The std::string Class

Chris Wild & Steven Zeil

May 25, 2013

Contents

1 Description

- One of the most commonly used data types is the character string.
- In C++, string is NOT a built-in data type.
- The string class supports the following member functions
  - non-default constructor which allows definition from a string literal or another "string" object
  - length function
  - append function to add to the end of a string
  - overload to the "+" operator to concatenate two strings
  - overload to the relational operators for string comparison
  - A variety of find... functions for searching through strings.
  - A substr function to extract portions of a string.

2 Example

```
#include <string>
#include <iostream>

using namespace std;

int main()
{
    string firstName, lastName, fullName;
    string greeting("Hello ");
```


```
    cout << "What is your name (first last separated by a space)? ";
    cin >> firstName >> lastName;
    greeting.append(lastName);
    greeting.append(", " + firstName);
    string banner(greeting.length() + 4,'$'); // construct string with bunch of '$'s
    cout << banner << endl;
    cout << "$ " << greeting + " $" << endl;
    cout << banner << endl << endl;
    if(firstName < lastName)
        cout << "your first name is alphabetically before your last\n";
    else
        cout << "your first name is alphabetically after your last\n";
    return 0;
}
```

Sample Execution

```
What is your name (first last separated by a space)? <b>chris wild</b>
$$$$$$$$$$$$$$$$$$$$
$ Hello wild, chris $
$$$$$$$$$$$$$$$$$$$$

your first name is alphabetically before your last
```

3 Tips - I/O

There are three approaches used for reading strings.

1. Read characters until a whitespace character is found
   - (Whitespace characters are: blank, tab and new line).
   - Use the regular *istream* extraction operator ('>>') for this purpose.
     - This technique is illustrated in the earlier program, where the strings *firstName* and *lastName* are read.

- Initial whitespace characters are ignored and a ***null termination*** character is placed after the last character read.
- One problem with this approach is that you cannot read a string with blank characters in it.

2. Read until the end of line. The following program should produce the same output as the first program.

```
#include <string>
#include <iostream>

using namespace std;

int main()
{
    string firstName, lastName, fullName;
    string greeting("Hello ");


    cout << "What is your name? ";
    getline (cin,  fullName);
    if (fullName.size() > 0 && fullName[0] == ' ')
       { // Trim leading blanks from fullName
         int charPosition = fullName.find_first_not_of (" ");
         fullName = fullName.substr(charPosition);
       }

    // Split fullName into parts
    int blankPosition = fullName.find(' ');
    if (blankPosition != string::npos)
      {
        firstName = fullName.substr (0, blankPosition);
        int charPosition = fullName.find_first_not_of (" ", blankPosition);
        lastName = fullName.substr (charPosition);
```


```
      }
    else
        lastName = fullName;

    greeting.append(lastName);
    greeting.append(", " + firstName);
    string banner(greeting.length() + 4,'$'); // construct string with bunch of '$'s
    cout << banner << endl;
    cout << "$ " << greeting + " $" << endl;
    cout << banner << endl << endl;
    if(firstName < lastName)
        cout << "your first name is alphabetically before your last\n";
    else
        cout << "your first name is alphabetically after your last\n";
    return 0;
}
```

- Use the function `getline` for this purpose.
  `getline` will put all characters into the string including blanks and tabs (but not newline characters).

3. Read until a special character is found.

```
#include <string>
#include <iostream>

using namespace std;

int main()
{
    string firstName, lastName, fullName;
    string greeting("Hello ");
```


```
    cout << "What is your name (first last separated by a space)? ";
    getline (cin, firstName, ' ');
    getline (cin, lastName);
    cin >> firstName >> lastName;
    greeting.append(lastName);
    greeting.append(", " + firstName);
    string banner(greeting.length() + 4,'$'); // construct string with bunch of '$'s
    cout << banner << endl;
    cout << "$ " << greeting + " $" << endl;
    cout << banner << endl << endl;
    if(firstName < lastName)
        cout << "your first name is alphabetically before your last\n";
    else
        cout << "your first name is alphabetically after your last\n";
    return 0;
}
```

- The special character is passed as the optional third parameter of the `getline` function.
- This code is simpler, but more "fragile" than the version before. If the person actually types blanks before their first name, we are in trouble. If the person types multiple blanks between the two parts of their name, we are likewise in trouble.
  - As a general rule, we have more control if we read an entire line and then use the string functions to extract what we want than we can get by relying on special characters to actually stop the input partway through a line.

- So which method of inputting strings is the best to use?
  - The answer depends on how much you know about your input. Use » when you know you want to skip leading whitespace and that you won't have whitespace inside the value you want to read.

  - Use `getline` when you want an entire line of data.
  - If you want a partial line of data that stops at something other than whitespace, use the 3-parameter form of `getline`.

- Remember always that » skips over leading whitespace and stops before (but does not consume) trailing whitespace. `getline` preserves leading whitespace and consumes (discards) its stopping character.