Parsing Equations

LAB 10

SECTION C

Kenneth A. Jacobson

SUBMISSION DATE:

8.12.2017

Problem

The problem in this lab was to parse a string given by the user and find the variables and operators within the string.

Analysis

This lab had us taking in a string and then performing operations on it

Design

In this I had to design a series of loops which would parse the equation effectively and add the names of variables and operators to a string which is printed at the end of the programs run.

Testing

In order to test this program I fed it a variety of strings and saw what it parsed out.

Comments

N/A

Source Code

```
//Author: Kenneth A. Jacobson
//1.12.2017
//Parsing natural english expressions
//Lab #10

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void main(){
        //delimeters to parse string with
        const char delimeters[] = "*+-/()";

        //strings to hold values from input
        char input[100];        char parsnip[100];

        //pointers for the parsed token, holding values, and length of the token
        char *token;            char *length;
        char *hold;                     char *token2;

        //strings for output and to store length of variables
        char variables[50];     char operators[15];
        char token_length[30];

        //integers for various purposes
        //counter variables
        int i;                          int count= 0;

        //variables to store lenght values
        int len;                int len2;
        int len3;

        //error check variables
        int trundle= 0;         int spaces= 0;
        char *error;

        //Initialize strings with null values
        variables[0]= '\0';
        operators[0]= '\0';
        token_length[0]= '\0';

        //input a string for input which is coppied into parsnip for parsing
```

```
printf("Enter an expression: ");
        scanf("%s", input);
        strcpy(parsnip, input);

//allocate memory for the token, token2, length, error, and hold pointers
token= (char*) malloc(sizeof(char)*6);
token2= (char*) malloc(sizeof(char)*6);
length= (char*) malloc(sizeof(char)*10);
error= (char*) malloc(sizeof(char));
hold= (char*) malloc(sizeof(char));

//create first token from parsnip and place it in token
token = strtok(parsnip, delimeters);
while(token != NULL){
        //start adding commas for readability after first variable is passed in
        if(count > 0){
                strcat(variables, ", ");
        }

        //find out how long the variable is
        len = strlen(token);

        //error check for a variable which excceds the length allowed
        if(len > 6){
                printf("Expression invalid: Variable (%s) exceeds 6 characters", token);
                trundle = 22;
                break;
        }

        //error check for more than 3 spaces
        sprintf(error, "%c", input[len+1]);
        if(!(strcmp(error, " "))){
                sprintf(error, "%c", input[len+2]);
                if(!(strcmp(error, " "))){
                        sprintf(error, "%c", input[len+3]);
                        if(!(strcmp(error, " "))){
                                sprintf(error, "%c", input[len+4]);
                                if(!(strcmp(error, " "))){
                                        printf("Expression invalid: Excessive spaces");
                                        trundle = 22;
                                        break;
                                }
                        }
                }
```

```
        }

        //print length to a string for later use
        sprintf(length, "%d", len);
        strcat(token_length, length);

        //add the token to the variables string
        strcat(variables, token);
        count++;
        token = strtok(NULL, delimeters);

        //reinitialize spaces to 0
        spaces= 0;
}

for(i=0; i < count-1; i++){
        //add commas for readability after the first operator is passed in
        if(i > 0){
                strcat(operators, ", ");
        }

        //pass lenght of first variable into len
        len = token_length[i] - '0';
        //store running total to keep moving along the string by variable
        len2 += len;
        len3 = len2+(i*1);

        //print the character at index len3 to hold
        sprintf(hold, "%c", input[len3]);

        //error check for parentheses
        if(!strcmp(hold, "(")){
                printf("Expression invalid: Parentheses are not allowed");
                trundle = 22;
                break;
        }

        //add hold to the end of the operators string then reinitialize hold to NULL
        strcat(operators, hold);
        sprintf(hold, '\0');
}

if(count > 10){
        //error statement for variable count
```

```
                printf("Expression invalid: Variable count exceeds 10");
        }
        if(trundle != 22){
                printf("\nExpression is valid\n");
                printf("Variables: %s\n", variables);
                printf("Operators: %s", operators);
        }

}
```

Screen Shots