

Programming Fundamentals – ENSF 337

Lab 4

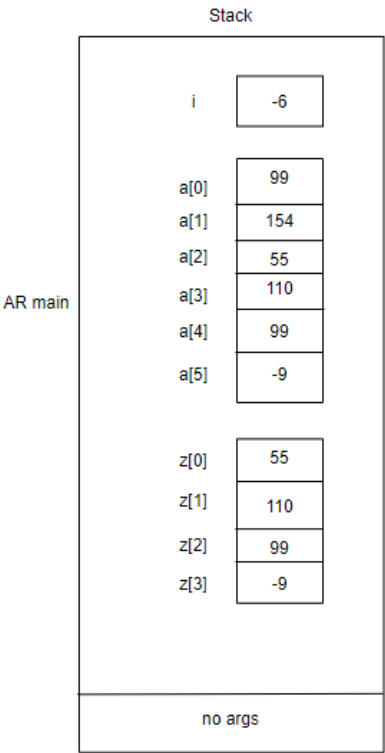
M. Moussavi

Jaxon Braun

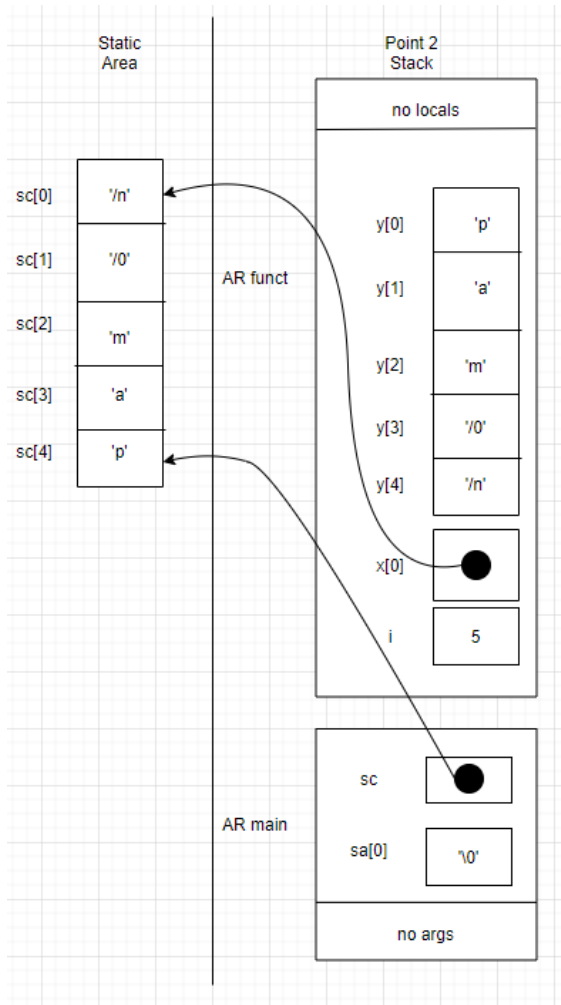
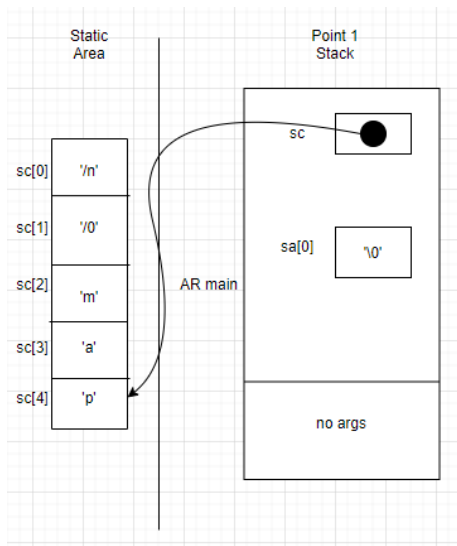
B01

Submitted on October 20, 2021

Exercise A: Pointer Arithmetic in C



Exercise B: More Practice with Pointers and Pointer Arithmetic



Exercise C: A Simple Macro

```
// lab2exC.c
// ENSF 337 Lab 4 Exercise C
// Completed by: Jaxon Braun
// Submission Date: October 20, 2021

#include <stdio.h>

#define ELEMENTS(array) (sizeof(array)/sizeof(array[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;
}
```

```
Array a has 5 elements and macro ELEMENTS returns 5
Array b has 20 elements and macro ELEMENTS returns 20
```

Exercise D: Duplicating Library Function, Using Pointer Arithmetic

```
/*
 * lab4exD.c
 * ENSF 337 Lab 4 Exercise D
 * Completed by: Jaxon Braun
 * Submission Date: October 20, 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 n);
/* 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 identical.
 *
 *   Returns a negative number if str1 is less than str2.
 *
 *   Returns a positive number if 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 library 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 strcat FUNCTION ... \n");
/* strcat append the first 3 characters of str5 to the end of my_string */
my_strcat(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_strcat(my_string, str2, 4);
printf("\nExpected to display: my_string contains \"tic-tac\"");
printf("\nmy_string contains: \"%s\"", my_string);


/* strcat append ONLY up to '\0' character from str3 -- not 6 characters */
my_strcat(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 = strcmp("ABCD", "ABND")) < 0)
    printf("\n\"ABCD\" is less than \"ABND\" ... strcmp returns %d", y);
```

```
if((y = strcmp("ABCD", "ABCD")) == 0)
    printf("\n\"ABCD\" is equal \"ABCD\" ... strcmp returns %d", y);
```

```
if((y = strcmp("ABCD", "ABCd")) < 0)
    printf("\n\"ABCD\" is less than \"ABCd\" ... strcmp returns %d", y);
```

```
if((y = strcmp("Orange", "Apple")) > 0)
    printf("\n\"Orange\" is greater than \"Apple\" ... strcmp returns %d\n", y);
```

```
return 0;
```

```
}
```

```
int my_strlen(const char *s){
```

```
    int length = 0;
```

```
    while (*(s+length) != '\0'){
```

```
        length++;
```

```
    }
```

```
    return length;
```

```
}
```

```
void my_strncat(char *dest, const char *source, int n){
```

```
    int dest_length = 0;
```

```
    while (*(dest+dest_length) != '\0'){
```

```
        dest_length++;
```

```
    }
```

```
    int i = 0;
```

```
    while (n > 0){
```

```
        *(dest+dest_length) = *(source+i);
```

```
        i++;
```

```
        dest_length++;
```

```
        n--;
```

```
    }
```

```
    *(dest+dest_length) = '\0';
```

```
}
```


