Programming Fundamental – ENSF 337

Lab 3

M. Moussavi

Jay Chuang

B01

September 28, 2019

/\*

\* lab3exe\_C.c

\* ENSF 337, lab3 Exercise C

\*

\* Completed by: Jay Chuang

\* Lab Section: B01

\*/

#include <stdio.h>

#include <stdlib.h>

void pascal\_triangle(int n);

/\* REQUIRES: n > 0 and n <= 20

PROMISES: displays a pascal\_triangle. the first 5 line of the function's output

should have the following format:

row 0: 1

row 1: 1 1

row 2: 1 2 1

row 3: 1 3 3 1

row 4: 1 4 6 4 1

\*/

int main() {

int nrow;

// These are ALL of the variables you need!

printf("Enter the number of rows (Max 20): \n");

scanf("%d", &nrow);

if(nrow <= 0 || nrow > 20) {

printf("Error: the maximum number of rows can be 20.\n");

exit(1);

}

pascal\_triangle(nrow);

return 0;

}

void pascal\_triangle(int n) {

int array[n][n];

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

{

for (int j = 0; j < (i+1); j++)

{

if (j == 0 || j == i)

{

array[i][j] = 1;

}

else

{

array[i][j] = array[i-1][j-1] + array[i-1][j];

}

printf ("%d ", array[i][j]);

}

printf ("\n");

}

}

**EXERCISE C OUTPUT**

Enter the number of rows (Max 20):

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

/\* lab3exe\_D.c

\* ENSF 337, Lab 3 Exercise D

\* Completed by: Jay Chuang

\* Lab Section: B01

\*/

#include <stdio.h>

#include <string.h>

int substring(const char \*s1, const char \*s2);

/\* REQUIRES

\* s1 and s2 are valid C-string terminated with '\0';

\* PROMISES

\* returns one if s2 is a substring of s1). Otherwise returns zero.

\*/

void select\_negatives(const int \*source, int n\_source,

int\* negatives\_only, int\* number\_of\_negatives);

/\* REQUIRES

\* n\_source >= 0.

\* Elements source[0], source[1], ..., source[n\_source - 1] exist.

\* Elements negatives\_only[0], negatives\_only[1], ..., negatives\_only[n\_source - 1] exist.

\* PROMISES

\* number\_of\_negatives == number of negative values in source[0], ..., source[n\_source - 1].

\* negatives\_only[0], ..., negatives\_only[number\_of\_negatives - 1] contain those negative values, in

\* the same order as in the source array. \*/

int main(void)

{

char s[] = "Knock knock! Who's there?";

int a[] = { -1279, 1894, -1047, 0, -103 };

int size\_a;

int i;

int negative[5];

int n\_negative;

size\_a = sizeof(a) / sizeof(a[0]);

printf("a has %d elements:", size\_a);

for (i = 0; i < size\_a; i++)

printf(" %d", a[i]);

printf("\n");

select\_negatives(a, size\_a, negative, &n\_negative);

printf("\nnegative elements from array a are as follows:");

for (i = 0; i < n\_negative; i++)

printf(" %d", negative[i]);

printf("\n");

printf("\nNow testing substring function....\n");

printf("Answer must be 1.substring function returned: %d\n" , substring(s, "Who"));

printf("Answer must be 0.substring function returned: %d\n" , substring(s, "knowk"));

printf("Answer must be 1.substring function returned: %d\n" , substring(s, "knock"));

printf("Answer must be 0.substring function returned: %d\n" , substring(s, ""));

printf("Answer must be 1.substring function returned: %d\n" , substring(s, "ck! Who's"));

printf("Answer must be 0.substring function returned: %d\n" , substring(s, "ck!Who's"));

return 0;

}

int substring(const char \*s1, const char\* s2)

{

int sizes1, sizes2;

for(sizes1 = 0; s1[sizes1] != '\0'; ++sizes1);

for(sizes2 = 0; s2[sizes2] != '\0'; ++sizes2);

for(int i = 0; i < sizes1; i++) {

int j = 0;

if(s1[i] == s2[j]) {

int temp = i;

while ((s1[i] == s2[j]) && (j < sizes2)) {

i++;

j++;

