C++ stl::string

Constructors

```
string();
string(const string& s);
string(size_type length, const char& ch);
string(const char* str);
string(const char* str, size_type length);
string(const string& str, size_type index, size_type length);
string(const_iterator start, input_iterator end);
~string();
```

Operators

```
bool operator  ==  (const string& c1, const string& c2);
bool operator!  =  (const string& c1, const string& c2);
bool operator  <  (const string& c1, const string& c2);
bool operator  >  (const string& c1, const string& c2);
bool operator  <=  (const string& c1, const string& c2);
bool operator  >=  (const string& c1, const string& c2);
string operator  +  (const string& s1, const string& s2 );
string operator  +  (const char* s, const string& s2);
string operator  +  (char c, const string& s2);
string operator  +  (const string& s1, const char* s);
string operator  +  (const string& s1, char c);
ostream& operator  <<  (ostream& os, const string& s);
istream& operator  >>  (istream& is, string& s);
string& operator  =  (const string& s);
string& operator  =  (const char* s);
string& operator  =  (char ch);
char& operator  []  (size_type index);
```

Members

a) string& **append**(const string& str);
a) string& **append**(const char* str);
b) string& **append**(const string& str, size_type i, size_type l);
c) string& **append**(const char* str, size_type num);
d) string& **append**(size_type e num, char ch);
e) string& **append**(input_iterator start, input_iterator end);
	a) Appends *str* to the end of the string.
	b) Appends a substring of *str* starting at position *i of size l*.
	c) Appends *num* characters of *str* to the string.
	d) Appends *num* repetitions of *ch* to the string.
	e) Appends the sequence from *start* to *end*, to the string

a) void **assign**(size_type num, const char& val);
b) void **assign**(input_iterator start, input_iterator end);
c) string& **assign**(const string& str);
c) string& **assign**(const char* str);
d) string& **assign**(const char* str, size_type num);
e) string& **assign**(const string& str,size_type index, size_type len);
f) string& **assign**(size_type num, const char& ch);
	a) Assigns to the string *num* copies of *val*.
	b) Assigns to the string the sequence from *start* to *end*.
	c) Assigns *str* to the string.
	d) Assigns num copies of str to the string
	e) Assigns a substring of *str* of length *len*, starting at *index*.
	f) Assigns *num* copies of *ch* to the string.

TYPE& **at**(size_type loc);
const TYPE& **at**(size_type loc) const;
	Returns a reference to the character at index loc.

iterator **begin**();
const_iterator **begin**() const;
	Returns an iterator to the first element of the string.

const char* **c_str**();	Returns a pointer to a const C string terminated with \0.
size_type **capacity**() const;	Returns the number of allocated positions in the string.
void **clear**();	Removes all the characters in the string.

a) int **compare**(const string& str);
a) int **compare**(const char* str);
b) int **compare**(size_type index, size_type length, const string& str ;
c) int **compare**(size_type i1, size_type l1, const string& str, size_type i2, size_type l2);
d) int **compare**(size_type index, size_type l1, const char* str, size_type l2);
	this < str returns <0; this == str returns 0; this > str returns >0.
	a) Compares the current string to str.
	b) Compares a substring starting at *index* of size *length* with *str.*
	c) Compares a substring of the current string (from index *i1* with *l1* character) to a substring of *str*(from *i2* of size *l2*).
	d) Compares a substring of the current string (from index with *l1* character) to a substring of *str*(from index 0 of size *l2*).

size_type **copy**(char* str, size_type n, size_type i = 0);	Copies *n* chars starting at *i* into an array of chars.
const char ***data**();	Returns a pointer to the first character of the string.
bool **empty**() const;	Returns true if the string is empty.

iterator **end**();
const_iterator **end**() const;
	Returns an iterator to the position just after the last element of the string.

iterator **erase**(iterator loc);
iterator **erase**(iterator start, iterator end);
string& **erase**(size_type index = 0, size_type num = npos);
	Erases the char at index *loc*, returns an iterator to the next char.
	Erases the chars from *start* (including) to *end* (excluding).
	Erases *num* characters from the string starting at *index*.

size_type **find**(const string& str, size_type index);	Returns the first occurence of str in the string, starting at index.

size_type **find**(const char* str, size_type index);
size_type **find**(const char* str, size_type index, size_type length);
size_type **find**(char ch, size_type index);
	String::npos is returned if no match is found.
	If len is given, returns the occurence of the 1st len characters.
	Returns the index of the 1st occurence of ch, starting at index.

size_type **find_first_not_of**(const string& str, size_type index = 0);
size_type **find_first_not_of**(const char* str, size_type index = 0);
size_type **find_first_not_of**(const char* str, size_type index, size_type num);
size_type **find_first_not_of**(char ch, size_type index = 0);
	Returns the index of the 1st occurence of a character in the string not matching a character in *str*, beggining at *index*. Searches for the 1st occurence of a char that doesnt match the 1st *num* chars of *str*. Searches for the 1st char different than *ch*.

