CSE 2341 Data Structures

Automatic Wordsearch Solving

Luke Wood

Project Description

This program is written in C++ and automatically solves word searches. The program can receive as many input files at once as the user desires and will solve the word searches as long as the input file is correctly formatted. The file consists of a word search prior to the number of words in the word bank, and the words to be found following the number of words.

Sample Input File:

```
catdog
meowhi
thisis
awords
seemse
5
sat
this
meow
dog
hi
```

Implementation

The implementation for this project was fairly simple. To begin, I simply read in the wordsearch while checking each line to make sure that it does not consist of only a single number. Following this, I then read in the words into a dynamically allocated array of type std::string. I then proceed to loop through the word search until a character matching one of the first characters in the word bank is found. Following this I determine the direction that the word is going and follow along that path until either the word ends

or there is an incorrect character. If the full word is found, this location is marked and is then printed to the output file at the end of the run.

Code

The Code below displays the implementation full source code of the project. Due to the simplicity of the project, I implemented no classes, one type and one only struct.

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <vector>

using namespace std;
//TYPE DECLARATIONS BEGIN

//Creating a Point type to store in my vector of answers
typedef std::pair<int, int> Point;

//Creating a struct for each word we find containing the point at and word name.
struct foundword
{
	string word;
	Point location;
	string dir;
};

//TYPE DECLARATIONS END

//FUNCTION DECLARATIONS BEGIN

//Creating a pseudo constructor to make it easier to get a foundword
foundword create_found_word(string word, int x,int y, string dir)
{
	foundword toreturn;
	Point temp(x,y);
	toreturn.word = word;
	toreturn.location = temp;
	toreturn.dir = dir;
	cout<<word<<"|"<<x<<"|"<<y<<"|"<<dir<<"\n";
	return toreturn;
}

//determine if the word was found
bool wordFound(string word, vector<foundword> fwords)
{
	bool toreturn = false;
	for(int i = 0; i < fwords.size(); i++)
	{
		if(word.compare(fwords[i].word) ==0)
		{
			toreturn = true;
		}
	}
	return toreturn;
```

```
}

//determine the direction string based on the input numbers
string determineDir(int dy,int dx)
{
	string dir = "";
	if(dy==-1)dir+="u";
	if(dy ==1)dir+="d";
	if(dx ==1)dir+="r";
	if(dx==-1)dir+="l";
	return dir;
}

//Basic function to determine if the char getting input is a digit
bool isNum(char c)
{
   return (c >='0' && c <='9');
}

//Function to write a foundword to a file
void write_found_word(const foundword wfound, ofstream& file)
{
file<<wfound.word<<"|"<<wfound.location.first<<"|"<<wfound.location.second<<"|"<<wfound.dir<
<"\n";
}

//Reformats the .in filename to a .out filename
string reformatName(string fileName)
{
string outName = fileName.substr(0,fileName.find_last_of("."));
outName = outName+".out";
return outName;
}

//FUNCTION DECLARATIONS END

//MAIN BEGIN
int main(int argc, char*argv[])
{
//Check to ensure that there are two strings passed to the program, the input file then
output file.
	if(!(argc>=2))
	{
		cout<<"need an input file.."<<"\n";
		return 0;
	}
	for(int runCount = 1; runCount < argc; runCount++)
	{
		//Continue on in program, declaring the variables I will need.
		ifstream in;
		ofstream out;
		vector<string> wordSearch;
		string* wordBank;
		vector<foundword> found;

		int wbanklen = 0;
		int height = 0;//Word search height.

