String Member Functions

Constructors

```
string();
string(const string& s2); // Copy constructor
string(const string& s2, size_type pos2, size_type len2);
string(const char* nts);        // convert from C-string
string(const char* buf, size_type bufsize);
string(size_type repetitions, char c);
```

Assignment Operators and Functions

```
string& operator=(const string& s2); // normal assignment
string& operator=(const char* nts); // convert from C-string
string& operator=(char c); // assign a single character
string& assign(const string& s2);
string& assign(const string& s2, size_type pos2, size_type len2);
string& assign(const char* nts);
string& assign(const char* buf, size_type buflen);
string& assign(size_type repetitions, char c);
void swap(string& s2);
```

Iterator Functions

```
iterator               begin();
iterator               end();
const_iterator         begin() const;
const_iterator         end() const;
reverse_iterator       rbegin();
reverse_iterator       rend();
const_reverse_iterator rbegin() const;
const_reverse_iterator rend()   const;
```

Size and Capacity

```
size_type size() const;
size_type length() const; // same as size()
size_type max_size() const;
void resize(size_type size, char c = '\0');
void clear();
bool empty() const;
size_type capacity() const;
void reserve(size_type capacity = 0);
```

Element Access

```
char&       operator[](size_type pos);
const char& operator[](size_type pos) const;
char&       at(size_type pos);
const char& at(size_type pos) const;
```

Append Functions and Operators

```
string& append(const string& s2);
string& append(const string& s2, size_type pos2, size_type len2);
string& append(const char* nts);
```

```
string& append(const char* buf, size_type buflen);
string& append(size_type repetitions, char c);
string& operator+=(const string& s2);
string& operator+=(const char* nts);
string& operator+=(char c);
```

Insert Functions

```
string& insert(size_type pos1, const string& s2);
string& insert(size_type pos1, const string& s2,
               size_type pos2, size_type len2);
string& insert(size_type pos1, const char* nts);
string& insert(size_type pos1, const char* buf, size_type buflen);
string& insert(size_type pos1, size_type repetitions, char c);
iterator insert(iterator pos1, char c);
void     insert(iterator pos1, size_type repetitions, char c);
```

Erase Functions

```
string& erase(size_type pos = 0, size_type len = npos);
iterator& erase(iterator pos);
iterator& erase(iterator start, iterator finish);
```

Replace Functions

```
string& replace(size_type pos1, size_type len1, const string& s2);
string& replace(size_type pos1, size_type len1, const string& s2,
                size_type pos2, size_type len2);
string& replace(size_type pos1, size_type len1, const char* nts);
string& replace(size_type pos1, size_type len1,
                const char* buf, size_type buflen);
string& replace(size_type pos1, size_type len1,
                size_type repetitions, char c);
string& replace(iterator start, iterator finish, const string& s2);
string& replace(iterator start, iterator finish, const char* nts);
string& replace(iterator start, iterator finish,
                const char* buf, size_type buflen);
string& replace(iterator start, iterator finish,
                size_type repetitions, char c);
```

Comparison Functions

```
// compare function returns -1 if *this < s2, 0 if *this == s2
// and +1 if *this > s2.
int compare(const string& s2) const;
int compare(size_type pos1, size_type len1, const string& s2);
int compare(size_type pos1, size_type len1, const string& s2,
            size_type pos2, size_type len2);
int compare(const char* nts);
int compare(size_type pos1, size_type len1, const char* nts);
int compare(size_type pos1, size_type len1,
            const char* buf, size_type buflen);
```

Substrings

```
string substr(pos = 0, len = npos) const;
// This constructor is repeated here because it creates substring
string(const string& s2, size_type pos2, size_type len2);
```

Conversions

```
// This constructor is repeated here because it converts
// a C string to a C++ string.
string(const char* nts);
const char* c_str() const; // Returns a null-terminated string
const char* data() const;  // Not null-terminated
// Copy characters to an array. Does not append a null-terminator
size_type copy(char* buf, size_type bufsize, size_type pos1 = 0) const;
```

Search Functions

```
size_type find(const string& s2, size_type pos1);
size_type find(const char* nts, size_type pos1);
size_type find(const char* buf, size_type pos1, size_type bufsize);
size_type find(char c, size_type pos1);
size_type rfind(const string& s2, size_type pos1);
size_type rfind(const char* nts, size_type pos1);
size_type rfind(const char* buf, size_type pos1, size_type bufsize);
size_type rfind(char c, size_type pos1);
size_type find_first_of(const string& s2, size_type pos1);
size_type find_first_of(const char* nts, size_type pos1);
size_type find_first_of(const char* buf, size_type pos1, size_type
bufsize);
size_type find_first_of(char c, size_type pos1);
size_type find_last_of(const string& s2, size_type pos1);
size_type find_last_of(const char* nts, size_type pos1);
size_type find_last_of(const char* buf, size_type pos1, size_type bufsize);
size_type find_last_of(char c, size_type pos1);
size_type find_first_not_of(const string& s2, size_type pos1);
size_type find_first_not_of(const char* nts, size_type pos1);
size_type find_first_not_of(const char* buf, size_type pos1,
                            size_type bufsize);
size_type find_first_not_of(char c, size_type pos1);
size_type find_last_not_of(const string& s2, size_type pos1);
size_type find_last_not_of(const char* nts, size_type pos1);
size_type find_last_not_of(const char* buf, size_type pos1,
                           size_type bufsize);
size_type find_last_not_of(char c, size_type pos1);
```

String-Related Global Operators and Functions

The following functions and operators are related to the `string` class but are not members of `string`. Along with the `string` class, the following definitions are in the `<string>` header:

```
// String concatenation:
string operator+(const string& s1, const string& s2);
string operator+(const char* nts, const string& s2);
string operator+(char c, const string& s2);
string operator+(const string& s1, const char* nts);
string operator+(const string& s1, char c);
// Equal
bool operator==(const string& s1, const string& s2);
bool operator==(const char* nts, const string& s2);
bool operator==(const string& s1, const char* nts);
// Not-equal
bool operator!=(const string& s1, const string& s2);
bool operator!=(const char* nts, const string& s2);
bool operator!=(const string& s1, const char* nts);
// Ordering:
bool operator<(const string& s1, const string& s2);
bool operator<(const char* nts, const string& s2);
bool operator<(const string& s1, const char* nts);
// Greater than:
bool operator>(const string& s1, const string& s2);
bool operator>(const char* nts, const string& s2);
bool operator>(const string& s1, const char* nts);
// Less-than or equal-to:
bool operator<=(const string& s1, const string& s2);
bool operator<=(const char* nts, const string& s2);
bool operator<=(const string& s1, const char* nts);
// Greater-than or equal-to:
bool operator>=(const string& s1, const string& s2);
bool operator>=(const char* nts, const string& s2);
bool operator>=(const string& s1, const char* nts);
// Input and output
ostream& operator<<(ostream& os, const string& s);
istream& operator>>(istream& is, string& s);
istream& getline(istream& is, string& s, char delimiter = '\n');
```