



EECS 204002

Data Structures 資料結構

Prof. REN-SONG TSAY 蔡仁松 教授

NTHU

C++ STL (STANDARD TEMPLATE LIBRARY)

<https://www.tutorialspoint.com/cplusplus/index.htm>



1.6

The Standard Template Library

Standard C++ Libraries

- The C++ Standard Library: a rich set of functions manipulating **files, strings**, etc.
- The Standard Template Library (STL): provide general-purpose classes and functions with templates that implement many commonly used algorithms and data structures like vectors, lists, queues, and stacks.

The STL

- An ISO C++ standard framework of about 10 **containers** and about 60 **algorithms** connected by **iterators**
 - Other organizations provide more containers and algorithms in the style of the STL
 - Boost.org, Microsoft, SGI, ...
- Probably the best known and most widely used example of generic programming

Four STL Components

- **Containers:** used to manage collections of objects of a certain kind.
- **Algorithms:** act on containers.
 - E.g. initialization, sorting, searching....
- **Iterators:** used to step through the elements of collections of objects.
- **Functors** are objects that can be treated as though they are a function or function pointer.

Containers

- A container is an object to store data, either built-in data types like int and float, or class objects
- The STL provides several basic types of containers
 - `<vector>` : one-dimensional array
 - `<list>` : double linked list
 - `<deque>` : double-ended queue
 - `<queue>` : queue
 - `<stack>` : stack
 - `<set>` : set
 - `<map>` : associative array

Containers

(hold sequences in different ways)

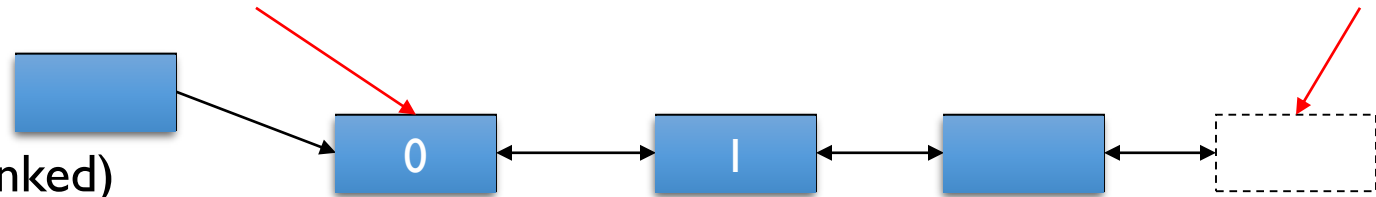
- **vector**

(array)



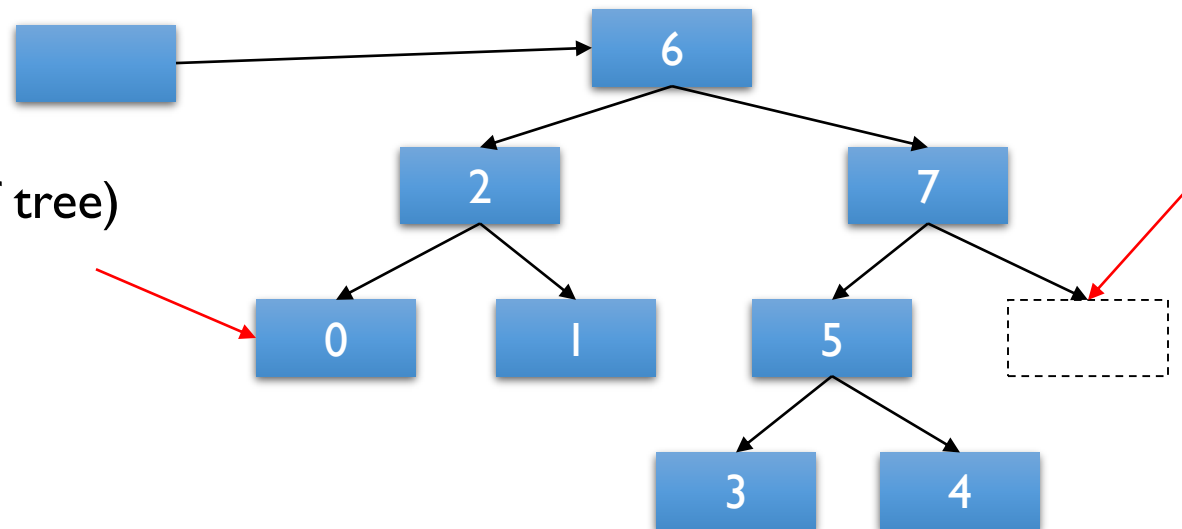
- **list**

(doubly linked)



