CptS 121 - Program Design and Development

Lab 3: C File Processing with Functions

Assigned: Week of February 3, 2014
Due: At the end of the lab session

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

- Open and close files
- Read, write to, and update files
- Manipulate file handles
- Apply standard library functions: fopen (), fclose (), fscanf (), and fprintf ()
- Compose decision statements ("if" conditional statements)
- Create and utilize compound conditions
- Implement and apply predicate functions
- Discover and distinguish between characters and how they are represented

II. Prerequisites:

Before starting this programming assignment, participants should be able to:

- Analyze a basic set of requirements for a problem and develop a representative structure chart
- Apply top-down design principles to a problem
- Utilize bottom-up C implementation for a problem
- Identify and implement programmer customized function prototypes, function definitions, and function calls
- Handle the 3 file format including: 1 header file, and 2 source files
- Distinguish between formal parameters and actual arguments
- Apply appropriate actual arguments to function calls as test inputs

III. Overview & Requirements:

This lab, along with your TA, will help you with C file processing. Once again we will take a modular approach to designing solutions to the problems below. We
solution.
Predicate functions return Boolean values false or true. In C we may return 0 for false and any non-zero value for true. Recall when processing files we follow
1) Open the files
2) Process the files
3) Close the files

Labs are held in a “closed” environment such that you may ask your TA questions. Please use your TAs knowledge to your advantage. You are required to move
students in need when you are finished with a task. You may work in pairs if you wish. However, I encourage you to compose your own solution to each problem
education in CptS 121 so work diligently.

Tasks:

Write a program that performs character processing on 10 characters read in from a file, and writes the results to output files. **Do NOT use loops or arrays to**
NOTE: You may **NOT** use the standard library functions found in <ctype.h>.

Your program should define the following functions:

- FILE * open_input_file (void) — Opens "input.dat" for reading.
- char read_character (FILE *infile) — Reads one character from the input file.
- int determine_ascii_value (char character) — Returns the ASCII value of the character passed into the function.
- int is_line (char character) — Determines if the character is a newline, if the character is a newline a 1 is returned otherwise a 0 is returned.
  Make sure that you #define two constants NEWLINE and NOT_NEWLINE as 1 and 0, respectively. Return the #defined constant.
- int number_lines (char character, int current_number_lines) — Determines if the character passed into the function indicates the end of a line (use i
  if so the function adds 1 to the current_number_lines and returns the value; otherwise it returns the current_number_lines without any modification.
- int is_vowel (char character) — Determines if the character is a vowel (note: the character may be lower or upper case), if the character is a vowel a
  Make sure that you #define two constants VOWEL and NOT_VOWEL as 2 and 0, respectively. Return the #defined constant.
- int number_vowels (char character, int current_number_vowels) — Determines if the character passed into the function is a vowel (use is_vowel ( )),
  if so the function adds 1 to the current_number_vowels and returns the value; otherwise it returns the current_number_vowels without any modifica
- int is_digit (char character) — Determines if the character is a digit (i.e. '0' - '9'), if the character is a digit a 3 is returned otherwise a 0 is returned.
  Make sure that you #define two constants DIGIT and NOT_DIGIT as 3 and 0, respectively. Return the #defined constant.
- int number_digits (char character, int current_number_digits) — Determines if the character passed into the function is a digit (use is_digit ( )),
  if so the function adds 1 to the current_number_digits and returns the value; otherwise it returns the current_number_digits without any modificatio
- int is_alpha (char character) — Determines if the character is an alpha character (i.e. 'a' - 'z', 'A' - 'Z'), if the character is an alpha character a 4 is retu

Make sure that you #define two constants ALPHA and NOT_ALPHA as 4 and 0, respectively. Return the #defined constant.

