Programming in C^{++}, Exercise List 9

Deadline: 05.05.2015

In this exercise, we will try out other ways of providing orders and hash functions to containers, and we compare the efficiency of hash map versus search trees.

1. In Task 6, we made a function **readfile** that reads a vector of words. In Task 8, we made two functions that construct frequency tables, one using `map`, and another one using `unordered_map`.

   Using the time measurement methods of Task 6, it should be easy to compare the two frequency counting functions on the file **1.html**. Make a table for input files of different lengths, (or you can use multiple copies of **1.html** as in Task 6/5.)

2. Rewrite the functions **frequencytable** using lambdas. There will be two versions, one for `map` and one for `unordered_map`.

   Unfortunately, there is no way around making the functions polymorphic, with the following signature (for `std::map`):

   ```
   template< typename D >
   std::map< std::string, unsigned int, D >
   frequencytable( const std::vector<std::string > & text, const D& cmp )
   ```

   If you didn't do this already in Task 8, you will now also have to make the printing functions `operator <<` polymorphic. See Task 8/4.

3. Rewrite the functions **frequencytable** one more time, this time using function pointers.

   If everything went well, the printing functions from the previous task will work.

4. It would be nice to know which word is most frequent. Write a function

   ```
   template< typename A, typename C >
   std::pair< A, unsigned int > most_frequent(
      const std::map< A, unsigned int, C > & mp )
   ```

that returns a pair consisting of the most frequent object, and its frequency in the map. Your function may crash shamelessly when the map is empty.

If you try to print the result, you will see an error message that goes over several pages, because there exists no `operator <<` for pairs.

Either define the operator, or store the result in a local variable (use **auto**), and print the components of the pair separately.