Standard Library and Standard Template Library (STL)

Standard Library Organizations

- namespace ***std***
- organized by ***headers***
- c libraries
  - header <cX> = header <X.h>

Containers

- Contains multiple items

<vector>	*one-dimensional array of T*
<list>	*doubly-linked list of T*
<deque>	*double-ended queue of T*
<queue>	*queue of T*
<stack>	*stack of T*
<map>	*associative array of T*
<set>	*set of T*
<bitset>	*array of booleans*

Algorithms

<algorithm>	*general algorithms*
<cstdlib>	***bsearch( ) qsort( )***

Iterators

<iterator>	*iterators and iterator support*

General Utilities

<utility>	*operators and pairs*
<functional>	*function objects*
<memory>	*allocators for containers*
<ctime>	*C-style date and time*

Diagnostics

<exception>	*exception class*
<stdexcept>	*standard exceptions*
<cassert>	*assert macro*
<cerrno>	*C-style error handling*

Strings

<string>	*string of T*
<cctype>	*character classification*
<cwtype>	*wide-character classification*
<cstring>	*C-style string functions*
<cwchar>	*C-style wide-character string functions*
<cstdlib>	*C-style string functions*

- <cstring>: strlen(), strcpy()
- <cstdlib>: atoi(), atof()

Input/Output

<iosfwd>	*forward declarations of I/O facilities*
<iostream>	*standard iostream objects and operations*
<ios>	*iostream bases*
<streambuf>	*stream buffers*
<istream>	*input stream template*
<ostream>	*output stream template*
<iomanip>	*manipulators*
<sstream>	*streams to/from strings*
<cstdlib>	*character classification functions*
<fstream>	*streams to/from files*
<cstdio>	***printf*** *( ) family of I/O*
<cwchar>	***printf*** *( ) -style I/O of wide characters*

Language Support

<limits>	numeric limits
<climits>	C-style numeric scalar-limit macros
<cfloat>	C-style numeric floating-point limit macros
<new>	dynamic memory management
<typeinfo>	run-time type identification support
<exception>	exception-handling support
<cstddef>	C library language support
<cstdarg>	variable-length function argument lists
<csetjmp>	C-style stack unwinding
<cstdlib>	program termination
<ctime>	system clock
<csignal>	C-style signal handling

Numerics

<complex>	*complex numbers and operations*
<valarray>	*numeric vectors and operations*
<numeric>	*generalized numeric operations*
<cmath>	*standard mathematical functions*
<cstdlib>	*C-style random numbers*

Container

- The most useful component of standard library, called Standard Template Library (STL)
- Container: an object that holds other objects
- Functionalities
  - Constructors
  - Set properties
  - Insert / remove / replace
  - Iterate / random access
  - Algorithms

Vector

- Most often used
- Array like, sequential container
- Implemented by array
- Cheap: append (*push_back()*),
  random access([])
- Expensive: insert in the middle, expansion

Vector

- *vector<T> v;*
- *v.push_back(T v)*: append
- *v[i], v.at(i)*

Iterating a sequential container

- Vector, List, Deque
- Using index
  - *for(into i; i < v.size(); i++) { … v[i] … }*
  - only works for vector/deque
- Using iterator
  - *for(vector<T>::iterator p=v.begin();*
    *p != v.end(); p++) { … *p … }*
- Using auto (C++11)
  - *for(auto p=v.begin(); p!=v.end(); p++) {… *p …}*
- Using abbreviation (C++11)
  - *for(auto p : v) { … *p … }*
- Using while

Generic Programming by Macro

- **Very powerful** if used correctly
  - for(auto val : v) { <operations> }
  - FORALL_BEGIN(v) <operations> FORALL_END
  - FORALL(v, <operations>)
  - MAP(v, <initialize>, <ops>, <finalize>)

- Why not by template function?
  - Very difficult to pass the operations
  - Function pointer? hard
  - Operation Class? hard to define generic operation

List

- Sequential container
- Implemented by doubly-linked list
- Cheap: append (*push_back()*)
  prepend (*push_front()*)
  insert in the middle
  expansion
- Expensive: random access (impossible)
- Only use iterator

Deque

- Array like, sequential container
- Implemented by dynamic array
- Cheap: append (*push_back()*)
  random access([])
  prepend (*push_front()*)
  expansion
- Expensive: insert in the middle