Help Topic: MOOS-IvP String Utilities

Spring 2020

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

MOOS-IvP String Utilities

The below describe a set of string utilities in the `MBUtils` library distributed with MOOS-IvP. To use them, add `#include "MBUtils.h"` in your source code, and add the `mbutil` library to the list of libraries your code links to, likely in your local `CMakeLists.txt` file.

These utilities are merely convenience functions. Often the C++ equivalent is either multiple lines of code, or relatively obtuse to read. In any event you're welcome to take a look in `moos-ivp/ivp/src/lib_mbutil/MBUtils.cpp` to check out their implementation.

The `stripBlankEnds()` function

The `stripBlankEnds()` function takes an STL string argument and returns another STL string with leading and trailing white space removed. White space is defined by either a blank space (ASCII 32) or a tab (ASCII 09). The value of the string argument remains unchanged.

```
string stripBlankEnds(string);
```

For example:

```
string original = "   Hello World! ";
string modified = stripBlankEnds(original);
cout << "original: [" << original << "]" << endl;
cout << "modified: [" << modified << "]" << endl;
```

Produces:

```
original: [    Hello World! ]
modified: [Hello World!];
```

The `doubleToString()` function

The `doubleToString()` function converts an argument of type `double` to an STL string. It takes an optional integer argument which in indicates the number of significant digits desired in the resulting string. If the second argument is left out, the default is 2.

```
string doubleToString(double, int=2);
```

For example:

```
double val  = 1234.56789;
string str1 = doubleToString(val);
string str2 = doubleToString(val, 4);
cout << "str1: " << str1 << endl;
cout << "str2: " << str2 << endl;
```

Produces:

```
str1: 1234.56
str2: 1234.5678
```

The `intToString()` function

The `intToString()` function converts an argument of type `int` to an STL string.

```
string intToString(int);
```

For example:

```
int    val = -198;
string str = intToString(val);
cout << "str: " << str << endl;
```

Produces:

```
str: -198
```

The `uintToString()` function

The `uintToString()` function converts an argument of type `unsigned int` to an STL string.

```
string intToString(unsigned int);
```

For example:

```
unsigned int val = 177;
string str = uintToString(val);
cout << "str: " << str << endl;
```

Produces:

```
str: 177
```

The `ulintToString()` function

The `ulintToString()` function converts an argument of type `unsigned long int` to an STL string.

```
string ulintToString(unsigned long int);
```

For example:

```
unsigned long int val = 18446744073709551615;
string str = ulintToString(val);
cout << "str: " << str << endl;
```

Produces:

```
str: 18446744073709551615
```

The `boolToString()` function

The `boolToString()` function converts an argument of type `bool` to an STL string.

```
string boolToString(bool);
```

For example:

```
bool   val = true;
string str = boolToString(val);
cout << "str: " << str << endl;
cout << "str: " << val << endl;
```

Produces:

```
str: true
str: 1
```

The `strBegins()` function

The `strBegins()` function determines if the first given STL string begins with second STL string. An optional third Boolean argument indicates whether the test should be case sensitive. This third argument defaults to `true`.

```
bool strBegins(string, string, bool=true);
```

For example:

```
string str = "Hello World!";

bool result1 = strBegins(str, "Hello");
bool result2 = strBegins(str, "hello");
bool result3 = strBegins(str, "hello", false);

cout << "result1: " << boolToString(result1) << endl;
cout << "result2: " << boolToString(result2) << endl;
cout << "result3: " << boolToString(result3) << endl;
```

Produces:

```
result1: true
result2: false
result3: true
```

The `strEnds()` function

The `strEnds()` function determines if the first given STL string ends with second STL string. An optional third Boolean argument indicates whether the test should be case sensitive. This third argument defaults to `true`.

```
bool strEnds(string, string, bool=true);
```

For example:

```
string str = "Hello World!";

bool result1 = strBegins(str, "World!");
bool result2 = strBegins(str, "world!");
bool result3 = strBegins(str, "world!", false);

cout << "result1: " << boolToString(result1) << endl;
cout << "result2: " << boolToString(result2) << endl;
cout << "result3: " << boolToString(result3) << endl;
```

Produces:

```
result1: true
result2: false
result3: true
```

The `strContains()` function

The `strContains(string, string)` function determines if the first given STL string contains the second STL string. Pattern matching is case sensitive.

```
bool strContains(string, string);
```

For example:

```
string str = "Hello World!";

bool result1 = strContains(str, "World");
bool result2 = strContains(str, "hello");

cout << "result1: " << boolToString(result1) << endl;
cout << "result2: " << boolToString(result2) << endl;
```

Produces:

```
result1: true
result2: false
```

The `strContains()` function

The `strContains(string, char)` function determines if the given STL string contains the given character. Pattern matching is case sensitive.

```
bool strContains(string, char);
```

For example:

```
string str = "Hello World!";

bool result1 = strContains(str, 'W');
bool result2 = strContains(str, 'h');

cout << "result1: " << boolToString(result1) << endl;
cout << "result2: " << boolToString(result2) << endl;
```

Produces:

```
result1: true
result2: false
```