**Course: ENSF614 – Fall 2023**

**Lab #2**

**Instructor: M. Moussavi**

**Student Name : Sieu Eric Diep & Cory Yang-Smith**

**Submission Date : Sep 27, 2023**

**Exercise A – Memory diagrams lab2exe\_A.cpp**

**Point one:**

**Point two: including labels to indicate the size in byes for each variable, array and function argument**

**Point three:**

**Point four:**

**Exercise B:**

/\*

\* lab2exe\_B.cpp

\* ENSF 614 Lab 2 Exercise B

\* Completed by Sieu Eric Diep

\*/

int my\_strlen(const char \*s);

/\* Duplicates strlen from <cstring>, 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 <cstring>, except return type is void.

\* REQUIRES:

\* dest points to the beginning of a string,

\* source points to the beginning of a string

\* an integer n which is the first n character of the source string

\* PROMISES

\* appending the first n character of string source to dest

\*/

int my\_strcmp(const char\* str1, const char\* str2);

/\* Compares two strings and return an integer

\* REQUIRES:

\* two cstrings, i.e. two char pointers, each points to a cstring

\* PROMISES:

\* return 0 if two string are identical

\* return a negative value if string 1 is less than string 2

\* return a positive value if string 1 is greater than string 2

\*/

#include <iostream>

#include <cstring>

using namespace std;

int main(void)

{

char str1[7] = "banana";

const char str2[] = "-tacit";

const char\* str3 = "-toe";

/\* point 1 \*/

char str5[] = "ticket";

char my\_string[100]="";

int bytes;

int length;

/\* using strlen libarary function \*/

length = (int) my\_strlen(my\_string);

cout << "\nLine 1: my\_string length is " << length;

/\* using sizeof operator \*/

bytes = sizeof (my\_string);

cout << "\nLine 2: my\_string size is " << bytes << " bytes.";

/\* using strcpy libarary function \*/

strcpy(my\_string, str1);

cout << "\nLine 3: my\_string contains: " << my\_string;

length = (int) my\_strlen(my\_string);

cout << "\nLine 4: my\_string length is " << length << ".";

my\_string[0] = '\0';

cout << "\nLine 5: my\_string contains:\"" << my\_string << "\"";

length = (int) my\_strlen(my\_string);

cout << "\nLine 6: my\_string length is " << length << ".";

bytes = sizeof (my\_string);

cout << "\nLine 7: my\_string size is still " << bytes << " bytes.";

/\* strncat append the first 3 characters of str5 to the end of my\_string \*/

my\_strncat(my\_string, str5, 3);

cout << "\nLine 8: my\_string contains:\"" << my\_string << "\"";

length = (int) my\_strlen(my\_string);

cout << "\nLine 9: my\_string length is " << length << ".";

my\_strncat(my\_string, str2, 4);

cout << "\nLine 10: my\_string contains:\"" << my\_string << "\"";

/\* strncat append ONLY up ot '\0' character from str3 -- not 6 characters \*/

my\_strncat(my\_string, str3, 6);

cout << "\nLine 11: my\_string contains:\"" << my\_string << "\"";

length = (int) my\_strlen(my\_string);

cout << "\nLine 12; my\_string has " << length << " characters.";

cout << "\n\nUsing strcmp - C library function: ";

cout << "\n\"ABCD\" is less than \"ABCDE\" ... strcmp returns: " <<

my\_strcmp("ABCD", "ABCDE");

cout << "\n\"ABCD\" is less than \"ABND\" ... strcmp returns: " <<

my\_strcmp("ABCD", "ABND");

cout << "\n\"ABCD\" is equal than \"ABCD\" ... strcmp returns: " <<

my\_strcmp("ABCD", "ABCD");

cout << "\n\"ABCD\" is less than \"ABCd\" ... strcmp returns: " <<

my\_strcmp("ABCD", "ABCd");

cout << "\n\"Orange\" is greater than \"Apple\" ... strcmp returns: " <<

my\_strcmp("Orange", "Apple") << endl;

return 0;

}

int my\_strlen(const char \*s){

int count = 0;

while(s[count] != '\0')

count++;

return count;

}

void my\_strncat(char \*dest, const char \*source, int n){

int pos = 0, i=0;

while(dest[pos] !='\0'){

pos++;

}

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

dest[pos+i] = source[i];

}

dest[pos+i] = '\0';

}

int my\_strcmp(const char\* str1, const char\* str2){

int pos = 0, result = 0;

do{

result = str1[pos] - str2[pos];

if(result !=0 )

return result;

else

pos++;

}while(str1[pos] != '\0' || str2[pos] != '\0');

return result;

}

**Program Output:**

A screenshot of a computer program

Description automatically generated

**Exercise C: AR Diagram**

**Exercise E:**

**cplx cplx\_add(cplx z1, cplx z2)**

**{**

**cplx result;**

**result.real = z1.real + z2.real;**

**result.imag = z1.imag + z2.imag;**

**return result;**

**}**

**void cplx\_subtract(cplx z1, cplx z2, cplx \*difference)**

**{**

**difference->real = z1.real - z2.real;**

**difference->imag = z1.imag - z2.imag;**

**}**

**void cplx\_multiply(const cplx \*pz1,const cplx \*pz2,cplx \*product)**

**{**

**product->real = (pz1->real)\*(pz2->real) - (pz1->imag)\*(pz2->imag);**

**product->imag = (pz1->real)\*(pz2->imag) + (pz2->real)\*(pz1->imag);**

**}**