size_type **find_first_of**(const string &str, size_type index = 0);
size_type **find_first_of**(const char* str, size_type index = 0);
size_type **find_first_of**(const char* str, size_type index, size_type num);
size_type **find_first_of**(char ch, size_type index = 0);
	Returns the index of the 1st occurence of any character in *str* or string::npos if no result is found. It searches starting at position *index* and ending at *num* (if specified). Or searches for the occurence of the single character *ch*.

size_type **find_last_not_of**(const string& str, size_type index = npos);
size_type **find_last_not_of**(const char* str, size_type index = npos);
size_type **find_last_not_of**(const char* str, size_type index, size_type num);
size_type **find_last_not_of**(char ch, size_type index = npos);
	Returns the index of the last occurence of the absence of any character in *str* or *ch* in the current string. string::npos is returned if no result is found. It searches in reverse order starting at position *index* an endinding at *num* (if specified).

size_type **find_last_of**(const string& str, size_type index = npos);
size_type **find_last_of**(const char* str, size_type index = npos);
size_type **find_last_of**(const char* str, size_type index, size_type num);
size_type **find_last_of**(char ch, size_type index = npos);
	Returns the index of the last occurence of any character in *str* or character *ch* in the current string. string::npos is returned if no result is found. It searches in reverse order starting at position *index* an ending at *num* (if specified).

istream& **getline**(istream& is, string& s, char delimiter = '\n');	Reads a line from *is* and saves it in *s*. getline reads data until *delimiter* is reached. getline is not a member of string class.
a) iterator **insert**(iterator i, const char& ch); b) string& **insert**(size_type index, const string& str); b) string& **insert**(size_type index, const char* str); c) string& **insert**(size_type i1, const string& str, size_type i2, size_type n); d) string& **insert**(size_type index, const char* str, size_type n); e) string& **insert**(size_type index, size_type n, char ch); f) void **insert**(iterator i, size_type n, const char& ch); g) void **insert**(iterator i, iterator start, iterator end);	a) inserts *ch* before the position pointed by i. b) inserts *str* at position *index.* c) inserts at position *i1* a substring of *str* starting at *i2* of *n* characters long. d) inserts, at position *index*, *n* characters of *str.* e) inserts, at position *index*, *n* copies of *ch*. f) inserts n copies of ch before the character denoted by i. g) inserts, before position i, the characters from start to end.
size_type **length**() const;	Returns the number of elements in the string.
size_type **max_size**() const;	Returns the maximum number of elements a string can hold. This number isn't influenced by the string's size or the number of allocated positions.
void **push_back**(const TYPE& val);	Inserts *val* at the end of the string.

reverse_iterator **rbegin**();
const_reverse_iterator **rbegin**() const;
	Returns a reverse iterator to the end of the string.

reverse_iterator **rend**();
const_reverse_iterator **rend**() const;
	Returns a reverse iterator to the begining of the string.

a) string& **replace**(size_type i, size_type n, const string& str);
a) string& **replace**(size_type i, size_type n, const char* str);
b) string& **replace**(iterator i1, iterator i2, const string& str);
b) string& **replace**(iterator i1, iterator i2, const char* str);
c) string& **replace**(size_type i1, size_type n1, const string& s, size_type i2, size_type n2);
d) string& **replace**(size_type i, size_type n1, const char* str, size_type n2);
e) string& **replace**(iterator start, iterator end, const char* str, size_type n);
f) string& **replace**(size_type i, size_type n1, size_type n2, char ch);
g) string& **replace**(iterator start, iterator end, size_type num, char ch);
	a) Replaces the characters starting at index *i,* with *n* characters long, with the characters from str.
	b) Replaces the characters from i1 to i2 with the characters from str.
	c) Replaces characters from i1 of length n1 with a substring of s, starting at i2 of length n2.
	d) Replaces chars from i of length n1 by the 1st n2 chars of str.
	e) Replaces chars from start to end by the 1st n chars of str.
	f) Replaces chars from i of length n1 by n2 copies of ch.
	g) Replaces chars from start to end by n2 copies of ch.

void **reserve**(size_type size);	Sets the minimum capacity of the string.
void **resize**(size_type num, const TYPE& val=TYPE());	Alters the size of the string to *num*, and if *val* is specified the new elements will be set to *val.*

size_type **rfind**(const string& str, size_type index);
size_type **rfind**(const char* str, size_type index);
size_type **rfind**(const char* str, size_type index, size_type num);
size_type **rfind**(char ch, size_type index);
	Searches the string in reverse order for the first occurence of *str/ch,* starting the search at position *index* and continuing the search util the beggining of the string, or position *num*, is reached. string::npos is returned if no result is found.

size_type **size**() const;	Returns the number of elements in the string.
string **substr**(size_type index, size_type length = npos);	Returns a substring of the string starting at *index* of size *npos*.
void **swap**(container& from);	Substitutes the elements of the string with the elements of string *from*.