Class CommandLine

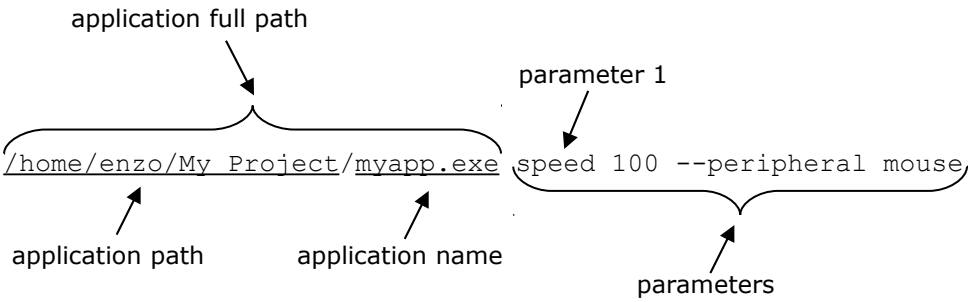
Propose: C++ class to handle data from the command line.

Version 2.0.0
Enzo Roberto Verlato - enzover@ig.com.br
https://github.com/FreeSource

Supported and tested platforms:

O.S.	Compiler	Make
WindowsXP SP2	MinGW gcc 4.6.1	gmake 3.82
Linux openSUSE 11.4 / 12.2	gcc 4.5.1 / 4.7.1	gmake 3.82
OpenIndiana 151a	gcc 3.4.3	gmake 3.81
FreeBSD 9.0	gcc 4.2.1	gmake 3.82
Solaris 9 / 10	gcc 3.3.2 / 3.4.6	gmake 3.80 / 3.81
Mac OS X 10.8.2	gcc 4.2.1	gmake 3.81

The standard structure of a command line:



Members:

```
string getCommandLine()

string getApplicationName()
string getApplicationPath()
string getApplicationFullPath()
string getCurrentWorkingDirectory()

bool hasParameters()
bool hasParameter( int parameterPosition )

int getParametersNumber()
string getAllParameters()
string getParameter( int parameterPosition )

gotoFirstParameter()
bool gotoNextParameter()
int getCurrentPosition()

string getCurrentParameter()
```

```
string getFirstParameter()
string getLastParameter()

setOptionPrefix( string optionPrefix )
setOptionPostfix( string optionPostfix )

string getOptionPrefix()
string getOptionPostfix()

bool hasOption( string option )

string getOptionValue( string option )
string getOptionLongValue( string option )

optionCaseSensitive()
optionCaseInsensitive()
bool isOptionCaseSensitive()
```

`string getCommandLine()` ⬆
Description: Retrieves the command line string for the current process.

Example:

```
#include <CommandLine.h>

#include <windows.h>
#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
PSTR szCmdLine, int iCmdShow ) {
    try {
        util::CommandLine commandLine;
        cout << commandLine.getCommandLine() << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp My first example
/home/enzo/myapp My first example
```

`string getApplicationName()` ⬆
Description: Retrieves the name of the application for the current process.

Example:

```
#include <CommandLine.h>
```

```cpp

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main( int argc, char *argv[] ) {
    try {
        util::CommandLine commandLine;
        cout << commandLine.getApplicationName() << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp
myapp
```

`string getApplicationPath()` ⬆
Description: Retrieving the application path of the current process, not including the name of the program itself.

Example:

```cpp
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        cout << commandLine.getApplicationPath() << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp
```

```
/home/enzo
```

`string getApplicationFullPath()` ⬆
Description: Retrieving the application path of the current process, including the name of the program itself.

Example:

```cpp
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        cout << commandLine.getApplicationFullPath() << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp
/home/enzo/myapp
```

`string getCurrentWorkingDirectory()` ⬆
Description: Retrieves the current working directory for the current process.

Example:

```cpp
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        cout << commandLine.getApplicationPath() << endl;
        cout << commandLine.getCurrentWorkingDirectory() << endl;
        return EXIT_SUCCESS;
    }
```

```
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux-hevv:/home/enzo/CommandLine/main # /home/enzo/myapp
/home/enzo
/home/enzo/CommandLine/main
```

---

bool hasParameters() 🡅
Description: Checks if the command line has parameters.

Example:

```
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        if( commandLine.hasParameters() ) {
            cout << "yes" << endl;
        }
        else {
            cout << "no" << endl;
        }
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp parameter1 parameter2 parameter3
yes
```

---

bool hasParameter(**int** parameterPosition) 🡅
Description: Checks if a specified parameter exists.

Example:

```
#include <CommandLine.h>

#include <iostream>
```

```cpp
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        if( commandLine.hasParameter( 2 ) ) {
            cout << "yes" << endl;
        }
        else {
            cout << "no" << endl;
        }
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp How To Pass Parameters To Main() And Use Them
yes
```

`int getParametersNumber()` ⬆

Description: Returns the total number of parameters on the command line for the current process, not including the name of the program itself.

Example:

```cpp
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        cout << commandLine.getParametersNumber() << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp The C++ Programming Language
4
```

`string getAllParameters()` ⬆
Description: Retrieves all the parameters on the command line for the current process.

Example:

```
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        cout << commandLine.getAllParameters() << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp High thoughts must have high language
High thoughts must have high language
```

`string getParameter( int parameterPosition )` ⬆
Description: Retrieves the parameter of the specified parameter position on the command line for the current process.

Example:

```
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        cout << commandLine.getParameter( 2 ) << endl;
```

```
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp C++ evolved from C
evolved
```

`gotoFirstParameter()` ⬆

Description: Points to the first parameter on the command line for the current process.

