C++ course project report

Janne Ronkonen

07 December 2012

1 The problem

The problem was to design, implement and test a String-class with all the normal functionality expected from it, include iterators, indexing, adding and deleting characters and substrings etc.

2 Design decisions

2.1 The class itself

The class members consists of char pointer to the beginning of the buffer (m_buf) that holds the actual contents of the string, and two size_t:s that are used to keep track of the size of the buffer (m_bufsize), and the logical size of the string (m_used). The m_used may be null if the string has no characters. In hindsight it would've probably been more consistent to never allow m_buf to be null.

2.2 The testdriver

The testdriver consists of a bunch of functions that take no arguments and return no value. A test is succesful if the function returns normally, and test fails if the function throws an exception. The driver's main() -function calls each of the test-functions and records how many of them failed and succeeded, and also records the names of the functions that failed, and prints this info into std::cout.

3 Challenges during the project

3.1 Getting it to compile

One problem was of course getting the whole thing to compile in the first place. Error messages from the g++ compiler aren't always very helpful, and finding out the "true" cause of an error that prevents compilation isn't always as trivial as one might think it is.

3.2 Keeping track of dynamically allocated memory correctly

One of the more difficult things was keeping track of the dynamically allocated buffers of memory and not leaking any memory. Thankfully C++ idioms such as RAII and using constructors, destructors and assignment-operator make this task somewhat simpler. One particular problem regarding memory was the resizing of the buffer found in push_back() and pop_back() -methods.

3.3 Implementation of the IO-operators

Because of the statefulness of IO-operations, I found them to be the most challenging to implement.

3.4 Maintaining separate header and implementation files by hand

Because the project-specification required separate header and implementation files, it became something of a problem to keep the header and implementation in synch by hand. Since this project is of trivial size, I figure there must be some better method of doing this sort of mundane task in bigger projects that may have orders of magnitude more files.

3.5 Testing

Since I was both the tester and the implementor of the class, I found it somewhat difficult to write good tests since I already was thinking how I would implement the tested functionality, and this in turn influenced how and what I would test. I feel that one cannot truly write comprehensive and exacting tests for his/her own code because of this phenomenom, and it would be better to separate these responsibilities in bigger and/or more important projects.

4 Errors, limitations and shortcomings

- the input-operator doesn't work correctly when used with std::stringstreams, but seems to work with other streams
- some of the functionality and/or code paths may not be tested
- the formatting of the code could be better
- the code could be commented better