



2007-02-20

Lecture #7

(Material on Test 1)

Today's Outline

C++ STL

C++ Templates

C++ STL

<http://www.sgi.com/tech/stl/>

The Standard Template Library, or STL, is a C++ library of

- **container classes**
- **algorithms**
- **iterators**

It provides many of the basic algorithms and data structures of computer science. The STL is a generic library, meaning that its components are heavily parameterized: almost every component in the STL is a template. You should make sure that you understand how templates work in C++ before you use the STL.

C++ Templates

```
template <class T>  
class Stack  
{  
public:  
    Stack(int = 10) ;  
    ~Stack() { delete [] stackPtr ; }  
    int push(const T&);  
    int pop(T&) ; // pop an element off the stack  
    int isEmpty()const { return top == -1 ; }  
    int isFull() const { return top == size - 1 ; }  
private:  
    int size ; // Number of elements on Stack  
    int top ;  
    T* stackPtr ;  
};
```

C++ Templates

```
// push an element onto the Stack  
template <class T>  
int Stack<T>::push(const T& item)  
{  
    if (!isFull())  
    {  
        stackPtr[++top] = item ;  
        return 1 ; // push successful  
    }  
    return 0 ; // push unsuccessful  
}
```

C++ Templates

```
// pop an element off the Stack  
template <class T>  
int Stack<T>::pop(T& popValue)  
{  
    if (!isEmpty())  
    {  
        popValue = stackPtr[top--] ;  
        return 1 ; // pop successful  
    }  
    return 0 ; // pop unsuccessful  
}
```

C++ Templates

```
#include <iostream>
#include "stack.h"
using namespace std ;
int main(int argc, char * argv[])
{
    Stack<float> fs(5) ;

    float f = 1.1 ;
    while (fs.push(f)) {
        cout << f << ' ' ;
        f += 1.1 ;
    }
    cout << endl << "Stack Full." << endl ;
    while (fs.pop(f))
        cout << f << ' ' ;
}
```

C++ Templates

```
#include <iostream>
#include "stack.h"
using namespace std ;
int main(int argc, char * argv[])
{
    Stack<int> is(5) ;

    int i = 1 ;
    while (is.push(i)) {
        cout << i << ' ' ;
        i += 1 ;
    }
    cout << endl << "Stack Full." << endl ;
    while (is.pop(i))
        cout << i << ' ' ;
}
```

C++ STL

1.)Containers

- hold the data

2.)Iterators

- allow you to traverse (walk-through) the data

3.)Algorithms

- manipulate the data (Merge, Union, Difference, Sort, Search)

C++ STL

Sequences

- **vector** (similar to an array)
- **deque** (similar to a vector)
- **list** (doubly linked list)
- **slist** (singly linked list)
- **bit_vector** (same as `vector<bool>`)

A Sequence is a variable-sized Container whose elements are arranged in a strict linear order. It supports insertion and removal of elements.

Why so many choices? Implementation determines efficiency

C++ STL

Associative Containers

- **set**
- **map**
- **multiset**
- **multimap**
- **hash_set**
- **hash_map**
- **hash_multiset**
- **hash_multimap**
- **hash**

An Associative Container is a variable-sized Container that supports efficient retrieval of elements (values) based on keys. It supports insertion and removal of elements, but differs from a Sequence in that it does not provide a mechanism for inserting an element at a specific position.

C++ STL

```
#include <string>
```

The `basic_string` class represents a Sequence of characters. It contains all the usual operations of a Sequence, and, additionally, it contains standard string operations such as search and concatenation.

C++ STL

Container adaptors

- **stack**
- **queue**
- **priority_queue**

provides a restricted subset of Container functionality

C++ STL

bitset

Bitset is very similar to `vector<bool>` (also known as `bit_vector`): it contains a collection of bits, and provides constant-time access to each bit.

two main differences between `bitset` and `vector<bool>`

- 1.)the size of a `bitset` cannot be changed**
- 2.)`bitset` is not a Sequence; in fact, it is not an STL Container at all. It does not have iterators, for example, or `begin()` and `end()` member functions. Instead, `bitset`'s interface resembles that of unsigned integers.**