|  |
| --- |
| **Course: Programming Fundamental – ENSF 337**  Lab #: 4  Instructor: Khedr  Student Name: Aleksander Berezowski  Lab Section: B04  Date submitted: October 20th |

**Exercise A**

Diagram

Description automatically generated

**Exercise B**

Diagram

Description automatically generated

**Exercise C**

*/\*  
\* File Name: lab4exe\_C.c  
\* Assignment: Lab 4 Exercise C  
\* Lab section: B04  
\* Completed by: Aleksander Berezowski  
\* Submission Date: On or before Oct 21, 2021  
\*/*#include <stdio.h>  
#define **ELEMENTS**(a) *sizeof*(a)/*sizeof*(a[0])  
  
  
*int* main()  
{  
   
 *int* size;  
 *int* a[] = {45, 67, 89, 24, 54};  
 *double* b[20] = {14.5, 61.7, 18.9, 2.4, 0.54};  
   
 size = **ELEMENTS**(a);  
  
  
 printf("Array a has 5 elements and macro ELEMENTS returns %d\n", size);  
   
 size = **ELEMENTS**(b);  
   
   
 printf("Array b has 20 elements and macro ELEMENTS returns %d\n", size);  
   
 *return* 0;  
}

Text

Description automatically generated

**Exercise D**

*/\*  
\* File Name: lab4exD.c  
\* Assignment: Lab 4 Exercise D  
\* Lab section: B04  
\* Completed by: Aleksander Berezowski  
\* Submission Date: On or before Oct 21, 2021  
\*/*#include <stdio.h>  
#include <string.h>  
  
*int* my\_strlen(*const char* \**s*);  
*/\* Duplicates strlen from <string.h>, except return type is int.  
 \* REQUIRES  
 \* s points to the beginning of a string.  
 \* PROMISES  
 \* Returns the number of chars in the string, not including the  
 \* terminating null.  
 \*/  
  
void* my\_strncat(*char* \**dest*, *const char* \**source*, *int*);  
*/\* Duplicates strncat from <string.h>, except return type is void.  
 \* dest and source point to the beginning of two strings.  
 \* PROMISES  
 \* appends source to the end of dest. If length of source is more than n.  
 \* Only copies the first n elements of source.  
 \*/  
  
int* my\_strncmp(*const char*\* *str1*, *const char*\* *str2*);  
*/\* Duplicates strcmp from <string.h>, except return type is int.  
 \* REQUIRES  
 \* str1 points to the beginning of a string, and str2 to the beginning of  
 \* another string.  
 \* PROMISES  
 \* Returns 0 if str1 and str2 are idntical.  
 \* Returns a negative number of str1 is less that str2.  
 \* Return a psitive nubmer of str2 is less than str1.  
 \*/  
  
int* main(*void*)  
{  
 *char* str1[7] = "banana";  
 *const char* str2[] = "-tacit";  
 *const char*\* str3 = "-toe";  
   
 *char* str5[] = "ticket";  
 *char* my\_string[100]="";  
 *int* bytes;  
 *int* length;  
 *int* y;  
   
 printf("\nTESTING strlen FUNCTION ... \n");  
   
 */\* using strlen function \*/* length = (*int*) my\_strlen(my\_string);  
 printf("\nExpected to display: my\_string length is 0.");  
 printf("\nmy\_string length is %d.", length);  
   
 */\* using sizeof operator \*/* bytes = *sizeof* (my\_string);  
 printf("\nExpected to display: my\_string size is 100 bytes.");  
 printf("\nmy\_string size is %d bytes.", bytes);  
   
