# Assignment Description

Write a program that reads words from a text file and displays all the non-duplicate words . Use a set container to accumulate non-duplicate words and display a list of the unique words in ascending order at the end of your program.

# 1 Readme Documentation

This program will read in a file containing words into a vector. Then, it will identify the non-duplicate words and display them.

# 2 Flowchart Screen Shots

# 3 UML and Use Case Diagrams

# 4 Source Code of All files (.h, .cpp)

#include *<iostream>*

#include *<iomanip>*

#include *<string>*

#include *<cctype>*

#include *<stdexcept>*

#include *<vector>*

#include *<fstream>*

#include *<algorithm>*

#include *<set>*

#include *<limits>*

**using** **namespace** **std**;

*/\**

*Program Name: Find Unique Words*

*Author: Wesley Hixon*

*Date Last Updated: 11/10/2024*

*Purpose: Reads*

*\*/*

vector<string> getWords(string filename);

set<string> getUniqueWords(vector<string> words);

int main(){

vector<string> words;

*// Gets a vector of words from input file "words.txt"*

**try**{

words = getWords("words.txt");

}*// If file opening failed, output error message and exit program.*

**catch**(**const** ifstream::failure& e){

perror(e.what());

**return** 0;

}

*// Makes a vector of unique words from words vector*

set<string> uniqueWords = getUniqueWords(words);

*// Output list of unique words*

cout << "The following is the unique words in the list of words provided: " << endl;

*// Creates pointer which iterates through set*

**for**(**auto** i = uniqueWords.begin(); i != uniqueWords.end(); i++){

*// Dereference pointer and output word*

cout << \*i << endl;

}

**return** 0;

}

vector<string> getWords(string filename){

vector<string> words;

ifstream inputFile;

**try**{

*// Automatically throws exception if there is an error reading file*

inputFile.exceptions(ifstream::badbit | ifstream::failbit);

inputFile.open(filename);

*// Get word and add to vector*

string word;

**while**(!inputFile.eof()){

getline(inputFile, word);

words.push\_back(word);

}

}**catch**(**const** ifstream::failure& e){

*// If file failed to open, add error message and throw to main()*

**if**(!inputFile.is\_open()){

string errorMessage = "Error opening file '" + filename +"': " + e.what();

**throw** ifstream::failure(errorMessage);

}

*// If getline failed, skip line and keep going*

**else** **if**(inputFile.fail()){

cerr << "Error reading file contents: " << e.what() << endl;

cerr << "Skipping line and continuing..." << endl;

inputFile.clear();

inputFile.ignore(10000, '\n');

}

}

*// Close file*

**if**(inputFile.is\_open()){

inputFile.close();

}

**return** words;

}

set<string> getUniqueWords(vector<string> words){

vector<string> wordsRead;

set<string> uniqueWords;

*// Go through words vector*

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

string currentWord = words[i];

*// Checks if the word being read has been seen before*

bool wordAlreadyRead = find(wordsRead.begin(), wordsRead.end(), currentWord) != wordsRead.end();

*// Checks if word is in unique words list*

bool wordIsUnique = uniqueWords.find(currentWord) != uniqueWords.end();

*// If word hasn't been seen before*

**if**(!wordAlreadyRead){

*// Add to vector of unique words*

uniqueWords.insert(words[i]);

}

*// If it has been seen before and it's in uniqueWords*

**else** **if**(wordIsUnique){

*// Get an iterator that points to the word in uniqueWords*

**auto** it = find(uniqueWords.begin(), uniqueWords.end(), currentWord);

*// Delete it*

uniqueWords.erase(it);

}

*// Add word to list of words already read*

wordsRead.push\_back(words[i]);

}

**return** uniqueWords;

}

# 5 Three Use Case Screen Shots

Test Data 1:

client

death

government

explanation

proposal

client

basis

health

nation

platform

mud

mom

bonus

midnight

nation

method

lab

distribution

distribution

news

college

movie

sympathy

improvement

setting

anxiety

movie

anxiety

expression

actor

writer

recommendation

promotion

obligation

student

actor

television

math

driver

television

entry

potato

stranger

definition

farmer

contribution

lab

contribution

gate

college

moment

law

depression

recognition

son

consequence

patience

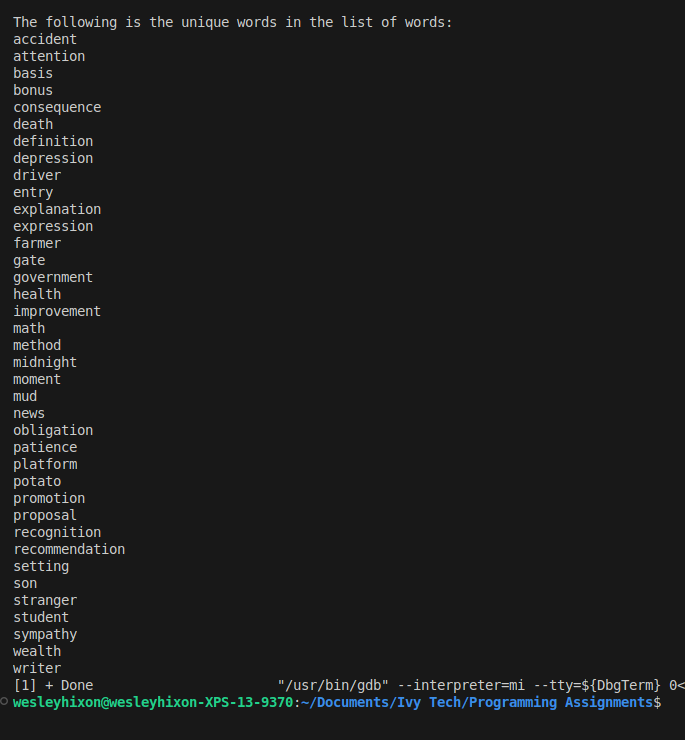
wealth

accident

attention

mom

law

Output 1:

Test data 2:

speed

common

hungry

impossible

shed

shed

tire

tire

sneak

declare

naive

fast

brush

misuse

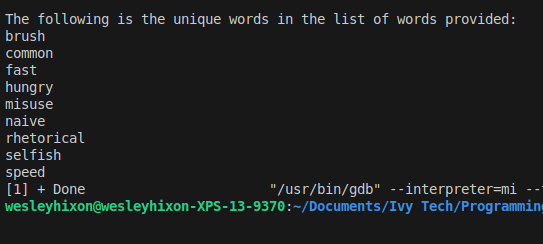
rhetorical

selfish

sneak

declare

impossible

Output 2:

Test Data 3:

dissection

description

luxury

fill

band

traditional

prune

windowless

exterior

10

spoiled

thai

deceased

filtering

apple

preen

positioning

screen

crushed

coke

drunkenness

deceased

thai

exterior

luxury

description

Output 3:

