Course C^{++}, Exercise List 6

28.03.2013

Topic of the exercise are `std::list< >` and `std::vector< >`.

1. Write a function

```
      #include <fstream>
      #include <vector>
      #include <string>

      std::vector< std::string> readfile( const std::string& name )
      {
         std::ifstream input( name. c_str( ));

         while( input. good( ) && !input. eof( ))
         {
            int c = input. get( );
         }

      }
```

   The function should read the complete text file **name**, and return its contents as a vector of strings. The strings are the words occurring in **name**. We assume that every whitespace (tab, space or return) is the start of new word. (Whitespace of more than one character should not result in empty words.)

   Find a text file that is not too small to test it on.

   Use

```
std::ostream& operator << ( std::ostream& stream,
                            const std::vector< std::string > & vect )
{
   for( std::vector< std::string > :: const_iterator
           p = vect. begin( );
           p != vect. end( );
           ++ p )
   {
      stream << *p << "\n";
   }
   return stream;
}
```

2. Consider the sorting functions:

   ```
   void sort( std::vector< std::string > & v )
   {
      for( unsigned int j = 0; j < v. size( ); ++ j )
         for( unsigned int i = 0; i < j; ++ i )
         {
            if( v[i] > v[j] )
            {
               std::string s = v[i];
               v[i] = v[j];
               v[j] = s;
            }
         }
   }
   ```

   and

   ```
   void sort( std::vector< std::string > & v )
   {
      for( unsigned int j = 0; j < v. size( ); ++ j )
         for( unsigned int i = 0; i < j; ++ i )
         {
            if( v[i] > v[j] )
            {
               std::string s = std::move( v[i] );
               v[i] = std::move( v[j] );
               v[j] = std::move( s );
            }
         }
   }
   ```

   Try to measure a difference in performance. Is there some? (Use input from a big file.) What is the cause of this difference?

3. Instead of indexing, it is much nicer in general to use iterators. `std::vector` has two iterator types, `std::vector<X> :: iterator` and `std::vector<X> :: const_iterator`. Since `v` is const, we can only use `const_iterator`.

You can see in `operator <<` above how iterators are used. Instead of the iterator declaration, one can use `auto` in C^{++}-11. (So one can write `for( auto p = v. begin( ); p != v. end( ); ++ p ))`

Rewrite the sorting function with iterators.

4. Rewrite the previous functions (reading a file, sorting a list of strings, printing a list of strings) using `std::list< >`. The difference shouldn't be big. When using list iterator, you have to be aware of the fact that `<` doesn't exist for list iterators. Replace it by `!=`.

   `std::list< >` is based on a doubly linked list. Linked lists use more memory, have no random access, but have the advantage that elements can be inserted or deleted at arbitrary positions.

5. Write a function

   ```
   void removeshortstrings( std::list< std::string > & lst,
                            unsigned int len )
   ```

   that removes all strings that are shorter than `len` from `lst`.

   Don't copy the list, use `erase( iterator it )`.