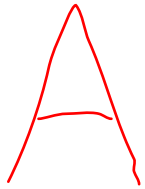
```
//William Saunders Bliss (wsb2) and John Felkins (jrf9)
//CPS 108
//Program reads a file and prints the 20 most frequently used words.  Program de
rived from sortdemo.cpp
//by Professor Astrachan.

#include <iostream>
#include <fstream>
#include <map>
#include <vector>
#include <algorithm>   // for exit
#include <iomanip>
#include <set>
#include <string>
#include <sstream>
#include <cstdlib>     // for exit
using namespace std;

#include "getopt.h"
#include "printer.h"

void ToLower(string & s)
    // postcondition: s all lower case
    {
        int len = s.length();
        for(int k=0; k < len; k++)
            {
                s[k] = tolower(s[k]);
            }
    }

void StripPunc(string & word)
     {
         int first = 0;
         int len = word.length();
         while(first < len && ispunct(word[first]))
             {
                 first++;
             }
         int last = len - 1;
         while(last >=0 && ispunct(word[last]))
             {
                 last--;
             }
         word = word.substr(first, last-first+1);
     }

ostream& operator <<(ostream& out, const pair<string,int>& p)
// post: string and int printed, int first
 {
     out << setw(6) << p.second << " " << p.first;
    return out;
 }

class WordTracker
{
  public:
    WordTracker();

    virtual void read();
    virtual void readFile(const string& filename);
    virtual void top(int n, vector<pair<string,int> >& v);

  protected:

    map<string,int> myMap;
};

WordTracker::WordTracker()
```



```
{
    // nothing to do currently
}

void WordTracker::readFile(const string& filename)
{
    ifstream input;
    input.open(filename.c_str());
    string word;
    while (input >> word){
        ToLower(word);                              //these lines make the code l
ower case and remove
        StripPunc(word);                            //punctuation
        myMap[word]++;
    }
}


void WordTracker::read()
{

    string word;
    while (cin >> word){
        ToLower(word);                              //these lines make the code l
ower case and remove
        StripPunc(word);                            //punctuation
        myMap[word]++;
    }
}

bool sortoccurs(pair<string,int>& lhs,
                pair<string,int>& rhs)
{
      if(lhs.second == rhs.second)
        {
            if(lhs.first > rhs.first)
              {
                  return false;                     //allows ties to be broken al
phabetically
              }
        }
    else return lhs.second > rhs.second;
}


void WordTracker::top(int n, vector<pair<string,int> >& v)
{
    v.clear();

    map<string,int>::iterator it = myMap.begin();
    while (it != myMap.end()){
        v.push_back(*it);
        it++;
        n++;
    }

    partial_sort(v.begin(), v.begin()+n, v.end(), sortoccurs);
    //sort(v.begin(), v.end(),sortoccurs);
}


int main(int argc, char *argv[])
{
    Printer * printer = new ColumnPrinter();
    ifstream input;
    string filename,line,w;
    int wordcount = 20;
    filename = "NULL";
    int cols = 1;                   //number of columns, set default 1
```

```
    struct option long_ops[] = {
        {"filename", optional_argument, 0, 'f'},
        {"count", optional_argument, 0, 'c'},
        {"width", optional_argument, 0, 'w'}
    };

    string shortArgs = "f:c:w:";
    extern char * optarg;
    char ch;

    while((ch = getopt_long(argc,argv,shortArgs.c_str(), long_ops, NULL)) != -1)
{
        switch (ch){
        case 'f':                           //filename
            filename = optarg;
            break;
        case 'w':                           //width
            cols = atoi(optarg);
            break;
        case 'c':
            wordcount = atoi(optarg);
            break;
        case '?':
            break;
        default:
            cout << "Use the right args" << endl;
        }
    }



    WordTracker tracker;

    if(filename=="NULL") tracker.read();
    else tracker.readFile(filename);
