# Librería string.h

La librería string.h del lenguaje de programación C proporciona una serie de funciones para trabajar con cadenas de caracteres (strings). Algunas de las funciones más comunes que se pueden encontrar en esta librería son:

1. Funciones para manipulación de cadenas:

   strlen(): devuelve la longitud de una cadena.

   strcpy(): copia una cadena en otra.

   strncpy(): copia una cantidad específica de caracteres de una cadena a otra.

   strcat(): concatena una cadena al final de otra.

   strncat(): concatena una cantidad específica de caracteres de una cadena al final de otra.

   strcmp(): compara dos cadenas y devuelve un valor indicando si son iguales o cuál es mayor.

2. Funciones para búsqueda de caracteres:

   strchr(): busca la primera ocurrencia de un carácter en una cadena.

   strrchr(): busca la última ocurrencia de un carácter en una cadena.

   strstr(): busca la primera ocurrencia de una cadena en otra.

3. Funciones para manipulación de memoria:

   memset(): establece un bloque de memoria a un valor específico.

   memcpy(): copia un bloque de memoria de una ubicación a otra.

   memcmp(): compara dos bloques de memoria y devuelve un valor indicando si son iguales o cuál es mayor.

# Librería string.h

## EJEMPLOS

Strlen() Supongamos que queremos obtener la longitud de una cadena de caracteres llamada "nombre" y guardarla en una variable llamada "longitud". El código para hacer esto sería el siguiente:

```
#include <stdio.h>

#include <string.h>

int main() {

   char nombre[] = "Juan";   // Declarar y asignar una cadena de caracteres

   int longitud = strlen(nombre);   // Obtener la longitud de la cadena

   printf("La longitud de la cadena es: %d\n", longitud); // Imprimir la longitud obtenida

   return 0;

}
```

La pantalla muestra

La longitud de la cadena es: 4

La función strcpy() se utiliza para copiar una cadena de caracteres a otra. En este ejemplo, se copiará una cadena llamada "origen" en otra cadena llamada "destino".

```
#include <stdio.h>

#include <string.h>

int main() {

   char origen[] = "Cadena a copiar";   // Declarar y asignar una cadena de caracteres

   char destino[20];   // Declarar una segunda cadena para copiar

   strcpy(destino, origen);   // Copiar la cadena origen en la cadena destino

   printf("La cadena copiada es: %s\n", destino);   // Imprimir la cadena copiada

   return 0;

}
```

La pantalla muestra

La cadena copiada es: Cadena a copiar

La función strcat() se utiliza para concatenar dos cadenas de caracteres. En este ejemplo, se concatenará una cadena llamada "apellido" en otra cadena llamada "nombre".

```
#include <stdio.h>
```

# Librería string.h

```
#include <string.h>

int main() {
   char nombre[20] = "Juan "; // Declarar y asignar una cadena de caracteres
   char apellido[] = "Pérez"; // Declarar una segunda cadena para concatenar
   strcat(nombre, apellido); // Concatenar la cadena apellido en la cadena nombre
   printf("El nombre completo es: %s\n", nombre); // Imprimir el nombre completo
   return 0;
}
```

La pantalla muestra

El nombre completo es: Juan Pérez

La función strncat() se utiliza para concatenar una cantidad específica de caracteres de una cadena en otra cadena. En este ejemplo, se concatenarán los primeros 3 caracteres de una cadena llamada "apellido" en otra cadena llamada "nombre".

```
#include <stdio.h>
#include <string.h>
int main() {
char nombre[20] = "Juan ";  // Declarar y asignar una cadena de caracteres
char apellido[] = "Perez";  // Declarar una segunda cadena para concatenar
strncat(nombre, apellido, 3);  // Concatenar los primeros 3 caracteres de la cadena apellido en la cadena nombre
 printf("El nombre completo es: %s\n", nombre);  // Imprimir el nombre completo
 return 0;
}
```

La pantalla muestra

El nombre completo es: Juan Per

La función strcmp() se utiliza en lenguaje de programación C para comparar dos cadenas de caracteres. Devuelve un valor entero que indica la relación entre las cadenas.

```
#include <stdio.h>
#include <string.h>
```

```
int main()
{
   char str1[] = "juan";
   char str2[] = "josesito";
   int resultado;
   resultado = strcmp(str1, str2);
   if (resultado == 0)
   {          printf("Las cadenas son iguales\n");     }
   else if (resultado < 0)
   {         printf("La cadena \"%s\" es menor que la cadena \"%s\"\n", str1, str2);     }
   else
   {        printf("La cadena \"%s\" es mayor que la cadena \"%s\"\n", str1, str2);      }
    return 0;
}
```

