

C++ Tricks

by Randy Charles Morin

This is just a small note to explain some of the new functions that I've added to the kbcafe C++ library of late. You can get the latest copies of the files of this library at http://ca.msusers.com/KBCafecom/files.msnw?fc_p=%2Fkbcafeolib&fc_a=0. The files in the kbcafeolib.zip file do not contain the changes that I mention herein. You should download the three latest files separate.

- Replace.h
- trim.h
- convert.h

1 Find and Replace

First, I notice that the Replace.h file was in error, so I corrected it. Now the findandreplace function works more as expected. I must have been drinking when I originally wrote that function.

Listing Error! Bookmark not defined.: findandreplace template function

```
template
void findandreplace( T& source, const T& find, const T& replace )
{
    size_t j;
    for ( ; (j = source.find( find )) != T::npos; )
    {
        source.replace( j, find.length(), replace );
    }
}
```

The template functions works with a collection. It takes a source collection and replaces any occurrences of one string in the collection with another. So, if you were using this on the string class, the function prototype would be as follows.

Listing Error! Bookmark not defined.: findandreplace with STL's string

```
void findandreplace( std::string & source,
    const std::string & find, const std::string & replace );
```

I only noticed the problem in Replace.h when I was writing a utility to read the contents of a file, do a search and replace on two strings and finally output the result to the standard output. I uploaded this utility to the kbcafeutil folder http://ca.msusers.com/KBCafecom/files.msnw?fc_p=%2Fkbcafeutils&fc_a=0. You can download the utility and the source code from this folder.

This utility shows a great candidate usage of the findandreplace function. Lets examine one function in this utility.

Listing Error! Bookmark not defined.: ProcessFile

```
void ProcessFile()
```

```

{
    std::ifstream file;
    file.open(filename.c_str());
    if (!file.is_open())
    {
        std::cerr << "Cannot Find File " << filename << std::endl;;
        return;
    }

    std::stringstream input;

    char buffer[2] = {0};
    while (file)
    {
        file.read(buffer, 1);
        input << buffer;
    }

    std::string str;
    str = input.str();
    kbcafe::findandreplace(str, searchtext, replacetext);

    std::cout << str << std::endl;
}

```

This function opens a file, reads its entire contents into an STL string, then runs the search and replace on that string. Finally the string is output to the standard output. In most cases, this will be the console window.

2 Trim

The second change to the library is the addition of a new file that simulates Visual Basic functions TRIM, LTRIM and RTRIM. These functions work with either of the STL string classes, i.e. string and wstring.

Listing Error! Bookmark not defined.: Trim Functions

```

// trim_right() family.
inline std::string trim_right ( const std::string & source ,
    const std::string & t = " " )
{
    std::string str = source;
    return str.erase ( str.find_last_not_of ( t ) + 1 ) ;
}

inline std::wstring trim_right ( const std::wstring & source ,
    const std::wstring & t = L" " )
{
    std::wstring str = source;
    return str.erase ( str.find_last_not_of ( t ) + 1 ) ;
}

// trim_left() family.
inline std::string trim_left ( const std::string & source ,
    const std::string & t = " " )
{
    std::string str = source;
    return str.erase ( 0 , source.find_first_not_of ( t ) ) ;
}

inline std::wstring trim_left ( const std::wstring & source ,
    const std::wstring & t = L" " )
{
    std::wstring str = source;
    return str.erase ( 0 , source.find_first_not_of ( t ) ) ;
}

// trim() family.

```

```

inline std::string trim ( const std::string & source ,
    const std::string & t = " " )
{
    std::string str = source;
    return trim_left ( trim_right ( str , t ) , t ) ;
}

inline std::wstring trim ( const std::wstring & source ,
    const std::wstring & t = L" " )
{
    std::wstring str = source;
    return trim_left ( trim_right ( str , t ) , t ) ;
}

```

I wish I could have templated these functions, but I found too many issues related to the second parameter. Basically, I had to template the second parameter to present the wide default space versus the 8-bit default space. This causes the template to be more difficult to use than I'd hoped. So rather than let the end user deal with that complexity, I just wrote the six functions without using templates.

3 Convert

The last change was the addition of the convert.h header file. This file presents three functions for converting to different string representations of decimals, hexadecimals and binary. I could have written more, but the requirements were only for these three at the time.

I needed a function to convert a binary string to a decimal string and vice versa.

Listing Error! Bookmark not defined.: Binary and Decimal

```

std::string binarytodecimal(const std::string & rhs)
{
    long sum = 0;

    for (int i=0;i <rhs.length();i++)
    {
        if (rhs[i] == '0')
        {
            sum *= 2;
            continue;
        } else
        if (rhs[i] == '1')
        {
            sum = (sum*2) + 1;
            continue;
        }
        else
        {
            break;
        }
    }

    std::stringstream ss;
    ss << sum;
    return ss.str();
};

std::string decimaltobinary(const std::string & rhs)
{
    std::stringstream ss;
    ss << rhs;
    long l;
    ss >> l;

    char buffer[256];

```

```
        ::itoa(1, buffer, 2);  
        return buffer;  
};
```

I also needed a way of representing a number in hex format in a string. One way of doing this would be to send it to a stream object using the hex manipulator, and then convert it back to a string object.

Listing Error! Bookmark not defined.: Hex Streams

```
#include <string>  
#include <sstream>  
#include <iostream>  
  
int main(int argc, char* argv[])  
{  
    std::stringstream ss;  
    ss << std::hex << 123;  
    std::string str = ss.str();  
    std::cout << str;  
    return 0;  
}
```

But I'd hate to do this each time I wanted to do the conversion. Rather, I wrote a function to do the same.

Listing Error! Bookmark not defined.: Hex String

```
std::string outputashex(unsigned long l)  
{  
    char buffer[1024];  
    ::itoa(l, buffer, 16);  
    return buffer;  
};
```

I used the C run-time library itoa function to perform the conversion. I could also have used a stringstream and the hex manipulator to the same affect. I would suspect the later would have been a better choice. I might make that change in the near future.

Note, that I couldn't find a manipulator that would allow me to represent an integer in binary format on a stream. I'm a little confused on why this would be. If anybody knows why, then I'd be glad to hear it. Or for that matter, is there something that I missed?

About the Author

Randy Charles Morin is the Lead Architect of SportMarkets Development from Toronto, Ontario, Canada and lives with his wife and two kids in Brampton, Ontario. He is the author of the www.kbcafe.com website, author of Wiley's Programming Windows Services book and co-author of many other programming books and articles.