    vector<pair<string,int> > v;
    tracker.top(0,v);
    int vectsize;
    if(v.size() < wordcount) vectsize = v.size();
    else vectsize = wordcount;
    printer->setCols(cols);
    vector<pair<string,int> > v2(vectsize);

    for(int i=0; i<vectsize; i++)
    {
        v2[i].first = v[i].first;
        v2[i].second = v[i].second;
    }

    printer->print(cout,v2);
    /*
    for(int k=0; k < vectsize; k++){
        cout << v[k] << endl;
    }
    */
    return 0;
}
```



```
//Name:                Matthew Territo and Mack Liu
//Course:              CPS108
//Assignment #1:       Wordcount Version 2.0


#include <istream>
#include <iostream>
#include <fstream>
#include <vector>
#include <map>
#include <algorithm>
#include <string>
#include <cstdlib>
#include <getopt.h>
using namespace std;

//-------------------------------------------------------------------------------
--------------
//-------------------------------------------------------------------------------
--------------
//                                    HELPER FUNCTIONS

//-------------------------------------------------------------------------------
--------------
//-------------------------------------------------------------------------------
--------------


static struct option long_ops[] =
{
    {"file",      required_argument,0,'f'},
    {"count",     required_argument,0,'c'},
    {"width",     required_argument,0,'w'},

    {0,0,0,0}
};

static string shortArgs = "f:c:w:";



void removePunct(string& word)
//eliminate leading and trailing punctuation
{
    int first = 0;
    int len = word.length();
    while (first < len && ispunct(word[first]))
    {
        first++;
    }

    int last = len - 1;
    while (last >= 0 && ispunct(word[last]))
    {
        last--;
    }
    word = word.substr(first,last-first+1);
}

void lowercase(string& word)
//make alphabetic character lowercase
{
    int len = word.length();
    for(int k=0; k < len; k++)
    {
        word[k] = tolower(word[k]);
    }
}
```



```
bool sortoccurs(const pair<string,int>& lhs,
                const pair<string,int>& rhs)
{
    if (lhs.second == rhs.second)
        return lhs.first < rhs.first;

    return lhs.second > rhs.second;
}

int getlength(int i)
{
    int l = 0;
    if (i == 0)
        return 1;

    while (i != 0)
    {
        i = i/10;
        l++;
    }
    return l;
}


//-------------------------------------------------------------------------------
--------------
//-------------------------------------------------------------------------------
--------------
//                          CLASS FOR TRACKING WORDS + IMPLEMENTATION

//-------------------------------------------------------------------------------
--------------
//-------------------------------------------------------------------------------
--------------




class WordTracker
{

  public:
    WordTracker() {};
    ~WordTracker() {};

    void process(istream& input, unsigned int& n)
    //counts the number of occurrances of each distinct word
    {
        string word;
        while(input >> word)
        {
            removePunct(word);
            lowercase(word);
            myMap[word]++;
        }

        top(n);
    }

    void print(unsigned int& n, int w)
    {
        cout << endl;

        int rows = n/w;
        if (n % w != 0)
            rows++;

        vector<int> maxint(w,0);
```

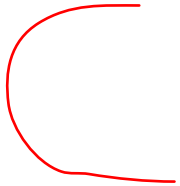
```
        for(int k=0; k<w; k++)
        {
            maxint[k] = getlength(myTop[k].second);
        }

        vector<int> max(w,0);

        // find max length string in each column
        for(int k=0; k<n; k++)
        {
            int index = k % w;
            if (myTop[k].first.length() > max[index])
                max[index] = myTop[k].first.length();
        }

        for(int k=0; k < rows; k++)
        {
            for(int j = k*w; j < (k+1)*w; j++)
            {
                if (j >= n)
                    break;

                int index = j % w;