- int number_alphas (char character, int current_number_alphas) – Determines if the character passed into the function is an alpha character (use is_a
if so the function adds 1 to the current_number_alphas and returns the value; otherwise it returns the current_number_alphas without any modificat

- int is_lower (char character) - Determines if the character is a lowercase character, if the character is a lowercase character a 5 is returned otherwis
Make sure that you #define two constants LOWER and NOT_LOWER as 5 and 0, respectively. Return the #defined constant.
- int number_lowers (char character, int current_number_lowers) – Determines if the character passed into the function is a lowercase character (use
if so the function adds 1 to the current_number_lowers and returns the value; otherwise it returns the current_number_lowers without any modificat

- int is_upper (char character) - Determines if the character is an uppercase character, if the character is an uppercase character a 6 is returned other
Make sure that you #define two constants UPPER and NOT_UPPER as 6 and 0, respectively. Return the #defined constant.
- int number_uppers (char character, int current_number_uppers) – Determines if the character passed into the function is a uppercase character (use
if so the function adds 1 to the current_number_uppers and returns the value; otherwise it returns the current_number_uppers without any modifica

- int is_space (char character) - Determines if the character is a whitespace character (i.e. space ' ', form feed '\f', new-line '\n', carriage return '\r', ho
if the character is a whitespace character a 7 is returned otherwise a 0 is returned. Make sure that you #define two constants WHITESPACE and NOT
Return the #defined constant.
- int number_spaces (char character, int current_number_spaces) – Determines if the character passed into the function is a space character (use is_s
if so the function adds 1 to the current_number_spaces and returns the value; otherwise it returns the current_number_spaces without any modificat

- int is_alnum (char character) - Determines if the character is an alpha or digit character, if the character is an alpha or digit character a 8 is returnec
Make sure that you #define two constants ALNUM and NOT_ALNUM as 8 and 0, respectively. Return the #defined constant.
- int number_alnums (char character, int current_number_alnums) – Determines if the character passed into the function is an alphanumeric characte
if so the function adds 1 to the current_number_alnums and returns the value; otherwise it returns the current_number_alnums without any modifica

- int is_punct (char character) - Determines if the character is a punctuation character (i.e. '.', '!', ',', etc.) if the character is a punctuation character a
Make sure that you #define two constants PUNCT and NOT_PUNCT as 9 and 0, respectively. Return the #defined constant.
- int number_puncts (char character, int current_number_puncts) – Determines if the character passed into the function is a punctuation character (u
if so the function adds 1 to the current_number_puncts and returns the value; otherwise it returns the current_number_puncts without any modificat

- void print_int (FILE *outfile, int number) – Prints an integer to an output file.
- void print_stats (FILE *outfile, char header[ ], int number) – Prints a line like the following:

Number Vowels: 45

where "Number of vowels" is the string represented by the variable header and 45 is represented by number.

- A main function that does the following:
  - Opens an input file input.dat for reading;
  - Opens an output file output_stats.dat for writing all data generated by print_stats ( );
  - Opens an output file output_ascii.dat for writing all ascii values of each character;
  - Checks to see if the files were opened successfully
  - Reads one character at a time from the input file (input.dat), until all 10 characters have been read; For each character that is read in, its corresp
output_ascii.dat; Hint: use the print_int ( ) function to print the ASCII values;
  - Prints the number of lines in the file to output_stats.dat;
  - Prints the number of vowels in the file to output_stats.dat;
  - Prints the number of digits in the file to output_stats.dat;
  - Prints the number of alpha characters in the file to output_stats.dat;
  - Prints the number of lowercase characters in the file to output_stats.dat;
  - Prints the number of uppercase characters in the file to output_stats.dat;
  - Prints the number of space characters in the file to output_stats.dat;
  - Prints the number of alphanumeric characters in the file to output_stats.dat;
  - Prints the number of punctuation characters in the file to output_stats.dat;
  - Closes all opened files;

Sample Execution:

The following sample session demonstrates how your program should work.
Assuming input.dat stores the following characters:
CptS 121 is really fun!

Your program should write the following to output_ascii.dat:
67
112
116
83
32
49
50
49
32
105
115
32
114
101
97

108
108
121
32
102
117
110
33
10

Your program should write the following to output_stats.dat:
Number Lines: 1
Number Vowels: 4
Number Digits: 3
Number Alphas: 15
Number Lowers: 13
Number Uppers: 2
Number Spaces: 5 -- Including newline ('\n')
Number Alnums: 18
Number Puncts: 1

IV. Submitting Labs:

- You are not required to submit your lab solutions. However, you should keep them in a folder that you may continue to access throughout the semester. on the Sloan 353 machines. These files are erased on a daily basis.

V. Grading Guidelines:

- This lab is worth 10 points. Your lab grade is assigned based on completeness and effort. To receive full credit for the lab you must show up on time and dismissed you.