CSE 326: Data Structures
Lecture #2
June 21, 2000

C++ Templates
(concluded)
Asymptotic Analysis
(beginning)


```
#include <iostream.h>
#include "vector.h"     // vector (our version, in Appendix B)
#include "mystring.h"   // string (our version, in Appendix B)

int main( )
{
   vector<string> v( 5 );
   int itemsRead = 0;
   string x;

   while( cin >> x )
   {
      if( itemsRead == v.size( ) )
         v.resize( v.size( ) * 2 );
      v[ itemsRead++ ] = x;
   }

   for( int i = itemsRead - 1; i >= 0; i-- )
      cout << v[ i ] << endl;
   return 0;
}
```


Template Function Example

```
template <class Comparable>
const Comparable & findMax ( const vector<Comparable>
   &a ) {
   int maxIndex = 0;
   for (int i = 1; i < a.size( ); i++)
        if (a [ maxIndex] < a[i])
              maxIndex = 1;
   return a [ maxIndex ];
}
```


Template Class Example

```
template <class Object>
   class MemoryCell
   {
    public:
      explicit MemoryCell( const Object & initialValue = Object( ) );
      const Object & read( ) const;
      void write( const Object & x );
    private:
      Object storedValue;
   };
```


(Separate) Method Implementation

```
template <class Object>
   MemoryCell<Object>::MemoryCell( const
 Object & initialValue )
     : storedValue( initialValue )
   {
   }
```


Analysis of Algorithms

- Analysis of an algorithm gives insight into how long the program runs and how much memory it uses
  - time complexity
  - space complexity
- Analysis can provide insight into alternative algorithms
- Input size is indicated by a number *n* (sometimes there are multiple inputs)
- Running time is a function of *n* such as
  $T(n) = 4n + 5$
  $T(n) = 0.5\ n \log n - 2n + 7$
  $T(n) = 2^n + n^3 + 3n$
- But...

Asymptotic Analysis

- Eliminate low order terms
  - $4n + 5 \Rightarrow 4n$
  - $0.5\ n \log n - 2n + 7 \Rightarrow 0.5\ n \log n$
  - $2^n + n^3 + 3n \Rightarrow 2^n$
- Eliminate coefficients
  - $4n \Rightarrow n$
  - $0.5\ n \log n \Rightarrow n \log n$
  - $n \log n^2 = 2\ n \log n \Rightarrow n \log n$

Rates of Growth

- Suppose a computer executes 10^{12} ops per second:

	10	100	1,000	10,000
n	10^{-11}s	10^{-10}s	10^{-9}s	10^{-8}s
n log n	10^{-11}s	10^{-9}s	10^{-8}s	10^{-7}s
n^2	10^{-10}s	10^{-8}s	10^{-6}s	10^{-4}s
n^3	10^{-9}s	10^{-6}s	10^{-3}s	1s
2^n	10^{-9}s	10^{18}s	10^{289}s	

10^4s = 2.8 hrs 10^{18}s = 30 billion years

Order Notation Definitions

- $T(n) \in O(f(n))$ if there are constants c and n_0 such that $T(n) \leq c\ f(n)$ for all $n \geq n_0$
- $T(n) \in \Omega(f(n))$ if there are constants c and n_0 such that $T(n) \geq c\ f(n)$ for all $n \geq n_0$
- $T(n) \in \theta(f(n))$ if $T(n) \in O(f(n))$ and $T(n) \in \Omega(f(n))$
- $T(n) \in o(f(n))$ if $T(n) \in O(f(n))$ and $T(n) \notin \theta(f(n))$

Examples

$10{,}000\ n^2 + 25\ n \in \theta(n^2)$

$10^{-10}\ n^2 \in \theta(n^2)$

$n \log n \in O(n^2)$

$n \log n \in \Omega(n)$

$n^3 + 4 \in o(n^4)$

$n^3 + 4 \in \omega(n^2)$