		//Trying to open the ifstream to the file input and read each line into the
vector of sting, keep track of size before \n.
```

```
		in.open(argv[runCount]);
		if(in.is_open())
		{
			cout<<"Reading in the word search.\n";
			string x;
			bool done = false;
			while (in>>x && !done)
			{
				if(isNum(x[0]))
				{
					done = true;
					wbanklen = atoi(x.c_str());
					//Read in length and convert to wbanklen using cstdlib
					//break was neccesary to prevent another the first word
in the word bank from being thrown away
					break;
				}
				else
				{
					wordSearch.push_back(x);
					height++;
				}
			}

			//Just showing that I have successfully read in the word search and
closing the ifstream.
			cout<<"Full Word Search:\n\n";
			for(int i = 0; i < wordSearch.size(); i++)
			{
				cout<<wordSearch[i]<<"\n";
			}
			cout<<"\n"<<"Reading in word bank.\n";
			//Reading in the word bank.
			wordBank = new string[wbanklen];
			for(int i = 0; i < wbanklen;i++)
			{
				string word;
				in>>word;
				wordBank[i] = word;
			}
			in.close();
			//file input is now complete.
			cout<<"Beginning to solve word search.\n";
			for(int i = 0; i <height;i++)
			{
				//Loop through each row
				for(int j = 0; j < wordSearch[0].length(); j++)
				{
					//Loop through each column
					for(int t = 0; t < wbanklen;t++)
					{
						//Check each word in word bank.
						if(wordSearch[i][j] == wordBank[t][0])
						{
							if(wordBank[t].length()<=1)
							{

found.push_back(create_found_word(wordBank[t],j,i,"N/A"));
							}
							else
```

```
                                                {
                                                        //Find direction
                                                        char toFind = wordBank[t][1];
                                                        bool itDone;
                                                        for(int x = -1; x <=1; x++)
                                                        {
                                                                for(int y = -1; y <=1;
y++)
                                                                {
                                                                        //Check that it
isnt the starting square and that everything is in index
                                                                        if((x!=0 ||
y!=0)&&!(i+x<0||i+x>=height)&&!(j+y<0||j+y>=wordSearch[0].length()))
                                                                        {

if(wordSearch[x+i][y+j] ==toFind)
                                                                                {
                                                                                        int
wlen = wordBank[t].length()-1;

//final i and final j
                                                                                        int
fi = i+x*wlen;
                                                                                        int
fj = j+y*wlen;
                                                                                        //If
the final i and final j are not outside the array it is automatically false.
                                                                                        bool
isRight = (!(fi < 0 || fi >height)&&!(fj<0||fj>wordSearch[0].length()));

if(wlen >=3)
                                                                                        {
        for(int z = 2; z <= wlen&&isRight; z++)
        {
                //Determine if each character in the wordBank matches
                if(!(wordBank[t][z]==wordSearch[i+(x*z)][j+(y*z)]))
                {
                        isRight =false;
                }
        }
                                                                                        }
//Finally, add in the word to the found list.

if(isRight)
                                                                                        {

        found.push_back(create_found_word(wordBank[t],j,i,determineDir(x,y)));
                                                                                        }
                                                                                }
```

```
                                                                        }
                                                                }
                                                        }
                                                }
                                        }
                                }
                        }
                }
                else
                {
                        cout<<"File was not opened.\n";
                }
                string tempfname = argv[runCount];
                out.open(reformatName(tempfname).c_str());
                //Write each word to the file.
                if(out)
                {
                        //Write out words that were found
                        for(int i = 0; i < found.size(); i++)
                        {
                                write_found_word(found[i],out);
                        }
                        //Write out words that werent found.
                        for(int i = 0; i < wbanklen; i++)
                        {
                                if(!wordFound(wordBank[i],found))
                                {

write_found_word(create_found_word(wordBank[i],-1,-1,"nf"),out);
                                }
                        }
                }
                out.close();

                //End Program
                delete[] wordBank;
        }
        return 0;
}
```

Results and Discussion

This program turned out to be very successful at solving word searches. Assuming correct file formatting, this project performed the task at hand flawlessly and allowed for even large word searches to be completed.

Concluding Remarks

While this project was simple, I felt that I learned slightly more about parsing files in C++ and managing the data at hand.