C strings vs. C++ strings  The standard string class is based on the standard `vector` template. C++ strings are easy to use and they manage their own memory automatically. Inside the C++ string is a C string, which can be extracted when needed.

There are a large number of functions defined for class string. Many of them imitate the functions in the `cstring` library. Most have at least two methods, one with a `string` parameter, and another with a `char*` or `const char*` parameter. This makes it very easy to mix code that uses literal quoted strings, character arrays, and C++ strings.

Using the STL string Class  You will need several of the following string functions and you should know about the others. There are many more functions for strings that are not mentioned here. This information is summarized from documentation at the website of cplusplus.com: www.cplusplus.com/reference/string/operators/

- Constructor prototypes:
  - `string ();` Make an empty string; content can be added later.
  - `string ( const string& s);` Allocate a new array for the chars and copy another C++ string.
  - `string ( const char * s );` Allocate a new array for the chars and copy a C string.

- There are two names for the function that returns a count of the number of characters in the string. They mean exactly the same thing:
  - `int size() const;`
  - `int length() const;`

- Assignments: assign new content to the string, replacing its current content.
  - `string& assign ( const string& str );` It works with a C++ string.
  - `string& assign ( const char* s );` It also works with a C string.
  - `void swap ( string& str );` Swap contents with another C++ string.

```cpp
#include <iostream>
#include <string>
using namespace std;

int main (void) {
    string buyer ("money");
    string seller;
    cout << " What do you want to buy? ";
    cin >> seller;
    cout << " Before swap, buyer has " << buyer;
    cout << " and seller has " << seller << endl;
    seller.swap (buyer);
    cout << " After swap, buyer has " << buyer;
    cout << " and seller has " << seller << endl;
    return 0;
}
```

Output:
What do you want to buy?  cabbage
Before swap, buyer has money and seller has cabbage
After swap, buyer has cabbage and seller has money
• Subscript is defined, and means the same thing as it does in C. There are two definitions: `const char& operator[] ( int pos ) const;` Read-only access to a constant string.
`char& operator[] ( int pos );` Read and write access to a non-constant string.

• Convert to a C string:
`const char* c_str ( ) const;` Extract the C string from inside the C++ string.

• String concatenation:
`string& operator+= ( const string& str );` Concatenate a C++ string to the end of this.
`string& operator+= ( const char* s );` Concatenate a C string to the end of this string.
`string& operator+= ( char c );` Concatenate a char to the end of this string.

• Searching: Search for the leftmost occurrence of any one of the chars given by the first parameter. Start searching at subscript 0 unless some other subscript is given by the second parameter. All of these variations return positions that can be used as subscripts. Like `strchr()`.
`int find_first_of ( const string& str, int pos = 0 ) const;` Search a C++ string.
`int find_first_of ( const char* s, int pos = 0 ) const;` Search a C string.
`int find_first_of ( char c, int pos = 0 ) const;` Search a char.

• Searching: Search for the rightmost occurrence of a char. The second parameter is the rightmost position that should be searched. It defaults to `npos`, the subscript of the end of the string. Like `strrchr()`.
`int find_last_of ( const string& str, int pos = npos ) const;` Search a C++ string.
`int find_last_of ( const char* s, int pos = npos ) const;` Search a C string.
`int find_last_of ( char c, int pos = npos ) const;` Search a char.

• Compare this string to another string. Return a negative number if this string is alphabetically smaller, 0 if they are equal, and a positive number if this is later in the alphabet. Like `strcmp()`.
`int compare ( const string& str ) const;` Compare a C++ string.
`int compare ( const char* s ) const;` Compare a C string.

• Global functions and operators that are defined for strings:
  - Input and output: `operator >>`, `operator <<` and `getline()`
  - Comparison operators `==`, `!=`, `<`, `>`, `<=`, `>=` These all use `string::compare()`.

// Demonstrate use of the constructor, find_first_of, comparison, and subscript.
#include <iostream>
#include <string>
using namespace std;

int main (void) {
{
  string str ("Replace the vowels in this sentence by asterisks.");
  int found;
  for (; ; ){
    found = str.find_first_of("aeiou");
    if (found == string::npos) break;
    str[found] = '*';
  }
  cout << str << endl;
  return 0;
}