// Author: Joan Guzman

// CIS 3100 BMWA Spring 2023

// Assignment Grade:

//Header File (named “HEADER.h):

#ifndef HEADER\_H

#define HEADER\_H

#include <iostream>

#include <fstream>

#include <iomanip>

#include <string>

using namespace std;

const int COLUMN\_WIDTH = 60;

bool isAcceptable(char c)

{

return (((c >= 48) && (c <= 57)) || ((c >= 65) && (c <= 90)) || ((c >=

97) && (c <= 122)));

}

class MyString {

private:

string str;

int frequency;

public:

MyString() {

str = "";

frequency = 0;

}

MyString(string s) {

str = s;

frequency = 1;

}

bool operator==(string rhs) const

{

return (str == rhs);

}

bool operator >(MyString s) const

{

if (frequency == s.frequency) {

return str < s.str;

}

return frequency > s.frequency;

}

bool operator <(MyString s) const

{

return frequency < s.frequency;

}

MyString operator ++(int)

{

MyString temp = \*this;

frequency++;

return temp;

}

friend ofstream& operator<<(ofstream& o, const MyString& obj);

};

// output to file

ofstream& operator<<(ofstream& o, const MyString& obj) {

o.width(COLUMN\_WIDTH);

o << left << obj.str << obj.frequency << endl;

return o;

}

bool freqcompare(const MyString& val1, const MyString& val2) {

return val1 > val2;

}

#endif

//CPP file:

#include <iostream>

#include "HEADER.h"

#include <fstream>

#include <iomanip>

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

string filein, fileout;

int wordcount = 0;

vector <string> speech;

vector <MyString> words;

string word;

string newword = "";

bool add;

cout << "Enter the source data file name: ";

cin >> filein;

fstream file(filein);

while (wordcount < 1 || wordcount > 5) {

cout << "How many Adjacent words in a phrase, enter 1-5: ";

cin >> wordcount;

}

cout << "Enter the phrase frequency file name: ";

cin >> fileout;

ofstream outfile(fileout);

while (file >> word) {

int ind = 0;

int sind;

for (int i = 0; i < word.length(); i++) {

if (!isAcceptable(word[i])) {

ind++;

continue;

}

break;

}

for (int i = word.length(); i > 0; i--) {

if (!isAcceptable(word[i])) {

sind = i;

continue;

}

break;

}

for (int x = ind; x < sind; x++) {

if (isAcceptable(word[x]) || word[x] == '-') {

newword += word[x];

}

}

if (newword.size() == 0) { // if blank space

continue;

}

transform(newword.begin(), newword.end(), newword.begin(), ::toupper);

speech.push\_back(newword);

newword = "";

}

for (int ind = 0; ind <= (speech.size() - wordcount); ind++) {

add = true;

newword = speech[ind];

for (int j = 1; j < wordcount; j++) {

if (speech.size() >= (ind + wordcount)) {

newword += " " + speech[ind + j];

}

}

if (!words.empty()) { // if the vector is NOT empty

for (int x = 0; x < (words.size() - 1); x++) {

if (words[x] == newword) {

words[x]++;

add = false;

break;

}

}

}

if (add) {

words.push\_back(MyString(newword));

}

newword = "";

}

sort(words.begin(), words.end(), freqcompare); // Need to make it alpha order as well

outfile << "The file: "<< fileout << " contains " << speech.size() << " words, and " << words.size() << " phrases." << endl;

for (int x = 0; x < words.size(); x++) {

outfile << words[x];

}

}

Program output:

A computer screen with white text

Description automatically generated with low confidence