**Course: ENSF 694** – Summer 2024

**Lab #:** Lab 2

**Instructor:** Mahmood Moussavi

**Student Name:** Ryan Baker

**Submission Date:** July 10, 2024

# Part I Exercise A

## my\_lab2exe\_A.cpp

/\*

 \*  my\_lab2exe\_A.cpp

 \*  ENSF 694 Lab 2 Part I Exercise A

 \*  Completed by: Ryan Baker

 \*  Development Date: July 4, 2024

 \*/

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.

 \*/

#include <iostream>

#include <cstring>

using namespace std;

const int MAX\_LENGTH = 100;

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: " <<

    strcmp("ABCD", "ABCDE");

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

    strcmp("ABCD", "ABND");

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

    strcmp("ABCD", "ABCD");

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

    strcmp("ABCD", "ABCd");

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

    strcmp("Orange", "Apple") << endl;

    return 0;

}

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

    int count = 0;

    // include a maximum length to prevent infinite loop

    for(int i = 0; i < MAX\_LENGTH; i++){

        if(\*(s + i) == 0){

            return count;

        } else {

            count++;

        }

    }

    return 0;

}

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

    // find null character

    // used MAX\_LENGTH to prevent infinite loop

    for(int i = 0; i < MAX\_LENGTH; i++){

        if(dest[i] == 0){

            // append characters to end of dest

            for(int j = 0; j < num; j++){

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

            }

            // ensure new string is null terminated

            dest[i + num] = '\0';

            // stop loop

            break;

        }

    }

}

## Program Output

Line 1: my\_string length is 0

Line 2: my\_string size is 100 bytes.

Line 3: my\_string contains: banana

Line 4: my\_string length is 6.

Line 5: my\_string contains:""

Line 6: my\_string length is 0.

Line 7: my\_string size is still 100 bytes.

Line 8: my\_string contains:"tic"

Line 9: my\_string length is 3.

Line 10: my\_string contains:"tic-tac"

Line 11: my\_string contains:"tic-tac-toe"

Line 12; my\_string has 11 characters.

Using strcmp - C library function:

"ABCD" is less than "ABCDE" ... strcmp returns: -1

"ABCD" is less than "ABND" ... strcmp returns: -1

"ABCD" is equal than "ABCD" ... strcmp returns: 0

"ABCD" is less than "ABCd" ... strcmp returns: -1

"Orange" is greater than "Apple" ... strcmp returns: 1

# Part I Exercise B

## lab2exe\_B.cpp

/\*

 \*  lab2exe\_B.cpp

 \*  ENSF 694 Lab 2 Part I Exercise B

 \*  Completed by: Ryan Baker

 \*  Development Date: July 4, 2024

 \*/

#include <iostream>

#include <assert.h>

using namespace std;

int sum\_of\_array(const int \*a, int n);

// REQUIRES

//   n > 0, and elements a[0] ... a[n-1] exist.

// PROMISES:

//  Return value is a[0] + a[1] + ... + a[n-1].

int main()

{

    int a[] = { 100 };

    int b[] = { 100, 200, 300, 400 };

    int c[] = { -100, -200, -200, -300 };

    int d[] = { 10, 20, 30, 40, 50, 60, 70 };

    int sum = sum\_of\_array(a, 1);

    cout << "sum of integers in array a is: " << sum << endl;

    sum = sum\_of\_array(b, 4);

    cout << "sum of integers in array b is: " << sum << endl;

    sum = sum\_of\_array(c, 4);

    cout << "sum of integers in array c is: " << sum << endl;

    sum = sum\_of\_array(d, 7);

    cout << "sum of integers in array d is: " << sum << endl;

    return 0;

}

int sum\_of\_array(const int \*a, int n)

{

    int sum = 0;

    // base case

    if(n == 1){

        return a[n-1];

    }

    sum += a[n-1] + sum\_of\_array(a, n-1);

    return sum;

}

## Program output

sum of integers in array a is: 100

sum of integers in array b is: 1000

sum of integers in array c is: -800

sum of integers in array d is: 280

# Part I Exercise C

* No submission required

# Part I Exercise D