vector

vector

```
std::vector<std::string> words1 {"the", "frogurt", "is", "also", "cursed"};
// words2 == words1
std::vector<std::string> words2(words1.begin(), words1.end());
// words3 == words1
std::vector<std::string> words3(words1);
// words4 is {"Mo", "Mo", "Mo", "Mo", "Mo"}
std::vector<std::string> words4(5, "Mo");
std::vector<int> x { 1, 2, 3 }, y, z;
```

assign

```
void assign( size_type count, const T& value );
void assign( InputIt first, InputIt last );
void assign( std::initializer_list<T> ilist );
```

[] operator

- this operator never inserts a new element into the container

```
reference operator[]( size_type pos );
reference at( size_type pos );
reference front();
reference back();
```

data

```
T* data();
```

iterator

```
iterator begin();
iterator end();
reverse_iterator rbegin();
reverse_iterator rend();
```

Capacity

```
bool empty() const;
size_type size() const;
size_type max_size() const;
void reserve( size_type new_cap );
size_type capacity() const;
void shrink_to_fit();
```

Modifiers

```
void clear();

// insert
iterator insert( iterator pos, const T& value );
void insert( iterator pos, size_type count, const T& value );
void insert( iterator pos, InputIt first, InputIt last );


iterator erase( iterator pos );
iterator erase( iterator first, iterator last );

void push_back( const T& value );
void push_back( T&& value );

void pop_back();

void resize( size_type count );
void resize( size_type count, T value = T() );
void resize( size_type count, const value_type& value );

void swap( vector& other );
```

emplace

```
// emplace
iterator emplace( const_iterator pos, Args&&... args );
std::vector<A> container;
A two { "two" };
A three { "three" };
std::cout << "emplace with A&:\n";
container.emplace(container.end(), two);

std::cout << "emplace with A&&:\n";
container.emplace(container.end(), std::move(three));

void emplace_back( Args&&... args );
reference emplace_back( Args&&... args );
```

string

operator =

```
basic_string& operator=( const basic_string& str );
basic_string& operator=( const CharT* s );
```

assign

```
basic_string& assign( size_type count, CharT ch );
basic_string& assign( const basic_string& str );
basic_string& assign( const basic_string& str, size_type pos, size_type count );
basic_string& assign( const basic_string& str, size_type pos, size_type count = npos);
basic_string& assign( basic_string&& str );
basic_string& assign( const CharT* s, size_type count );
basic_string& assign( const CharT* s );
basic_string& assign( InputIt first, InputIt last );
basic_string& assign( const StringViewLike& t );
basic_string& assign( const StringViewLike& t, size_type pos, size_type count = npos);
```

access

```
reference       at( size_type pos );
reference operator[]( size_type pos );
CharT& back();
const CharT* data() const;
const CharT* c_str() const;
```

iterator

```
iterator begin();
iterator end();
reverse_iterator rbegin();
reverse_iterator rend();
```

Capacity

```
bool empty() const;
size_type size() const;
size_type length() const;
size_type max_size() const;
void reserve( size_type new_cap = 0 );
size_type capacity() const;
void shrink_to_fit();
```

operations

```
void clear();

basic_string& erase( size_type index = 0, size_type count = npos );
iterator erase( iterator position );
iterator erase( iterator first, iterator last );

void push_back( CharT ch );
void pop_back();
basic_string substr( size_type pos = 0, size_type count = npos ) const;
size_type copy( CharT* dest, size_type count, size_type pos = 0 ) const;
void resize( size_type count );
void resize( size_type count, CharT ch );
void swap( basic_string& other );
```

insert

```
basic_string& insert( size_type index, size_type count, CharT ch );
basic_string& insert( size_type index, const CharT* s );
basic_string& insert( size_type index, const CharT* s, size_type count );
basic_string& insert( size_type index, const basic_string& str );
basic_string& insert( size_type index, const basic_string& str, size_type index_str,
size_type count = npos );
iterator insert( iterator pos, CharT ch );
void insert( iterator pos, size_type count, CharT ch );
void insert( iterator pos, InputIt first, InputIt last );
iterator insert( const_iterator pos, InputIt first, InputIt last );
iterator insert( const_iterator pos, std::initializer_list<CharT> ilist );
```

append

```
basic_string& append( size_type count, CharT ch );
basic_string& append( const basic_string& str );
basic_string& append( const basic_string& str, size_type pos, size_type count );
basic_string& append( const basic_string& str, size_type pos, size_type count = npos );
basic_string& append( const CharT* s, size_type count );
basic_string& append( const CharT* s );
basic_string& append( InputIt first, InputIt last );
```

