Ejercitación – Serie I

Strings

Conceptos que se agregan en esta serie: Arrays como cadena de caracteres conocidos habitualmente como Strings.

Preferentemente utilice lógica de punteros para desarrollar sus funciones

---

Para probar las funciones solicitadas, será necesario generar programas y funciones con la posibilidad de ingresar los datos e imprimir los resultados según corresponda.

Salvo se especifique, las funciones solicitadas NO deben imprimir ningún tipo de dato o información. Si fuese necesario imprimir algún resultado, estos deberán imprimirse por fuera de la función a partir del valor retornado.

Recuerden que no está permitido el uso de variables globales.

Ejercicio 1:

Implemente y pruebe su propia versión de las siguientes funciones declaradas en string.h, antecediendo al nombre de las mismas el prefijo **my_**.

```
int strcmp (const char *s1, const char *s2);
int strncmp(const char *s1, const char *s2, size_t n);
```

Respete sus prototipos y utilice el man para saber cómo se debe comportar cada una de ellas.

Para verificar el funcionamiento de las funciones realizadas, ejecute el siguiente código, el cual deberá tener la misma salida que se visualiza luego del mismo.

```
#include <stdio.h>
int main()
{
char s1[]="abcdefghaij";
char s2[20]="ABZ", *sx="por las dudas.";
int aux,i;
    for (i=0; i<=3;i++) {
        switch(i){
            case 1: sx=s1; break;
            case 2: sx="abcde"; break;
            case 3: sx=s2; break;
        }
        printf ("\nComparamos s1: %s\n\ty sx: %s \n",s1,sx);
        aux= my_strcmp(s1,sx);
        if (!aux)
            printf (" --> son iguales\n",sx);
        else{
            printf (" -- s1 es %s que sx",(aux<0)?"menor":"mayor");
            printf ("\ny si solo comparamos las primeras 4 letras\n");
            if (my_strncmp(s1,sx,4))
                printf (" --> tambien son distintos\n",sx);
            else
                printf (" --> son coincidentes\n",sx);
        }
    }
return 0;
}
```

```
Comparamos s1: abcdefghaij
       y sx: por las dudas.
 -- s1 es menor que sx
y si solo comparamos las primeras 4 letras
 --> tambien son distintos
Comparamos s1: abcdefghaij
       y sx: abcdefghaij
 --> son iguales
Comparamos s1: abcdefghaij
       y sx: abcde
 -- s1 es mayor que sx
y si solo comparamos las primeras 4 letras
 --> son coincidentes
Comparamos s1: abcdefghaij
       y sx: ABZ
 -- s1 es mayor que sx
y si solo comparamos las primeras 4 letras
 --> tambien son distintos
```

Ejercicio 2:

Implemente y pruebe su propia versión de las siguientes funciones declaradas en string.h, antecediendo al nombre de las mismas el prefijo **my_**.

```
size_t strlen(const char *s);
char *strcpy(char *dest, const char *src);
char *strncpy(char *s1, const char *s2, size_t n);
char *strcat(char*s1, const char *s2);
char *strncat(char*s1, const char *s2, size_t n);
```

Respete sus prototipos y utilice el man para saber cómo se debe comportar cada una de ellas.

Está prohibido el uso de cualquier función preexistente.

Para verificar el funcionamiento de las funciones implementadas, ejecute el siguiente código, el cual deberá tener la misma salida que se visualiza luego del mismo.

```
#include <stdio.h>
#define ABZ "ABC-XYZ"

int main()
{
char s1[80]="abcdefghaijklmnopqrstuvwxyz";
char s2[80];
char *s3="1234567890-";

    printf ("Los largos de s1 y s3 son: %d y %d\n",my_strlen(s1), my_strlen(s3));

    my_strcpy(s2,s1);
    my_strcat(s2,s3);
    my_strncat(s1,s3,4);
    printf ("En s1 tengo:[%s]\ny en s2:[%s]\n",s1,s2);

    my_strncpy (s1+10,ABZ, my_strlen(ABZ));
    printf ("Ahora en s1 tengo:[%s]\n ",s1);
return 0;
}
```

```
Los largos de s1 y s3 son: 27 y 11
En s1 tengo:[abcdefghaijklmnopqrstuvwxyz1234]
y en s2:[abcdefghaijklmnopqrstuvwxyz1234567890-]
Ahora en s1 tengo:[abcdefghaiABC-XYZqrstuvwxyz1234]
```