                tab(maxint[index] - getlength(myTop[j].second));
                cout << myTop[j].second;
                tab(2);
                cout << myTop[j].first;
                tab(max[index] - myTop[j].first.length() + 4);
            }
            cout << endl;
        }
    }

  private:

    map<string, int>  myMap;
    vector<pair<string,int> >  myTop;

    void top(unsigned int& n)
    {
        myTop.clear();
        map<string,int>::iterator it = myMap.begin();
        while (it != myMap.end()){
            myTop.push_back(*it);
            it++;
        }
        if (n > myTop.size())
            n = myTop.size();

        partial_sort(myTop.begin(), myTop.begin()+n, myTop.end(), sortoccurs);
    }

    void tab(int n)
    {
        for(int k=0; k<n; k++)
        {
            cout << " ";
        }
    }
};




//--------------------------------------------------------------------------------
--------------
//--------------------------------------------------------------------------------
--------------
//                                    MAIN PROGRAM
```



```
//--------------------------------------------------------------------------------
--------------
//--------------------------------------------------------------------------------
--------------




int main(int argc, char* argv[])
{
    int w = 1;
    unsigned int n = 20;
    extern char * optarg;
    char ch;

    WordTracker tracker;


    string filename = "";

    while( (ch = getopt_long(argc, argv, shortArgs.c_str(), long_ops, NULL)) !=
-1)
    {
        switch(ch)
        {
        case 'c':
            n = atoi(optarg);
            break;
        case 'f':
            filename = optarg;
            break;
        case 'w':
            w = atoi(optarg);
            break;
        }

    }
    if(filename != "")
    {
        ifstream infile(filename.c_str());
        tracker.process(infile, n);
    }

    else
    {
        tracker.process(cin, n);
    }

    tracker.print(n, w);
    return 0;
}
```

```
//Names: Andrew Portnoy
//       Robert DeMason
//       Sandeep Kagzi
//Course: CPS 108
//Date: 01/27/2004

#include <fstream>
using namespace std;

#include "wordsprocessor.h"

void WordsProcessor:: process(const string& filename, int count,
                              Formatter* format, Reader* reader,
                              Sorter* sorter, Printer* printer)
    //post: processes and prints out the count most frequently
    //occuring words
{
    if(filename == "")
        reader->read(cin, format, myMap);
    else
    {
        ifstream fin(filename.c_str());
        reader->read(fin, format, myMap);
    }
    map<string,int>::iterator it = myMap.begin();
    while (it != myMap.end())
    {
        myPairs.push_back(*it);
        it++;
    }
    if(count >= myPairs.size())
        sort(myPairs.begin(),myPairs.end(),*sorter);
    else
        partial_sort(myPairs.begin(),myPairs.begin()+count, myPairs.end(),
                     *sorter);
    printer->print(cout, count, myPairs);
}
```

```
//Names: Andrew Portnoy
//       Robert DeMason
//       Sandeep Kagzi
//Course: CPS 108
//Date: 01/27/2004

#ifndef _WORDSPROCESSOR_H
#define _WORDSPROCESSOR_H

#include <vector>
#include <map>
#include <string>
using namespace std;

#include "sorter.h"
#include "reader.h"
#include "formatter.h"
#include "printer.h"

class WordsProcessor
{
public:
    WordsProcessor()
    {

    }
    virtual ~WordsProcessor()
    {

    }
    virtual void process(const string& filename, int count,
                         Formatter* format, Reader* reader,
                         Sorter* sorter, Printer* printer);
private:
    map<string, int> myMap;
    vector<pair<string, int> > myPairs;
};

#endif
```



```
/*
  Name:        Vinh Nguyen
  Partner:     Erik Schmidt
  Date Due:    1/12/04
  Course:      CPS 108
  Assignment:  wordlines
  Program:     wordcount
  Version:     2.0
  Purpose:     This program reads a text file and
               determines the most commonly occurring
               words in the file.
