CS900 Lab - Day 12
July 15th, 2003
Instructor: Thomas Hofmann

New Topics Covered
Manipulating and couting strings, Lecture: Day 12

Installing
Copy the file word.cpp from the course directory

```
cp /course/cs900/day12/src/word.cpp .
```

Compiling
You should use the gnu C++ compiler g++. Your program will be in cr.cpp . To compile type

```
g++ word.cpp -o word
```

Problem 1: Streams

Familiarize yourself with the C++ stream class. For what follows you need to include the headers for string, iostream, fstream and map (which is already prepared in the file word.cpp).

1. (a)

Reimplement the example discussed in class by defining the following function

```
int countTokens(string filename);
```

which opens the file a specified by the path/file name and counts the number of tokens. You can use the text files provided in the course directory

```
/course/cs900/day12/data/ulyss.txt, ./bible.txt, ./kant.txt
```

1. (b)

Write another function that counts the number of characters in the file

```
int countCharacters(string filename);
```

1. (c)

Write a functions that count the number of occurrences of a particular token:

```
int count(string token, string filename);
```

Test your function by testing how many times the word “happy” occurs in the different texts.

1. (d)

Write a function that counts how many digits (0-9) occur in a file.

```
int countDigits(string filename);
```

1. (e)

Filter a file by keeping only tokens with exactly n characters and writing those to an output file. The prototype of the function could be

```
void filter(string infile, string outfile, int length);
```

Here the arguments infile/outfile refer to the filenames of the input and output files, respectively, while length denotes the desired length of the tokens.

Problem 2: Maps

2.(a)

Implement the WordMap class discussed in class. In particular implement methods

```
void operator+=(const Word& w);
```

Which should increment the word count of the specified word by one. Also implement a print method to print out all the words with their counts.

```
void print() const;
```

Hint: You have to use the following iterator:

```
WordMap wm;
// …
WordMap::const_iterator i;
for (i=wm.begin(); i!=wm.end(); i++) {
  //-- do something
}
```

2.(b)

In order to test whether a certain word is stored in a map, you can use the count method defined for STL maps. Implement the following method for the class WordMap:

```
bool contains(const Word& w) const;
```

2.(c)

Implement a method that computes a total word count for all the words in the map (the sum of all frequencies):

```
int total() const;
```

2.(d)

Write a constructor that creates a WordMap from a file by tokenizing the file, and using the Word methods clean and lowercase to preprocess every word.

```
WordMap(const string& fname) ;
```

Problem 3: Word Statistics

3.(a)

Using a WordMap generated from one of the text files, write a method that sorts all the words contained in the map according to their frequency and prints out the top N words in order.

```
void printTop(int N=30) const ;
```

Hint: You may want to create an array of words and an array of frequencies using the new operator:

```
Word *words  = new  Word[size()];
int  *counts = new  int[size()];
//-- copy map into array
//-- sort array e.g. using bubblesort
delete [] words;
delete [] counts;
```

3.(b)

Write a program that generates (normalized) words at random according to their frequency in the processed text. For example, if the word “many” occurs 4000 times in a text with 1 million tokens, then the probability of generating “many” should be 0.004.

Hint: Generate a random number between 0 and the total number of tokens -1 (computed with the total method). Then iterate through the map and select the right word (you need a good idea!).