Ejercicio 3:

Implemente y pruebe su propia versión de las siguientes funciones declaradas en string.h, antecediendo al nombre de las mismas el prefijo **my_**.

```
char *strchr(char *s, int c);
char *strstr(const char *s1, const char *s2);
```

Respete sus prototipos y utilice el man para saber cómo se debe comportar cada una de ellas.

Está prohibido el uso de cualquier función preexistente, a excepción de las implementas en puntos anteriores.

Para verificar el funcionamiento de las funciones realizadas, ejecute el siguiente código, el cual deberá tener la misma salida que se visualiza luego del mismo.

```
#include <stdio.h>
void informamos (const char *s1,const char *s2)
{
    if (s2) // o s2!=NULL
    {
        printf ("\nEncontrado en la posicion %d",s2-s1);
        printf ("\ny a partir de ahi tenemos: %s",s2);
    }
    else
        printf ("\nNo encontrado en el texto");
}
int main()
{
char *s1="Estamos desarrollando funciones";
char *sx, s2[]="carro", aux;
    printf ("s1: %s",s1);
    for (aux='e'; aux<='h'; aux++)
    {
        printf ("\nBuscamos el caracter:%c",aux);
        sx=my_strchr(s1,aux);
        informamos (s1,sx);
    }
    printf ("\nAhora buscamos el texto: %s",s2);
    sx=my_strstr(s1,s2);
    informamos (s1,sx);
    printf ("\nAhora buscamos el texto: %s",s2+1);
    sx=my_strstr(s1,s2+1);
    informamos (s1,sx);
return 0;
}
```

```
s1: Estamos desarrollando funciones
Buscamos el caracter:e
Encontrado en la posicion 9
y a partir de ahi tenemos: esarrollando funciones
Buscamos el caracter:f
Encontrado en la posicion 22
y a partir de ahi tenemos: funciones
Buscamos el caracter:g
No encontrado en el texto
Buscamos el caracter:h
```

```
No encontrado en el texto
Ahora buscamos el texto: carro
No encontrado en el texto
Ahora buscamos el texto: arro
Encontrado en la posicion 11
y a partir de ahi tenemos: arrollando funciones
```

Ejercicio 4:

Implemente y pruebe su propia versión de la siguiente función declarada en string.h, antecediendo al nombre de la misma el prefijo **my_**.

```
size_t strspn(const char *s1, const char *s2);
```

Respete su prototipo y utilice el man para saber cómo se debe comportar cada una de ellas.

Está prohibido el uso de cualquier función preexistente, a excepción de las implementas en puntos anteriores.

Para verificar el funcionamiento de la función desarrollada, ejecute el siguiente código, el cual deberá tener la misma salida que se visualiza luego del mismo.

```
#include <stdio.h>
int main()
{
   char s1[] = "Hola a todos";
   char *sx = "al Ho";
   char *sy = "alHo";
   char *sz = "axHo";

   printf( "s1=[%s] \n", s1 );
   printf( "spn_sx: %d  sx=[%s]\n", my_strspn(s1, sx), sx);
   printf( "spn_sy: %d  sy=[%s]\n", my_strspn(s1, sy), sy);
   printf( "spn_sz: %d  sz=[%s]\n", my_strspn(s1, sz), sz);
return 0;
}
```

```
s1=[Hola a todos]
spn_sx: 7  sx=[al Ho]
spn_sy: 4  sy=[alHo]
spn_sz: 2  sz=[axHo]
```

Ejercicio 5:

Implemente y pruebe su propia versión de la siguiente función declarada en string.h, antecediendo al nombre de la misma el prefijo **my_**.

```
char *strpbrk(const char *str1, const char *str2);
```

Respete su prototipo y utilice el man para saber cómo se debe comportar cada una de ellas.

Está prohibido el uso de cualquier función preexistente, a excepción de las implementas en puntos anteriores.