*/

#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <cstdio>
#include <cctype>
#include <stdexcept>
using namespace std;

#include <unistd.h>
#include "ctimer.h"
#include <getopt.h>

#include "wordtracker.h"
#include "reader.h"
#include "wordsorter.h"
#include "printer.h"

void usage()
{
  cout<<"wordcount -f filename -c wordcount"<<endl;
  cout<<" wordcount --file[=filename] --count[=wordcount]"<<endl;
}


static struct option long_ops[] = {
    {"file",       no_argument,        0,'f'},
    {"count",       no_argument,      0,'c'},
    {"width",       no_argument,      0,'w'},


    {0,0,0,0}
};

static string shortArgs = "f:c:w:";

int main(int argc, char * argv[])
{
    // Variables for parsing command line arguments
    string filename = "";
    int width= 1;
    int count = 20;
    extern char * optarg;
    char ch;

    Sorter * sorter = new WordSorter();
    Reader * reader = new SimpleReader();
    Printer * printer = new ColumnPrinter();
    ifstream input;
    bool fromFile = false;
    bool abort = false;

    WordTracker tracker;

    while ( (ch = getopt_long(argc,argv, shortArgs.c_str(),
                              long_ops,NULL)) != -1){
```



```
        switch (ch){
           case 'c':
               count = atoi(optarg);
               break;
           case 'f':
               filename = optarg;
               input.open(filename.c_str());
               // Check for invalid filename
               if (!input)
               {
                   cout << "wordcount: " << "Invalid file!" << endl;
                   abort = true;
               }

               fromFile = true;
               break;
           case 'w':
               width = atoi(optarg);
               break;
           default:
              usage();
              abort = true;
        }
    }

    if (!abort)
    {
        // Setup printer
        printer->setNumWords(count);
        printer->setCols(width);

        // Setup tracker
        tracker.setReader(reader);
        tracker.setSorter(sorter);
        tracker.setPrinter(printer);
        if (fromFile)
        {
            tracker.setInput(input);
        }
        else
        {
            tracker.setInput(cin);
        }

        tracker.setOutput(cout);

        // Load words into tracker
        tracker.readWords();

        // Sort words in tracker
        tracker.sortWords();

        // Display results
        tracker.showWords();
    }

    return 0;
}
```

```
/*
  Name:      Vinh Nguyen
  Partner:   Erik Schmidt
  Date Due:  1/26/04
  Course:    CPS 108
  Purpose:   This file is the implementation file for the
             WordTracker class.  The WordTracker class defines an
             object used to count the number of unique words in a
             file and to output the results.
*/

#ifndef _WORDTRACKER_CPP
#define _WORDTRACKER_CPP

#include "wordtracker.h"
#include <vector>
#include <iostream>
#include <algorithm>
#include "reader.h"
#include "sorter.h"
#include "printer.h"

// Constructors and destructors

WordTracker::WordTracker()
{
}

WordTracker::~WordTracker()
{
}

// Accessors

Reader  * WordTracker::getReader()
// Post:  Returns a copy of myReader
{
    return myReader;
}

Sorter * WordTracker::getSorter()
// Post:  Returns a copy of mySorter
{
    return mySorter;
}

Printer * WordTracker::getPrinter()
// Post:  Returns a copy of myPrinter
{
    return myPrinter;
}

istream * WordTracker::getInput()
// Post:  Returns a copy of myInput
{
    return myInput;
}

ostream * WordTracker::getOutput()
// Post:  Returns a copy of myOutput
{
    return myOutput;
}

void WordTracker::showWords()
// Post:  myPrinter is used to output myContents
{
    myPrinter->print(*myOutput, myContents);
}
```



```
// Mutators

void WordTracker::setReader (Reader * & targetReader)
// Pre:   targetReader is a Reader
// Post:  targetReader is bound to the WordTracker by assigning
//        targetReader to myReader
{
    myReader = targetReader;
}

void WordTracker::setSorter (Sorter * & targetSorter)
// Pre:   targetSorter is a Sorter
// Post:  targetSorter is bound to the WordTracker by assigning
//        targetSorter to mySorter
{
    mySorter = targetSorter;
}

void WordTracker::setPrinter (Printer * & targetPrinter)
// Pre:   targetPrinter is a Printer
// Post:  targetPrinter is bound to the WordTracker by assigning
//        targetPrinter to myPrinter
{
    myPrinter = targetPrinter;
}

void WordTracker::setInput (istream & targetInput)
// Pre:   targetInput is an istream
// Post:  targetInput is assigned to myInput
{
    myInput = &targetInput;