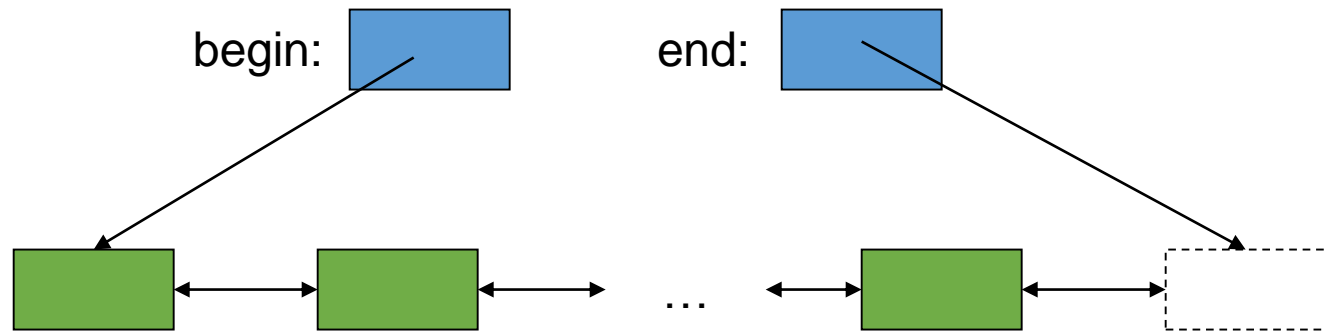
- **set**

(a type of tree)



Basic Iterator Model

- A pair of iterators define a sequence
 - The beginning (points to the first element – if any)
 - The end (points to the one-beyond-the-last element)



- An iterator is a type that supports the “iterator operations”
 - ++ Go to next element
 - * Get value
 - == Does this iterator point to the same element?
- Some iterators support more operations (e.g. --, +, and [])

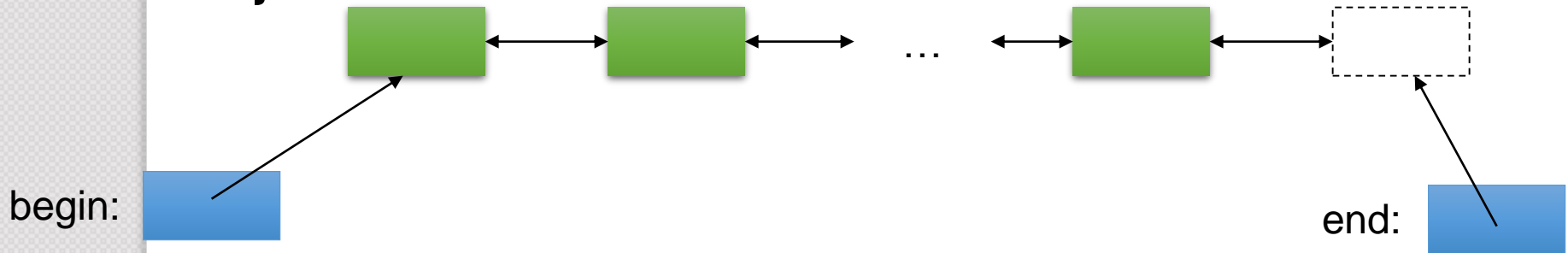
Algorithms

- Algorithms such as searching and sorting are provided in the STL.
- Require user to specify comparison operator $<$ or custom comparator function.

The simplest algorithm: find()

// Find the first element that equals a value

```
template<class In, class T>  
In find(In first, In last, const T& val)  
{  
    while (first!=last && *first != val) ++first;  
    return first;  
}
```



We can ignore (“abstract away”) the differences between containers

<algorithm> sort

<http://www.cplusplus.com/reference/algorithm/sort/>

```
// sort algorithm example
#include <iostream>    // std::cout
#include <algorithm>    // std::sort
#include <vector>      // std::vector

bool myfunction (int i, int j) { return (i<j); }

struct myclass {
    bool operator() (int i,int j) { return (i<j); }
} myobject;

int main () {
    int myints[] = {32,71,12,45,26,80,53,33};
    std::vector<int> myvector (myints, myints+8);           // 32 71 12 45 26 80 53 33

    // using default comparison (operator <):
    std::sort (myvector.begin(), myvector.begin()+4);       //(12 32 45 71)26 80 53 33

    // using function as comp
    std::sort (myvector.begin()+4, myvector.end(), myfunction); // 12 32 45 71(26 33 53 80)

    // using object as comp
    std::sort (myvector.begin(), myvector.end(), myobject); //(12 26 32 33 45 53 71 80)

    // print out content:
    std::cout << "myvector contains:";
    for (std::vector<int>::iterator it=myvector.begin(); it!=myvector.end(); ++it)
        std::cout << ' ' << *it;
    std::cout << '\n';

    return 0;
}
```

