//

// main.c

// Chpt8PP1

//

// Created by Randy McMillan on 11/3/13.

// Copyright (c) 2013 Randy McMillan. All rights reserved.

//

/\*

\*

\* 1. Write and test a function deblank that takes a string output and a string input argument and returns a copy of the input argument with all blanks removed.

\*

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define LINE 80

#define ZERO 0

#define BLANK ' '

char input[LINE];

char output[LINE];

void myMain();

void promptInput();

void deBlank(char \*input, char \*output, int loopLength);

char \*scanLine(char \*dest, int dest\_len);

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

{

myMain();

return 0;

}

void myMain()

{

promptInput();

deBlank(scanLine(input, LINE), output, (int)strlen(input) + 1);

printf("%s", output);

}

void promptInput()

{

printf("Please enter a string to be deblanked --> ");

}

void deBlank(char \*input, char \*output, int loopLength)

{

// mind my p's and q's

int p = ZERO;

int q = ZERO;

while (p < loopLength) {

if (input[p] != BLANK) { // if not blank

output[q] = input[p];

q++; // increment output index by one when true

}

p++;// traverse array

}

}

// Based on Figure 8.15 in Book

char \*scanLine(char \*dest, int dest\_len)

{

int i, ch;

i = ZERO;

for (ch = getchar(); ch != '\n' && ch != EOF && i < dest\_len - 1; ch = getchar()) {

dest[i++] = ch;

}

dest[i] = '\0';

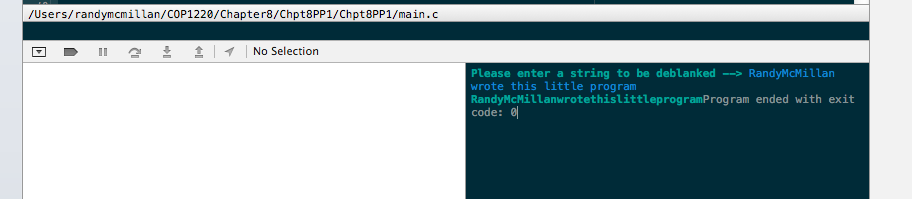
while (ch != '\n' && ch != EOF) {

ch = getchar();

}

return dest;

}



//

// main.c

// Chpt8PP5

//

// Created by Randy McMillan on 11/3/13.

// Copyright (c) 2013 Randy McMillan. All rights reserved.

//

/\*

\*

\*

\* 5. Write a program that takes nouns and forms their plurals on the basis of these rules:

\* a. If noun ends in “y”, remove the “y” and add “ies”.

\* b. If noun ends in “s”, “ch”, or “sh”, add “es”.

\* c. In all other cases, just add “s”.

\* Print each noun and its plural. Try the following data:

\* chair dairy boss circus fly dog church clue dish

\*

\*

\*/

#include <stdio.h>

#define ZERO 0

#define LINE 80

void myMain();

void promptInput();

char \*scanLine(char \*dest, int dest\_len);

char string[LINE];

char modified[LINE];

char pluralize(char string[LINE]);

char \*reverse\_string(char \*str);

#define Y "y"

#define S "s"

// reversed for logic in pluralize()

#define H "h"

#define C "c"

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

{

myMain();

return 0;

}

void myMain()

{

promptInput();

scanLine(string, LINE);

strncpy(modified, string, (size\_t)strlen(string));

// strncpy(modified, modified, 1);

printf("%c", pluralize(modified));

}

void promptInput()

{

printf("Please enter a NOUN to be pluralized --> ");

}

// Based on Figure 8.15 in Book

char \*scanLine(char \*dest, int dest\_len)

{

int i, ch;

i = ZERO;

for (ch = getchar(); ch != '\n' && ch != EOF && i < dest\_len - 1; ch = getchar()) {

dest[i++] = ch;

}

dest[i] = '\0';

while (ch != '\n' && ch != EOF) {

ch = getchar();

}

return dest;

}

char pluralize(char string[LINE])

{

char plural = ' ';

int done = 0;

while (done != 1) {

// printf("before - %s",string);

if (!strncmp(reverse\_string(string), "s", 1)) {

reverse\_string(string);

// string[strlen(string)-1] = 0;

strcat(string, "es");

printf("%s\n", string);

done = 1;

}

reverse\_string(string);

if (!strncmp(reverse\_string(string), "y", 1)) {

reverse\_string(string);

string[strlen(string) - 1] = 0;

strcat(string, "ies");

printf("%s\n", string);

done = 1;

}

reverse\_string(string);

// printf("before - %s",string);

if (!strncmp(reverse\_string(string), "hc", 2)) {

reverse\_string(string);

// string[strlen(string)-1] = 0;

strcat(string, "es");

printf("%s\n", string);

done = 1;

}

reverse\_string(string);

// printf("before - %s",string);

if (!strncmp(reverse\_string(string), "hs", 2)) {

reverse\_string(string);

// string[strlen(string)-1] = 0;

strcat(string, "es");

printf("%s\n", string);

done = 1;

}

// reverse\_string(string);

// printf("before - %s",string);

if (strncmp(reverse\_string(string), "s", 1)) {

//reverse\_string(string);

//string[strlen(string)-1] = 0;

strcat(string, "s");

printf("%s\n", string);

done = 1;

}

}

return plural;

}

///common reverse string technique

char \*reverse\_string(char \*str)

{

char temp;

size\_t len = strlen(str) - 1;

size\_t i;

size\_t k = len;

for (i = 0; i < len; i++) {

temp = str[k];

str[k] = str[i];

str[i] = temp;

k--;

if (k == (len / 2)) {

break;

}

}

return str;

}

