

## C++ Strings – A Reference

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

```
string s1 = "hello";
string s2 = s1;           // s2 is "hello"
s1 = s1 + " there";     // s1 is "hello there"
```

OR

```
#include <string>
using std::string;
```

```
if (s1 == s2)           // are they equal?
cin >> s1;             // Read chars from keyboard into s1 until white space is encountered.
getline(cin, s1);      // Read chars from keyboard into s1 until \n is encountered.
```

Prototype	Example	Description
bool ← empty()	string a = "test"; if (a.empty()); // false	Returns true if nothing is in the string, false otherwise.
int ← length()	string a = "hello"; int len = a.length(); // 5	Returns the number of chars in the string. Same as size().
char ← at(int pos)	string a = "hello"; char c = a.at(0); // same as c = a[0] a.at(0) = 'H'; // a is "Hello"	Returns the char at position pos. Same functionality as using [pos] notation.
char * ← c_str()	string a = "hello"; char *p = a.c_str(); // "hello"	Returns a char array representing the string with a null terminator at the end.
int ← compare(string s)	string a = "test"; if (a.compare("test") == 0) // same	Compares the string to s, returns 0 if the string and s are equal, < 0 if the string is before s, and > 0 if after s. Just like < > ==.
int ← compare(int start, int numChars, string s)	string a = "compare test"; if (a.compare(2, 3, "yes test") == 0) // compare "mpa" to "yes"	Compares numChars chars starting at position start to string s.  Note: g++ compiler requires string to be first arg, not last.
erase(int start) erase(int start, int numChars)	string a = "erase this"; a.erase(0, 6); // Leaves "this" a.erase(2); // Leaves "th"	Erases all chars to the right of the starting position start. Erases numChars number of chars beginning at position start.
append(string s, int start, int numChars)	string a = "again"; a.append("state", 0, 2); // "against"	Appends numChars chars from s to the string, beginning at slot start in s.
resize(int size)	string a = "to"; a.resize(4); a[2] = "b"; a[3] = "e"; // a = "tobe"	Resizes the string so it can hold additional characters.
string ← substr(int start, int numChars)	string a = "a nice play"; string b = a.substr(2, 4); // "nice"	Return a substring starting at slot start with numChars chars.
int ← find(string s) int ← find(string s, int start) int ← rfind(string s) int ← rfind(string s, int start)	string a = "this is a test"; int pos = a.find("is"); // returns 2 pos = a.find("is", 3); // returns 5 pos = a.rfind("is"); // returns 5	Returns the position of the first occurrence of s (starting at position start) or string::npos if the string is not found.  Reverse find starts searching at back of string.
int ← find_first_of(string s) int ← find_last_of(string s)	string a = "this is a test"; pos = a.find_first_of("iou"); // 2 pos = a.find_last_of("iou"); // 5	Returns the first (or last) occurrence of any char in s or string::npos if none of the chars could be found.
int ← find_first_not_of(string s) int ← find_last_not_of(string s)	string a = "this is a test"; pos = a.find_first_not_of("ihst"); // 4 pos = a.find_last_not_of("ihst"); // 11	Returns first (or last) occurrence of any char not in s or string::npos if at least one of the chars in s could be found in every position of the string.
replace(int start, int numChars, string s)	string a = "hello there"; a.replace(0, 5, "stay"); // "stay there"	Replaces numChars chars starting at position start with s.
insert(int start, string s)	a = "this is a test"; a.insert(8, "not "); // "this is not a test"	Inserts s at position start in the string.
swap(string s)	string a = "one", b = "two"; a.swap(b); // a = "two", b = "one"	Swaps string s with the string.

\* Note: There are many variations of the functions above which are not displayed here.

```
// Prints all items in a string that are separated by a common delimiter.
```

```
// Example call:
```

```
void parse(string parseString, string delimiter)
{
    string value;
    int startPos = 0, pos = parseString.find(delimiter);
    while (pos != string::npos)
    {
        value = parseString.substr(startPos, pos - startPos);
        cout << value << endl;
        startPos = pos + delimiter.length();
        pos = parseString.find(delimiter, startPos);
    }
    value = parseString.substr(startPos, parseString.length() - startPos);
    cout << value << endl;
}
```

```
parse("this::is::a::test", "::");
```