CS2263 Assignment 4

Stephen Cole

3553803

**htags.c**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<stdbool.h>

#include"htags.h"

#define MAX\_TYPE\_TAGS 99

#define MAX\_TAG\_SIZE 20

#define MAX\_NUM\_TAGS 5000

#define MAX\_STRING\_LENGTH 255

void instantiateInputArray(char\*\* inputArr, char\* line, FILE\* fp)

{

int i = 0;

while(fgets(line, MAX\_STRING\_LENGTH, fp))

{

inputArr[i] = (char\*)malloc(MAX\_STRING\_LENGTH);

strcpy(inputArr[i], line);

i++;

}

}

int getlines(FILE\* fp, char\* line)

{

int numLines=0;

while(fgets(line, MAX\_STRING\_LENGTH, fp))

numLines++;

rewind(fp);

return numLines;

}

void scanWord(char\* start, char\* word)

{

int counter = 0;

if(\*start == '<')

start++;

while(\*start != '>' && \*start != ' ' && \*start != '/')

{

word[counter] = \*start;

counter++;

start++;

}

word[counter] = '\0';

return;

}

**htags.h**

#ifndef INTEGER\_H

#define INTEGER\_H

void instantiateInputArray(char\*\* inputArr, char\* line, FILE\* fp);

int getlines(FILE\* fp, char\* line);

void scanWord(char\* start, char\* word);

#endif

**findtags.c**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<stdbool.h>

#include"htags.h"

#define MAX\_TYPE\_TAGS 99

#define MAX\_TAG\_SIZE 20

#define MAX\_NUM\_TAGS 5000

#define MAX\_STRING\_LENGTH 255

/\*

\* This whole program is a mess

\*

\*/

int main(int argc, char\* argv[])

{

FILE \*fp;

char \*line = (char\*)malloc(MAX\_STRING\_LENGTH);

fp = fopen(argv[1], "r");

if(!fp)

perror(argv[1]), exit(1);

int numLines = getlines(fp, line);

char\*\* inputArr = (char\*\*)malloc(numLines\*sizeof(char\*));

instantiateInputArray(inputArr, line, fp);

char\*\* tagTable = (char\*\*)malloc(MAX\_NUM\_TAGS\*sizeof(char\*));

for(int i=0;i < MAX\_NUM\_TAGS;i++){

tagTable[i] = (char\*)malloc(MAX\_TAG\_SIZE);

}

int tagIndex = 0;

for(int i=1; i<numLines; i++)

{

char\* tagLocation;

tagLocation = strstr(inputArr[i], "<");

if(tagLocation != NULL)

{

if(\*(tagLocation + 1) == '/' || \*(tagLocation + 1) == '!')

{

if(\*(tagLocation + 2) == '-')

continue;

tagLocation++;

tagLocation++;

}

scanWord(tagLocation, tagTable[tagIndex]);

tagIndex++;

tagLocation = strstr((tagLocation + 1), "<");

while(tagLocation != NULL)

{

if(\*(tagLocation + 1) == '/' || \*(tagLocation + 1) == '!')

{

if(\*(tagLocation + 2) == '-')

break;

tagLocation++;

tagLocation++;

}

scanWord(tagLocation, tagTable[tagIndex]);

tagIndex++;

tagLocation = strstr((tagLocation + 1), "<");

}

}

}

char\*\* foundTags = (char\*\*)malloc(MAX\_TYPE\_TAGS\*sizeof(char\*));

for(int i=0;i < MAX\_TYPE\_TAGS;i++){

foundTags[i] = (char\*)malloc(MAX\_TAG\_SIZE);

}

int numFoundTags = 0;

bool check = true;

for(int i=0;i<tagIndex; i++)

{

for(int j=0;j<numFoundTags;j++)

{

if(strcmp(tagTable[i],foundTags[j]) == 0)

{

check = false;

break;

}

}

if(check)

{

strcpy(foundTags[numFoundTags],tagTable[i]);

int tagCount = 1;

for(int k=i+1; k<tagIndex; k++)

{

if(strcmp(foundTags[numFoundTags],tagTable[k]) == 0)

tagCount++;

}

printf("%s\t%d\n",foundTags[numFoundTags], tagCount);

numFoundTags++;

}

check = true;

}

fclose(fp);

free(line);

free(tagTable);

free(inputArr);

free(foundTags);

return EXIT\_SUCCESS;

}

makefile

htags: findtags.c htags.c

gcc -std=c99 -o htags -Wall findtags.c htags.c

test: sample\_test hello\_test

sample\_test: htags

./htags ./A4Data/Sample.html > sample\_test.result

./TestPassed.sh ./sample\_test.result ./A4Data/sample\_test.expected

hello\_test: htags

./htags ./A4Data/HelloWorld.html > hello\_test.result

./TestPassed.sh ./hello\_test.result ./A4Data/hello\_test.expected

A screenshot of a cell phone

Description automatically generated

outputs:

**sample\_test.result**

html 2

head 2

meta 1

title 2

body 2

strong 2

ol 2

li 4

blink 2

p 4

**hello\_test.result**

html 2

head 2

meta 1

title 2

body 2

p 2

The data structures in this program include, inputArr which is a string array containing each line from the html file passed in. tagTable is another string array containing every tag found in the inputArr. The last data structure is foundTags which contains the tags that have already been printed out.