) )
  MITK_INFO << "Substring " << str2
            << " found at the expected position.";
```

Find characters in strings:

`find_first_of`, `find_last_of`, `find_first_not_of`, `find_last_not_of`

Case Insensitive Comparison

```
std::string str1 = "foo";
std::string str2 = "Foo";
```

How to compare them case insensitive?
The ==operator will not match.

Boost offers a solution:

```
if ( boost::iequals(str1, str2) )
  MITK_INFO << "Strings are equal.";
```

Case Insensitive Comparison: Standard C++

```
std::string str1 = "foo";
std::string str2 = "Foo";
```

How to compare them case insensitive?
The ==operator will not match.

What about standard C++?

```
std::locale locale;

bool equal = str1.size() == str2.size();
for ( unsigned int n = 0; n < str1.size() && equal; ++n )
{
  if ( std::toupper(str1[n], locale) != std::toupper(str2[n], locale) )
    equal = false;
}

if ( equal )
  MITK_INFO << "Strings are equal.";
```

Locale

Set of features which differ for different cultures:
decimal point, date order, currency symbol, ...

```
a -> A ?
A == À ?
```

---

default constructor
uses global locale

```
std::locale myLocale;
```

```
std::locale::classic();
```

which is classic („C“)
locale by default

global locale can be changed

```
std::locale::global(std::locale("en_US"));
```

Global Locale in MITK

should **not** be used arbitrarily

```
$ git grep "std::locale::global"
Core/Code/IO/mitkPointSetReader.cpp:
                std::locale::global(std::locale("C"));
Modules/ContourModel/IO/mitkContourModelReader.cpp:
                std::locale::global(std::locale("C"));
Modules/ContourModel/IO/mitkContourModelSetReader.cpp:
                std::locale::global(std::locale("C"));
```

Case Insensitive Comparison (2)

Qt

```
QString str1("bar");
QString str2("Bar");

if ( str1.compare(str2, Qt::CaseInsensitive) == 0 )
  MITK_INFO << "Strings are equal.";
```

Poco

```
#include "Poco/String.h"

std::string str1 = "bar";
std::string str2 = "Bar";

If ( Poco::icompare(str1, str2) == 0 )
  MITK_INFO << "Strings are equal.";
```

`PACKAGE_DEPENDS Poco` in `CmakeLists.txt` is necessary

Conclusion

Don't mistake compare() for ==operator.

Case insensitive string comparison:

- Available in many libraries: Boost, Qt, Poco
- Just a few lines to write on your own

Locales can be constructed using global default

- One should not change the global locale without strong reasons