Para verificar el funcionamiento de la función desarrollada, ejecute el siguiente código, el cual deberá tener la misma salida que se visualiza luego del mismo.

```
int main()
{
   char s1[] = "Hola a todos, hoy es ...";
   char *sx = "xyz123";
   char *sy = "bd .";
   char *sz = "bruc";

   printf( "s1=[%s] \n", s1 );
   printf( "sx=[%s] : %s\n", sx,my_strpbrk(s1,sx));
   printf( "sy=[%s] : %s\n", sy,my_strpbrk(s1,sy));
   printf( "sz=[%s] : %s\n", sz,my_strpbrk(s1,sz)); // antiguamente esta línea hubiese hecho caer a la aplicación
```

```
return 0;
}
```

```
s1=[Hola a todos, hoy es ...]
sx=[xyz123] : y es ...
sy=[bd .] :  a todos, hoy es ...
sz=[bruc] : (null)
```

Ejercicio 6:

Desarrollar las siguientes funciones

```
char *my_strLower (char *str1, const char *str2);
char *my_strnLower(char *str1, const char *str2,int n);
char *my_strUpper (char *str1, const char *str2);
char *my_strnUpper(char *str1, const char *str2,int n);
```

Las cuales copiarán el contenido de **s2** a **s1** transformando todos los caracteres a minúscula (Lower) o a mayúscula (Upper), según corresponda.

En todos los casos, la función devolverá el valor de **s1**.

En los casos de `my_strnLower` y `my_strUpper`, se transferirán solo **n** caracteres, y se agregará el finalizador de strings ('\0').

Por ejemplo, dado el siguiente valor de str2, "El dos (2), es Menor que el 5 (cinco)", tendremos las siguientes salidas en cada caso:

`puts (my_strLower (s1,str2));`	el dos (2), es menor que el 5 (cinco)
`puts (my_strnLower(s1,str2,10));`	el dos (2)
`puts (my_strUpper (s1,str2));`	EL DOS (2), ES MENOR QUE EL 5 (CINCO)
`puts (my_strnUpper(s1,str2,9));`	EL DOS (2

Ejercicio 7:

7a.- Desarrollar una función que invierta el sentido del texto de forma tal que quede de atrás hacia adelante.

```
int my_strInvert (char *str1);
```

Por ejemplo, si se recibe el texto "Hola mundo", la salida deberá ser "odnum aloH".

La función debe retornar la cantidad de caracteres del string, y debe soportar recibir NULL como parámetro, en cuyo caso retornará 0 (cero).

7b.- Realice una función que reciba un strings y devuelva el carácter que más veces se repite. En caso que haya más de un carácter que cumpla esta condición, se debe retornar aquel que posea el mayor código ASCII.

Ejercicio 8:

8.a.- Desarrollar una función para reemplazar un determinado carácter de un texto por otro.

Por ejemplo, si tenemos el texto: "Hola, ¿cómo están todos?"

Y se solicita cambiar la letra 'o' por la 'U', el texto debería quedar: "HUla, ¿cómU están tUdUs?"

El valor de retorno será la cantidad de cambios realizados, en este caso: 4.

La función debe soportar recibir NULL como parámetro, en cuyo caso retornará 0 (cero).

Los cambios se realizan sobre el mismo string.

8.b- Ídem al punto a, pero para reemplazar cadenas de caracteres.

Ejercicio 9:

Basado en el las funciones realizadas en el punto 1, realice las modificaciones necesarias para que las funciones de comparación no sean case sensitive. Es decir que Alfa sea igual a alfa o ALFA.

```
int my_strxcmp (const char *s1, const char *s2);
int my_strxncmp(const char *s1, const char *s2, size_t n);
```

Ejercicio 10:

Realice una aplicación que permita ingresar varios nombres y que finalizado el mismo, imprima:

- El nombre que este primero según orden alfabético
- El nombre que este último según orden alfabético

El fin de ingreso se da cuando dentro del texto ingresado exista un número.

El ingreso debe permitir incluir espacios como separación de nombres.

La evaluación de primero/último no debe ser case sensitive

El largo máximo de ingreso queda a criterio del desarrollador.