}

if (s2[j] == '\0')

return 1;

else

i = temp;

}

}

return 0;

}

void select\_negatives(const int \*source, int n\_source,

int\* negatives\_only, int\* number\_of\_negatives)

{

int position = 0;

\*number\_of\_negatives = 0;

for (int i = 0; i < n\_source; i++)

{

if (source[i] < 0)

{

negatives\_only[position] = source[i];

position+=1;

\*number\_of\_negatives+=1;

}

}

return;

}

**EXERCISE D OUTPUT**

a has 5 elements: -1279 1894 -1047 0 -103

negative elements from array a are as follows: -1279 -1047 -103

Now testing substring function....

Answer must be 1.substring function returned: 1

Answer must be 0.substring function returned: 0

Answer must be 1.substring function returned: 1

Answer must be 0.substring function returned: 0

Answer must be 1.substring function returned: 1

Answer must be 0.substring function returned: 0

/\* File: palindrome.c

\* ENSF 337

\* Exercise E - Lab 3

\* Completed by: Jay Chuang

\* Lab Section: B01

\*/

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define SIZE 100

/\* function prototypes\*/

int is\_palindrome (const char \*str);

/\* REQUIRES: str is pointer to a valid C string.

\* PROMISES: the return value is 1 if the string a is palindrome.\*/

void strip\_out(char \*str);

/\* REQUIRES: str points to a valid C string terminated with '\0'.

\* PROMISES: strips out any non-alphanumerical characters in str\*/

int main(void)

{

int p =0;

char str[SIZE], temp[SIZE];

fgets(str, SIZE, stdin);

/\* Remove end-of-line character if there is one in str.\*/

if (str[strlen(str) - 1] == '\n')

str[strlen(str) - 1] = '\0';

strcpy(temp,str);

/\* This loop is infinite if the string "done" never appears in the

\* input. That's a bit dangerous, but OK in a test harness where

\* the programmer is controlling the input. \*/

while(strcmp(str, "done") !=0) /\* Keep looping unless str matches "done". \*/

{

#if 1

strip\_out(str);

p = is\_palindrome(str);

#endif

if(!p)

printf("\n \"%s\" is not a palindrome.", temp);

else

printf("\n \"%s\" is a palindrome.", temp);

fgets(str, SIZE, stdin);

/\* Remove end-of-line character if there is one in str.\*/

if(str[strlen(str) - 1] == '\n')

str[strlen(str) - 1]= '\0';

strcpy(temp, str);

}

return 0;

}

void strip\_out(char \*str)

{

int i;

int index = 0;

for(i = 0; str[i]; i++)

{

if(isupper(str[i]) == 1) //if str[i] is capital

{

str[i] = tolower(str[i]); //make into lowercase

}

if(isalnum(str[i]) != 0) //if str[i] is not alphanumeric

{

str[index++] = str[i]; //move array

}

if(isupper(str[i]) == 1) //if str[i] is capital

{

str[i] = tolower(str[i]); //make into lowercase

}

}

str[index] = '\0';

}

int is\_palindrome (const char \*str)

{

int length = strlen(str);

int half;

if(length % 2 == 1)

{

half = (length-1)/2;

}

else

{

half = (length/2);

}

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

{

if(str[i] != str[length-1-i])

return 0;

}

return 1;

}

**EXERCISE E OUTPUT**

"Radar" is a palindrome.

"Madam I'm Adam" is a palindrome.

"Alfalfa" is not a palindrome.

"He maps spam, eh?" is a palindrome.

"I did, did I?" is a palindrome.

" I prefer pi." is a palindrome.

"Ed is on no side" is a palindrome.

"Am I loco, Lima?" is a palindrome.

" Bar crab." is a palindrome.

"A war at Tarawa." is a palindrome.

"Ah, Satan sees Natasha" is a palindrome.

" Borrow or rob?" is a palindrome.

"233332" is a palindrome.

"324556" is not a palindrome.

"Hello world!!" is not a palindrome.

" Avon sees nova " is a palindrome.

"Can I attain a 'C'?" is a palindrome.

"Sept 29, 2005." is not a palindrome.

"Delia failed." is a palindrome.

"Draw nine men $$ inward" is a palindrome.