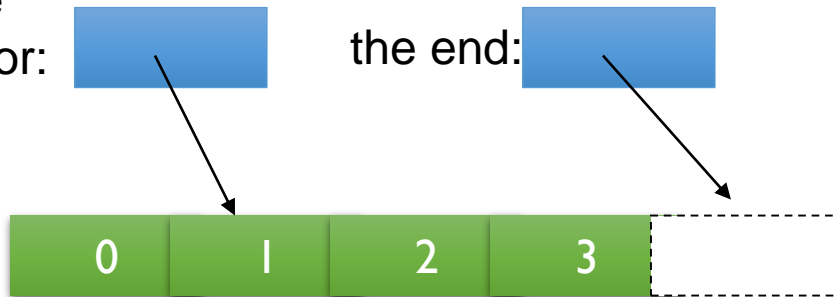
Output:

myvector contains: 12 26 32 33 45 53 71 80

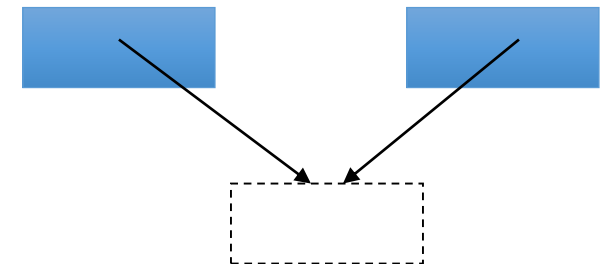
Algorithms and iterators

- The end of the sequence is “one past the last element”
 - **not** “the last element”
 - That’s necessary to elegantly represent an **empty** sequence
 - One-past-the-last-element isn’t an element
 - You can compare an iterator pointing to it
 - You can’t dereference it (read its value)
- Returning the end of the sequence is the standard idiom for “not found” or “unsuccessful”

some
iterator:



An empty sequence:



Functor

- A Functor can be a general function
- Functor can also be a class object, which overloads `operator()`
- The STL algorithm takes user's Functor to perform customized actions.
- In the `std::sort`, Functor *Comp* lets user define customized comparison function.
- One may define a complex comparison function
 - E.g. triple comparison : $(1,1,3) ? (1,2,1)$

Container adaptors

Container adaptors

Container adaptors provide a different interface for sequential containers.

stack	adapts a container to provide stack (LIFO data structure) (class template)
queue	adapts a container to provide queue (FIFO data structure) (class template)
priority_queue	adapts a container to provide priority queue (class template)

Simple Sample Code :

<vector> (Like Dynamic Array)

```
// vector example
#include <iostream>
#include <vector>

int main ()
{
//initialize
std::vector<int> myVector;

//-----
// ADD ELEMENTS
int myData=9;

// add ten integer
for( int i=0 ; i < 10 ; i++) {
    myData++;
    myVector.push_back(myData);
}
```

```
// TRAVEL ELEMENTS
// get the number of elements
int vSize = myVector.size();

for( int i=0 ; i < vSize ; i++){
    std::cout<< myVector[i] << '\n';
}
//-----
// DELETE SPECIFIED ELEMENT
// get the iterator of beginning elements
auto iter = myVector.begin();
// remove 2nd element in this vector
myVector.erase(iter+1);

myVector[0] = 5 ;
// print out
for( int i=0 ; i < myVector.size() ; i++){
    std::cout<< myVector[i] << '\n';
}
```

References

- **CPP Reference**
 - <http://www.cplusplus.com/reference/stl/>
 - <http://en.cppreference.com/w/>
- **M\$ Doc**
 - [http://msdn.microsoft.com/en-us/library/c191tb28\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/c191tb28(v=vs.100).aspx)
- https://en.wikipedia.org/wiki/Standard_Template_Library

References

- Introduction to Algorithms, 3rd ed., by Cormen et al.
- C++ Primer, 5th Edition, by Stanley B. Lippman et al.
- The C++ Programming Language, by Bjarne Stroustrup
- Inside the C++ Object Model , by Stanley B. Lippman
- C++ Templates: The Complete Guide, by David Vandevoorde et al.
- C++ Coding Standards: 101 Rules, Guidelines, and Best Practices, by Herb Sutter

References

- *C++ Primer 5th*
 - [http://books.google.com.tw/books?hl=zh-TW&id=JlHMLyxqJfgC&q=operator+overaling#v=onepage&q=chapter%2014&f=false](http://books.google.com.tw/books?hl=zh-TW&id=JlHMLyxqJfgC&q=operator+overloading#v=onepage&q=chapter%2014&f=false)
- MIT's Introduction to C++
 - <http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-096-introduction-to-c-january-iap-2011/lecture-notes/>
- MSDN C++ Reference:
 - [http://msdn.microsoft.com/en-us/library/3bstk3k5\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/3bstk3k5(v=vs.100).aspx)
- NTU OCW:
 - <http://ocw.aca.ntu.edu.tw/ntu-ocw/index.php/ocw/cou/101SI12>

