Michael Campbell

X52.9008 – 01

Tues 6:30-9:00PM

June 15, 2010

HW # 4 --Chapter 7 # 4, 5, 6

Chapter 10 # 3, 6

7.4) Write a program that calculates the average of an array of 10 floating-point values.

#include<stdio.h>

int main (void)

{

int i = 0;

float sum = 0;

float average = 0;

float values[10] = {0};

for (i = 1; i < 11; i++){

printf ("Input first Value\n");

scanf ("%f", &values[i]);

sum += values[i];

}

average = sum / 10;

printf("The average is %.3f", average);

return 0;

}

7.5) What output would you expect form the following program?

#include <stdio.h>

int main (void)

{

int numbers[10] = {1, 0, 0, 0, 0, 0, 0, 0, 0, 0};

int i, j;

for ( j = 0; j < 10; ++j )

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

numbers[j] += numbers[i];

for ( j = 0; j < 10; ++j )

printf (“%i ”, numbers[j]);

printf (“\n”);

return 0;

}

Output: 1 1 2 4 8 16 32 64 128 256

7.6) You don’t need to use an array to generate Fibonacci numbers. You can simply use three variables: two to store the previous Fibonacci numbers and one to store the current one. Rewrite Program 7.3 so that arrays are not used. Because you’re no longer using an array, you need to display each Fibonacci number as you generate it.

#include<stdio.h>

int main (void)

{

int prior = 0;

int last = 1;

int current = 0;

int i;

printf ("%i %i ", prior, last);

for (i = 1; i < 16; i++){

current = prior + last;

prior = last;

printf ("%i ", current);

last = current;

}

return 0;

}

10.3) The countWords function from Programs 10.7 and 10.8 incorrectly counts a word that contains an apostrophe as two separate words. Modify this function to correctly handle this situation. Also, extend the function to count a sequence of positive or negative numbers, including any embedded commas and periods, as a single word.

//Function to determine if a character is alphabetic

#include <stdio.h>

#include <stdbool.h>

bool alphabetic (const char c)

{

if ( (c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c == '\'') || (c == ',') || (c == '.') || (c == '-') || (c >= '0' && c <='9'))

return true;

else

return false;

}

/\* Function to count the number of words in a string \*/

int countWords (const char string[])

{

int i, wordCount = 0;

bool lookingForWord = true, alphabetic (const char c);

for ( i = 0; string [i] != '\0'; ++i )

if (alphabetic(string[i]))

{

if ( lookingForWord )

{

++wordCount;

lookingForWord = false;

}

}

else

lookingForWord = true;

return wordCount;

}

int main (void)

{

const char text1[] = "Well, here goes.";

const char text2[] = "And here we go... again.";

const char text3[] = "We're already done...";

const char number1[] = "123,456";

const char number2[] = "-12,345";

int countWords (const char string[]);

printf ("%s - words = %i\n", text1, countWords (text1));

printf ("%s - words = %i\n", text2, countWords (text2));

printf ("%s - words = %i\n", text3, countWords (text3));

printf ("%s - words = %i\n", number1, countWords (number1));

printf ("%s - words = %i\n", number2, countWords (number2));

return 0;

}

Output:

Well, here goes. - words = 3

And here we go... again. - words = 5

We're already done... - words = 3

123,456 - words = 1

-12,345 - words = 1

10.6) Write a function called removeString to remove a specified number of characters from a character sting. The function should take three arguments: the source string, the starting index number in the source string, and the number of characters to remove. So, if the character array text contains the string “the wrong song”, the call

removeString (text, 4, 6);

has the effect of removing the characters “wrong” (the word “wrong” plus the space that follows) from the array text. The resulting string inside text is then “the song”.

#include <stdio.h>

int main (void)

{

char removeString (char text[], int start, int length);

char text[] = "the wrong song";

printf("%c\n", removeString(text, 4, 6));

return 0;

}

char removeString (char text[], int start, int length)

{

int i = 0, j = 0;

int total = start + length;

char result[100];

for (i = 0; i < start; i++)

result[i] = text[i];

for (j = 0; text[j] != '\0'; ++j)

result[i + j] = text[j + total];

result[i + j] = '\0';

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

}

Output:

the song