ENSF 614 – Fall 2021

Lab 2 – Tuesday, September 28

Student Name: Bhavyai Gupta

Submission date: September 28, 2021

# Exercise A – AR Diagram for Point One

Diagram

Description automatically generated with medium confidence

# Exercise A – AR Diagram for Point Two

Diagram, engineering drawing

Description automatically generated

# Exercise A – AR Diagram for Point Three

Diagram

Description automatically generated

# Exercise A – AR Diagram for Point Four

Diagram

Description automatically generated

# Exercise B – Source File

/\*

 \* File Name:               my\_lab2exe\_B.c

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 2 Exercise B

 \* Lab section:             B01

 \* Completed by:            Bhavyai Gupta

 \* Submission Date:         September 28, 2021

 \*/

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 n);

/\*

 \* Duplicates strncat from <string.h>, except return type is void.

 \*

 \* REQUIRES

 \*     dest and source pointing to the beginning of the string

 \*     n is greater than or equal to 0

 \*     sizeof(dest) > len(dest) + n

 \*

 \* PROMISES

 \*     Copies first n characters from source to dest. If n is

 \*     smaller than length of string, then characters are copied

 \*     from source till '\0' is encountered.

 \*/

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

/\*

 \* Duplicates strncat from <string.h>, except return type is void.

 \*

 \* REQUIRES

 \*     dest and source pointing to the beginning of the string

 \*

 \* PROMISES

 \*     Returns 0 is two strings are exactly equal, otherwise difference

 \*     of ASCII values of the first two characters that are different.

 \*/

#include <stdio.h>

#include <string.h>

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 C library function \*/

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

    printf("\nLine 1: my\_string length is %d.", length);

    /\* using sizeof operator \*/

    bytes = sizeof(my\_string);

    printf("\nLine 2: my\_string size is %d bytes.", bytes);

    /\* using strcpy C libarary function \*/

    strcpy(my\_string, str1);

    printf("\nLine 3: my\_string contains: %s", my\_string);

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

    printf("\nLine 4: my\_string length is %d.", length);

    my\_string[0] = '\0';

    printf("\nLine 5: my\_string contains:\"%s\"", my\_string);

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

    printf("\nLine 6: my\_string length is %d.", length);

    bytes = sizeof(my\_string);

    printf("\nLine 7: my\_string size is still %d bytes.", bytes);

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

    my\_strncat(my\_string, str5, 3);

    printf("\nLine 8: my\_string contains:\"%s\"", my\_string);

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

    printf("\nLine 9: my\_string length is %d.", length);

    my\_strncat(my\_string, str2, 4);

    printf("\nLine 10: my\_string contains:\"%s\"", my\_string);

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

    my\_strncat(my\_string, str3, 6);

    printf("\nLine 11: my\_string contains:\"%s\"", my\_string);

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

    printf("\nLine 12; my\_string has %d characters.", length);

    printf("\n\nUsing strcmp - C library function: ");

    printf("\n\"ABCD\" is less than \"ABCDE\" ... strcmp returns: %d", my\_strcmp("ABCD", "ABCDE"));

    printf("\n\"ABCD\" is less than \"ABND\" ... strcmp returns: %d", my\_strcmp("ABCD", "ABND"));

    printf("\n\"ABCD\" is equal than \"ABCD\" ... strcmp returns: %d", my\_strcmp("ABCD", "ABCD"));

    printf("\n\"ABCD\" is less than \"ABCd\" ... strcmp returns: %d", my\_strcmp("ABCD", "ABCd"));

    printf("\n\"Orange\" is greater than \"Apple\" ... strcmp returns: %d\n", my\_strcmp("Orange", "Apple"));

    return 0;

}

int my\_strlen(const char \*s)

{

    int count = 0;

    while (\*s != '\0')

    {

        count++;

        s++;

    }

    return count;

}

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

{

    // move dest pointer to end of string pointed by dest

    while (\*dest != '\0')

    {

        dest++;

    }

    for (int i = 0; i < n; i++)

    {

        // this condition will break the loop if n < strlen(source)

        if(\*source == '\0')

            break;

        \*dest = \*source;

        dest++;

        source++;

    }

    \*dest = '\0';

}

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

{

    while(!(\*str1 == '\0' || \*str2 == '\0'))

    {

        if(\*str1 != \*str2)

        {

            return (int) (\*str1 - \*str2);

        }

        else

        {

            str1++;

            str2++;

        }

    }

    return (int) (\*str1 - \*str2);

}

# Exercise B – Program Output

Text

Description automatically generated

# Exercise C – AR Diagram

Chart, diagram

Description automatically generated

# Exercise E – Source Code of functions

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

{

    struct cplx result;

    result.real = z1.real + z2.real;

    result.imag = z1.imag + z2.imag;

    return result;

}

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

{

    (\*difference).real = z1.real - z2.real;

    (\*difference).imag = z1.imag - z2.imag;

}

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

{

    double a = (\*pz1).real;

    double b = (\*pz1).imag;

    double c = (\*pz2).real;

    double d = (\*pz2).imag;

    (\*product).real = (a\*c - b\*d);

    (\*product).imag = (a\*d + b\*c);

}