}

void WordTracker::setOutput (ostream & targetOutput)
// Pre:   targetOutput is an ostream
// Post:  targetOutput is assigned to myOutput
{
    myOutput = &targetOutput;
}

void WordTracker::setAll (Reader * & targetReader,
                          Sorter * & targetSorter,
                          Printer * & targetPrinter,
                          istream & targetInput,
                          ostream & targetOutput)
    // Pre:   targetReader is a Reader,
    //        targetSorter is a Sorter,
    //        targetPrinter is a Printer
    //        targetInput is an istream
    //        targetOuput is an ostream
    // Post:  targetReader, targetSorter, and targetPrinter are bound
    //        to the WordTracker by assigning targetReader to
    //        myReader, targetSorter to mySorter, targetPrinter to
    //        myPrinter, targetInput to myInput, and targetOutput to
    //        myOutput
{
    setReader(targetReader);
    setSorter(targetSorter);
    setPrinter(targetPrinter);
    setInput(targetInput);
    setOutput(targetOutput);
}

void WordTracker::readWords()
// Post:  myReader is used to read words into myContents
{
    // Use myReader to read words from myInput
    myReader->read(*myInput);
```

```
    // Get the unsorted vector of words from myRead and store the
    //     words in myContents
    myContents = myReader->getVector();
}

void WordTracker:: sortWords()
// Post:  mySorter is used to sort myContents
{
    // Sort myContents using mySorter
    sort(myContents.begin(), myContents.end(), *mySorter);
}

#endif
```

```
/*
  Name:      Vinh Nguyen
  Partner:   Erik Schmidt
  Date Due:  1/26/04
  Course:    CPS 108
  Purpose:   This file is the header file for the WordTracker class.
             The WordTracker class defines an object used to count
             the number of unique words in a file and to output the
             results.
*/

#ifndef _WORDTRACKER_H
#define _WORDTRACKER_H

#include <iostream>
#include <vector>
#include "reader.h"
#include "sorter.h"
#include "printer.h"

class WordTracker
{
  public:
    // Constructors and destructors
    WordTracker();
    WordTracker(WordTracker &);
    ~WordTracker();

    // Accessors
    Reader  * getReader();
    // Post:  Returns a pointer to of myReader

    Sorter * getSorter();
    // Post:  Returns a pointer to mySorter

    Printer * getPrinter();
    // Post:  Returns a pointer to myPrinter

    istream * getInput();
    // Post:  Returns a copy of myInput

    ostream * getOutput();
    // Post:  Returns a copy of myOutput

    void showWords();
    // Post:  myPrinter is used to output myContents

    // Mutators
    void setReader (Reader * & targetReader);
    // Pre:   targetReader is a Reader
    // Post:  targetReader is bound to the WordTracker by assigning
    //        targetReader to myReader

    void setSorter (Sorter * & targetSorter);
    // Pre:   targetSorter is a Sorter
    // Post:  targetSorter is bound to the WordTracker by assigning
    //        targetSorter to mySorter

    void setPrinter (Printer * & targetPrinter);
    // Pre:   targetPrinter is a Printer
    // Post:  targetPrinter is bound to the WordTracker by assigning
    //        targetPrinter to myPrinter

    void setInput (istream & targetInput);
    // Pre:   targetInput is an istream
    // Post:  targetInput is assigned to myInput

    void setOutput (ostream & targetOutput);
    // Pre:   targetOutput is an ostream
```



```
    // Post:  targetOutput is assigned to myOutput

    void setAll (Reader * & targetReader, Sorter * & targetSorter,
                 Printer * & targetPrinter, istream & targetInput,
                 ostream & targetOutput);
    // Pre:   targetReader is a Reader,
    //        targetSorter is a Sorter,
    //        targetPrinter is a Printer
    //        targetInput is an istream
    //        targetOuput is an ostream
    // Post:  targetReader, targetSorter, and targetPrinter are bound
    //        to the WordTracker by assigning targetReader to
    //        myReader, targetSorter to mySorter, targetPrinter to
    //        myPrinter, targetInput to myInput, and targetOutput to
    //        myOutput

    void readWords();
    // Post:  myReader is used to read words into myContents

    void sortWords();
    // Post:  mySorter is used to sort myContents

  private:
    Reader * myReader;
    Sorter * mySorter;
    Printer * myPrinter;
    vector <pair <string, int> > myContents;
    istream * myInput;
    ostream * myOutput;
};

#include "wordtracker.cpp"
#endif
```