 */\* using strcpy C libarary function \*/* strcpy(my\_string, str1);  
 printf("\nExpected to display: my\_string contains banana.");  
 printf("\nmy\_string contains %s", my\_string);  
   
 length = (*int*) my\_strlen(my\_string);  
 printf("\nExpected to display: my\_string length is 6.");  
 printf("\nmy\_string length is %d.", length);  
   
 my\_string[0] = '\0';  
 printf("\nExpected to display: my\_string contains \"\".");  
 printf("\nmy\_string contains:\"%s\"", my\_string);  
   
 length = (*int*) my\_strlen(my\_string);  
 printf("\nExpected to display: my\_string length is 0.");  
 printf("\nmy\_string length is %d.", length);  
   
 bytes = *sizeof* (my\_string);  
 printf("\nExpected to display: my\_string size is still 100 bytes.");  
 printf("\nmy\_string size is still %d bytes.", bytes);  
   
 printf("\n\nTESTING strncat FUNCTION ... \n");  
 */\* strncat append the first 3 characters of str5 to the end of my\_string \*/* my\_strncat(my\_string, str5, 3);  
 printf("\nExpected to display: my\_string contains \"tic\"");  
 printf("\nmy\_string contains \"%s\"", my\_string);  
   
 length = (*int*) my\_strlen(my\_string);  
 printf("\nExpected to display: my\_string length is 3.");  
 printf("\nmy\_string length is %d.", length);  
  
 my\_strncat(my\_string, str2, 4);  
 printf("\nExpected to display: my\_string contains \"tic-tac\"");  
 printf("\nmy\_string contains:\"%s\"", my\_string);  
   
 */\* strncat append ONLY up ot '\0' character from str3 -- not 6 characters \*/* my\_strncat(my\_string, str3, 6);  
 printf("\nExpected to display: my\_string contains \"tic-tac-toe\"");  
 printf("\nmy\_string contains:\"%s\"", my\_string);  
   
 length = (*int*) my\_strlen(my\_string);  
 printf("\nExpected to display: my\_string has 11 characters.");  
 printf("\nmy\_string has %d characters.", length);  
   
 printf("\n\nUsing strcmp - C library function: ");  
 printf("\nExpected to display: \"ABCD\" is less than \"ABCDE\"");  
 printf("\n\"ABCD\" is less than \"ABCDE\"", strcmp("ABCD", "ABCDE"));  
   
   
 printf("\n\nTESTING strcmp FUNCTION ... \n");  
   
 *if*((y = my\_strncmp("ABCD", "ABND")) < 0)  
 printf("\n\"ABCD\" is less than \"ABND\" ... strcmp returns %d", y);  
  
 *if*((y = my\_strncmp("ABCD", "ABCD")) == 0)  
 printf("\n\"ABCD\" is equal \"ABCD\" ... strcmp returns %d", y);  
   
 *if*((y = my\_strncmp("ABCD", "ABCd")) < 0)  
 printf("\n\"ABCD\" is less than \"ABCd\" ... strcmp returns %d", y);  
   
 *if*((y = my\_strncmp("Orange", "Apple")) > 0)  
 printf("\n\"Orange\" is greater than \"Apple\" ... strcmp returns %d\n", y);  
   
 *return* 0;  
}  
  
*int* my\_strlen(*const char* \**s*){  
 *int* counter = 0;  
 *//While the character at s is a thing, keep increasing the counter  
 while*(\*(*s*+counter))  
 counter++;  
 *//Above breaks after hitting last character  
 return* counter;  
}  
  
*void* my\_strncat(*char* \**dest*, *const char* \**source*, *int n*) {  
 *int* i, j;  
  
 *//make i point to the last space (the null thingy)  
 for* (i = 0; \*(*dest*+i) != '\0'; i++);  
  
 *//keep appending to the end of dest as long as j is smaller than n  
 for* (j = 0; \*(*source*+j) != '\0' && j < *n*; j++) {  
 \*(*dest* +i + j) = \*(*source* + j);  
 }  
  
 *//set the last letter to the null characters* \*(*dest* +i + j) = '\0';  
}  
  
*int* my\_strncmp(*const char*\* *str1*, *const char*\* *str2*) {  
  
 *int* i = 0;  
  
 *while*(\*(*str1*+i) || \*(*str2* + i)){  
 *if*( (*int*)\*(*str1*+i) > (*int*)\*(*str2*+i) )  
 *return* 1;  
 *else if* ( (*int*)\*(*str1*+i) < (*int*)\*(*str2*+i) )  
 *return* -1;  
  
 i++;  
 }  
 *return* 0;  
  
}

Text

Description automatically generatedText

Description automatically generated

**Exercise E**

Text

Description automatically generated