+= operator

```
basic_string& operator+=( const basic_string& str );
basic_string& operator+=( CharT ch );
basic_string& operator+=( const CharT* s );
basic_string& operator+=( std::initializer_list<CharT> ilist );
basic_string& operator+=( const StringViewLike& t );
```

compare

```
int compare( const basic_string& str ) const;
int compare( size_type pos1, size_type count1, const basic_string& str ) const;
int compare( size_type pos1, size_type count1, const basic_string& str, size_type pos2,
size_type count2 ) const;
int compare( const CharT* s ) const;
int compare( size_type pos1, size_type count1, const CharT* s ) const;
int compare( size_type pos1, size_type count1, const CharT* s, size_type count2 ) const;
```

replace

```
basic_string& replace( size_type pos, size_type count, const basic_string& str );
basic_string& replace( const_iterator first, const_iterator last, const basic_string& str
);
basic_string& replace( size_type pos, size_type count, const basic_string& str, size_type
pos2, size_type count2 );
basic_string& replace( size_type pos, size_type count, const basic_string& str, size_type
pos2, size_type count2 = npos );
basic_string& replace( const_iterator first, const_iterator last, InputIt first2, InputIt
last2 );
basic_string& replace( size_type pos, size_type count, const CharT* cstr, size_type count2
);
basic_string& replace( const_iterator first, const_iterator last, const CharT* cstr,
size_type count2 );
basic_string& replace( size_type pos, size_type count, const CharT* cstr );
```

search

```
size_type find( const basic_string& str, size_type pos = 0 ) const;
size_type find( const CharT* s, size_type pos, size_type count ) const;
size_type find( const CharT* s, size_type pos = 0 ) const;
size_type find( CharT ch, size_type pos = 0 ) const;

size_type rfind( const basic_string& str, size_type pos = npos ) const;
size_type rfind( const CharT* s, size_type pos, size_type count ) const;
size_type rfind( const CharT* s, size_type pos = npos ) const;
size_type rfind( CharT ch, size_type pos = npos ) const;

size_type find_first_of( const basic_string& str, size_type pos = 0 ) const;
size_type find_first_of( const basic_string& str, size_type pos = 0 ) const noexcept;
size_type find_first_of( const CharT* s, size_type pos, size_type count ) const;
size_type find_first_of( const CharT* s, size_type pos = 0 ) const;
size_type find_first_of( CharT ch, size_type pos = 0 ) const;

size_type find_first_not_of( const basic_string& str, size_type pos = 0 ) const;
size_type find_first_not_of( const CharT* s, size_type pos, size_type count ) const;
size_type find_first_not_of( const CharT* s, size_type pos = 0 ) const;
```

```
size_type find_last_of( const basic_string& str, size_type pos = npos ) const;
size_type find_last_of( const CharT* s, size_type pos, size_type count ) const;
size_type find_last_of( const CharT* s, size_type pos = npos ) const;
size_type find_last_of( CharT ch, size_type pos = npos ) const;

size_type find_last_not_of( const basic_string& str,
                            size_type pos = npos ) const;
size_type find_last_not_of( const basic_string& str,
                            size_type pos = npos ) const noexcept;
size_type find_last_not_of( const CharT* s, size_type pos = npos ) const;
size_type find_last_not_of( CharT ch, size_type pos = npos ) const;

static const size_type npos = -1;
```

numeric conversion

```
int       stoi( const std::string& str, std::size_t* pos = nullptr, int base = 10 );
long      stol( const std::string& str, std::size_t* pos = nullptr, int base = 10 );
long long stoll( const std::string& str, std::size_t* pos = nullptr, int base = 10 );
float       stof( const std::string& str, std::size_t* pos = nullptr );
double      stod( const std::string& str, std::size_t* pos = nullptr );
long double stold( const std::string& str, std::size_t* pos = nullptr );
```

string conversion

```
std::string to_string( int value );
std::string to_string( unsigned value );
std::string to_string( float value );
std::string to_string( double value );
```