Example:

```
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        commandLine.gotoNextParameter();
        cout << commandLine.getCurrentParameter() << endl;
        commandLine.gotoFirstParameter();
        cout << commandLine.getCurrentParameter() << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp Principles and Practice using C++
and
Principles
```

---

bool `gotoNextParameter()` ⬆

Description: Points to the next parameter on the command line for the current process. Returns false if is at the last parameter.

Example:

```
#include <CommandLine.h>
```

```cpp

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        do
            cout << commandLine.getCurrentParameter() << endl;
        while( commandLine.gotoNextParameter() );
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp GCC the GNU Compiler Collection
GCC
the
GNU
Compiler
Collection
```

`int getCurrentPosition()` ⬆

Description: Retrieves the current position parameter on the command line for the current process.

Example:

```cpp
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        do
            cout << commandLine.getCurrentPosition() << endl;
        while( commandLine.gotoNextParameter() );
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
```

```
    }
}

```

Output:

```
linux:/home/enzo # ./myapp Principles and Practice using C++
1
2
3
4
5
```

`string getCurrentParameter()` ⬆
Description: Retrieves the current parameter on the command line for the current process.

Example:

```
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        do
            cout << commandLine.getCurrentParameter() << endl;
        while( commandLine.gotoNextParameter() );
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp Principles and Practice using C++
Principles
and
Practice
using
C++
```

`string getFirstParameter()` ⬆
Description: Retrieves the first parameter on the command line for the current process.

Example:

```
#include <CommandLine.h>

```

```cpp
#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        cout << commandLine.getFirstParameter() << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp C++ Development Environment
C++
```

`string getLastParameter()` ⬆

Description: Retrieves the last parameter on the command line for the current process.

Example:

```cpp
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        cout << commandLine.getLastParameter() << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp The C++ Standard Library
Library
```

`setOptionPrefix( string optionPrefix )`
Description: Define the prefix (string added in front of the option name) used to recognize an option on the command line.

Example:

```
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        commandLine.setOptionPrefix( "--" );
        cout << commandLine.getOptionValue( "price" ) << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp --price 0.99
0.99
```

`setOptionPostfix( string optionPostfix )`
Description: Define the postfix (string added to the end of the option name) used to recognize an option on the command line.

Example:

```
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        commandLine.setOptionPostfix( "=" );
        cout << commandLine.getOptionValue( "price" ) << endl;
        return EXIT_SUCCESS;
    }
```

```
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp price=0.99
0.99
```

`string getOptionPrefix()` ⬆

Description: Returns the prefix (string added in front of the option name) used to recognize an option on the command line.

Example:

```
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        commandLine.setOptionPrefix( "--" );
        cout << commandLine.getOptionPrefix() << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp
--
```

`string getOptionPostfix()` ⬆

Description: Returns the postfix (string added to the end of the option name) used to recognize an option on the command line.

Example:

```
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

```

```cpp
using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        commandLine.setOptionPostfix( "=" );
        cout << commandLine.getOptionPostfix() << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp
=
```

---

bool hasOption(string option) ⬆

Description: Checks if the specified option exists.

Example:

```cpp
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        commandLine.setOptionPostfix( ":" );
        if( commandLine.hasOption( "ISBN-10" ) ) {
            cout << "yes" << endl;
        }
        else {
            cout << "no" << endl;
        }
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp Paperback: 208 pages Publisher: O'Reilly
Media; 1 edition (August 19, 2011) Language: English ISBN-10: 1449397670
Weight: 14.4 ounces
yes
```

`string getOptionValue( string option )` ⬆

Description: Retrieves the value of the specified option on the command line for the current process.

Example:

```cpp
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        commandLine.setOptionPostfix( ":" );
        cout << commandLine.getOptionValue( "Language" ) << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp Paperback: 208 pages Publisher: O'Reilly
Media; 1 edition (August 19, 2011) Language: English ISBN-10: 1449397670
Weight: 14.4 ounces
English
```

`string getOptionLongValue( string option )` ⬆

Description: Retrieves the long value of the specified option (a range of parameters delimited by the next option if it exists) on the command line for the current process.

Example:

```cpp
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
```

```
    try {
        util::CommandLine commandLine;
        commandLine.setOptionPrefix( "--" );
        cout << commandLine.getOptionLongValue( "peripheral" );
        cout << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp --speed 100 --peripheral mouse display
keyboard --price 1000
mouse display keyboard
```

`optionCaseSensitive()` ⬆

Description: Differ use of uppercase and lowercase letters on the option parameter for the other functions. Option parameter is case sensitive by default.

Example:

```
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        commandLine.optionCaseSensitive();
        commandLine.setOptionPrefix( "--" );
        cout << commandLine.getOptionLongValue( "PERIPHERAL" );
        cout << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp --peripheral mouse

```

`optionCaseInsensitive()` ⬆

Description: No differ use of uppercase and lowercase letters on the option parameter for the other functions.

Example:

```
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        commandLine.optionCaseInsensitive();
        commandLine.setOptionPrefix( "--" );
        cout << commandLine.getOptionLongValue( "PERIPHERAL" );
        cout << endl;
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp --peripheral mouse
mouse
```

---

bool isOptionCaseSensitive() 🡅
Description: Checks if the options are case sensitive.

Example:

```
#include <CommandLine.h>

#include <iostream>
#include <cstdlib>
#include <stdexcept>

using std::cout;
using std::endl;
using std::string;
using std::runtime_error;

int main() {
    try {
        util::CommandLine commandLine;
        if( commandLine.isOptionCaseSensitive() ) {
            cout << "yes" << endl;
        }
        else {
            cout << "no" << endl;
```

```
        }
        return EXIT_SUCCESS;
    }
    catch ( runtime_error &error ) {
        cout << "Exception occurred: " << error.what() << endl;
        return EXIT_FAILURE;
    }
}

```

Output:

```
linux:/home/enzo # ./myapp
yes
```