Assignment Three

Solution

Data Structures

The file will be read into a single character string, with a maximum size of 10,000 characters.

Unique tag names will be identified in an index array of character pointers holding the beginning and end of the first occurrence of each tag name and their frequency will be held in a companion array.

Functions

- readString: reads the character stream into the character array and return it
- findTagStart: searches the string from a given address until the first alpha character after ‘<’ (but not “</”, or “<!”) is found, returning its address
- findTagEnd: search the string from an address until the first non-alpha character is found, returning the last alpha-character address.
- addTag: checks for uniqueness, adds the tag to the list and updates the count
- reportTags: prints out the tag names found and the number of occurrences.

```
readString(char* html, int htmlMax);

char* findTagStart(char* html);

char* findTagEnd(char* html);

int addTag(char* start, char* end, char* tags[][2], int tagMax, int* nTags,
int tagCount);

void reportTags(char* tags[][2], int nTags, int* tagCount);
```

How I Built it

1. Getting Ready
    a. Built htags.h file
    b. Began htags.c file
    c. Began main.c (skeleton) file
    d. Wrote the Makefile
2. Wrote readString()
    a. Added readString() to main.c, printing out the string after it.
    b. Tested it
3. Wrote findTagStart()
    a. Added findTagStart () to main.c, printing out the character it returned a pointer to.
    b. Tested it
        i. Struggled with when the “<!” case. Wrote a scratch program to test ctype’s isalpha() to be sure I understood it. Realized that I thought I was messing up <!html> (which should be skipped), but it was really <html> and therefore was working correctly.
4. Wrote findTagEnd()
    a. Added findTagEnd () to main.c, printing out the character it returned a pointer to.
    b. Tested it

5. Created a loop in main.c to do this for the entire string
    a. Printed out the characters between the start and end pointers each time through the loop. This should give me only the opening tags.
        i. Realized that there's no string function to deal with a printing a string delineated by two pointers, so wrote one:

            ```
            int putnchar(char* start, int nchar);
            ```

        ii. Tested it in a scratch program
    b. Tested it.
        i. Ends with a segfault. Somewhere I'm accessing past the end of the string.
            1. Put printf() statements before each of my functions calls. Looks like it dies in findTagStart().
            2. I do A LOT of putting in printf statements. This does not help me.
            3. I remove the loop. All works for the first tag case.
            4. Finally realize that findTagStart has finished the string, but I haven't tested before calling findTagEnd, so the segfault is coming from there.
        ii. Put a check in for a NULL from findTagStart before calling findTagEnd.
        iii. Program runs, reporting the opening/"solo" tag names.
6. Counting tagname occurrences
    a. Added the tagList and tagCount structures to main()
    b. Added an init function for setting the values of tagCount: initInt() in htags
    c. In addTag
        i. added the code outline and looping structure
        ii. filled in the two sections
            1. add a tag to the list
            2. find a tag and increase count
        iii. got a subscript error on compile and realized I had an off-by-1 error.
        iv. Also realized the function doesn't need to report anything, set type to void
        v. Inserted printf() at various locations to check progress.
7. Reporting
    a. Simple implementation, but segfault.
    b. Seemed too simple to fail like that (accessing existing values), so went back to addTag.
        i. Printed each potential match from list against the tag to add and each time through the matching loop. Not a problem, but realized that I was breaking from the matching loop once a match had been found and not returning, so regardless of match the tag was being added to the array and maxing out the array.