TESTING strlen FUNCTION ...

Expected to display: my_string length is 0.
my_string length is 0.
Expected to display: my_string size is 100 bytes.
my_string size is 100 bytes.
Expected to display: my_string contains banana.
my_string contains banana
Expected to display: my_string length is 6.
my_string length is 6.
Expected to display: my_string contains "".
my_string contains:""
Expected to display: my_string length is 0.
my_string length is 0.
Expected to display: my_string size is still 100 bytes.
my_string size is still 100 bytes.

TESTING strncat FUNCTION ...

Expected to display: my_string contains "tic"
my_string contains "tic"
Expected to display: my_string length is 3.
my_string length is 3.
Expected to display: my_string contains "tic-tac"
my_string contains:"tic-tac"
Expected to display: my_string contains "tic-tac-toe"
my_string contains:"tic-tac-toe"
Expected to display: my_string has 11 characters.
my_string has 11 characters.

Using strcmp - C library function:

Expected to display: "ABCD" is less than "ABCDE"
"ABCD" is less than "ABCDE"

TESTING strcmp FUNCTION ...

"ABCD" is less than "ABND" ... strcmp returns -1
"ABCD" is equal "ABCD" ... strcmp returns 0
"ABCD" is less than "ABCd" ... strcmp returns -1
"Orange" is greater than "Apple" ... strcmp returns 1

Exercise E: Reading Numeric Input as String

```
/* prog_two.c

* ENSF 337 Lab 4 Exercise E

* Completed by: Jaxon Braun

* Submission Date: October 20, 2021

*/

#include <stdio.h>

#include <limits.h>

#include "read_input.h"

#include "read_input.c"

#include "read_double.c"

#define SIZE 50

int main(void)

{

    double n = 0;

    char digits[SIZE];

    int y = EOF;

    while (1)

    {

        printf("\n\nEnter an real number or press Ctrl-D to quit: ");

        y = read_real(digits, SIZE, &n);

        if(y == 1)

            printf("\nYour real number value is: %f", n);

        else if(y == EOF){

            printf("\nGood Bye.\n");

            break;

        }

        else

            printf("\n%s is an invalid real rumber.", digits);

    }

    return 0;

}
```

```

/* read_int.c

* ENSF 337 Lab 4 Exercise E

* Completed by: Jaxon Braun

* Submission Date: October 20, 2021

*/

#include "read_input.h"

#include <math.h>

int read_real(char* digits, int n, double * num){

    if(get_string(digits, n)== EOF)

        return EOF;

    if(is_valid_double(digits)){

        if(digits[0] == '-')

            *num = -convert_to_double(digits + 1);

        else if(digits[0] == '+')

            *num = convert_to_double(digits + 1);

        else

            *num = convert_to_double(digits);

        return 1;

    }

    return 0;

}

int is_valid_double(const char* digits){

    int valid = 1;

    int i;

    if(digits[0] == '+' || digits[0] == '-')

        i = 1;

    else

        i = 0;

    if (digits[i] == '\0')

        valid = 0;

```

```

else

    while (valid && (digits[i] != '\0')) {

        if (digits[i] < '.' || digits[i] > '9' || digits[i] == '/')

            valid = 0;

        i++;

    }

    return valid;

}

```

```

double convert_to_double(const char *digits){

    double sum = 0;

    int i = 0;

    double decimal_point = 0;

    while(digits[i] != '\0') {

        if (digits[i] != '.' && decimal_point == 0){

            sum = (10 * sum) + (digits[i] - '0');

            i++;

        }

        else if (digits[i] == '.'){

            decimal_point = 1;

            i++;

        }

        else{

            double divisor = 1;

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

                divisor *= 10.0;

            }

            sum = sum + ((digits[i] - '0') / divisor);

            decimal_point++;

            i++;

        }

    }

    return sum;

}

```

```
Enter an real number or press Ctrl-D to quit: 23.4
Your real number value is: 23.400000

Enter an real number or press Ctrl-D to quit: .56
Your real number value is: 0.560000

Enter an real number or press Ctrl-D to quit: -.23
Your real number value is: -0.230000

Enter an real number or press Ctrl-D to quit: -0.45
Your real number value is: -0.450000

Enter an real number or press Ctrl-D to quit: -0.0000067
Your real number value is: -0.000007

Enter an real number or press Ctrl-D to quit: 564469999
Your real number value is: 564469999.000000

Enter an real number or press Ctrl-D to quit: +8773469
Your real number value is: 8773469.000000

Enter an real number or press Ctrl-D to quit: +.5
Your real number value is: 0.500000

Enter an real number or press Ctrl-D to quit:
Good Bye.
```