La pantalla muestra

La cadena "juan" es mayor que la cadena "josesito"

Se busca la primera aparición del carácter 'm' en la cadena "Hola mundo" sin utilizar un puntero. Se recorre la cadena utilizando un bucle for y se comprueba si cada carácter es igual al carácter que se busca. Si se encuentra el carácter, se imprime el mensaje indicando la posición en la cadena y se sale del programa utilizando la instrucción return. Si no se encuentra el carácter, se imprime un mensaje indicándolo.

```
#include <stdio.h>
#include <string.h>
int main()
{
   char str[] = "Hola mundo";
   char buscar = 'd';   // se usan comillas simples para un caracter
   int i;
   for (i = 0; i < strlen(str); i++)
   {
      if (str[i] == buscar)
      {
         printf("El caracter '%c' se encuentra en la cadena '%s' en la posicion %d\n", buscar, str, i);
         return 0;     // Se encontró el carácter, se sale del programa
      }
   }
   printf("El caracter '%c' no se encuentra en la cadena '%s'\n", buscar, str);
   return 0;    // No se encontró el carácter, se sale del programa
}
```

En este ejemplo, se busca la primera aparición del carácter 'm' en la cadena "Hola mundo" sin utilizar un puntero. Se recorre la cadena utilizando un bucle for y se comprueba si cada carácter es igual al carácter que se busca. Si se encuentra el carácter, se imprime el mensaje indicando la posición en la cadena y se sale del programa utilizando la instrucción return. Si no se encuentra el carácter, se imprime un mensaje indicándolo.

Este método funciona bien para cadenas pequeñas, pero puede ser menos eficiente para cadenas grandes ya que tiene que recorrer toda la cadena para buscar el carácter. En este caso, la función strchr() que utiliza punteros es más eficiente. Como veremos en el siguiente ejemplo

La función strchr() se utiliza en lenguaje de programación C para buscar la primera aparición de un carácter en una cadena de caracteres. Devuelve un puntero al carácter encontrado, o un puntero nulo si el carácter no se encuentra en la cadena.

```
#include <stdio.h>
#include <string.h>
```

```
int main()
{   char str[] = "Hola mundo";
    char *ptr;
    ptr = strchr(str, 'm');// me informa la `posisición del caracter m en la variable str
    if (ptr != NULL)
    {  printf("%d %d\n", str, ptr );   // se imprime la posición donde empieza la variable str y donde apunta ptr
       printf("El caracter 'm' se encuentra en la cadena \"%s\" en la posición %d\n", str, ptr - str);
    }
    else
    {  printf("El caracter 'm' no se encuentra en la cadena \"%s\"\n", str);
    }
    return 0;
}
```

La pantalla muestra

6487552 6487557

El caracter 'm' se encuentra en la cadena "Hola mundo" en la posici¾n 5

En este ejemplo, se busca la primera aparición del carácter 'm' en la cadena "Hola mundo". La función strchr() devuelve un puntero al carácter encontrado, por lo que se comprueba si el valor del puntero es nulo o no. Si el puntero no es nulo, se imprime el mensaje indicando la posición del carácter en la cadena utilizando la aritmética de punteros para calcular la distancia entre el puntero devuelto y el inicio de la cadena. En este caso, el carácter 'm' se encuentra en la posición 5 de la cadena "Hola mundo".

La función strrchr() se utiliza en lenguaje de programación C para buscar la última aparición de un carácter en una cadena de caracteres. Devuelve un puntero al carácter encontrado, o un puntero nulo si el carácter no se encuentra en la cadena.

En el siguiente ejemplo, se busca la última aparición del carácter 'o' en la cadena "Hola mundo". La función strrchr() devuelve un puntero al carácter encontrado, por lo que se comprueba si el valor del puntero es nulo o no. Si el puntero no es nulo, se imprime el mensaje indicando la posición del carácter en la cadena utilizando la aritmética de punteros para calcular la distancia entre el puntero devuelto y el inicio de la cadena. En este caso, la última aparición del carácter 'o' se encuentra en la posición 7 de la cadena "Hola mundo".

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str[] = "Hola mundo";
```

```
    char *ptr;
    ptr = strrchr(str, 'o');
    if (ptr != NULL)
    {
        printf("El caracter 'o' se encuentra en la cadena \"%s\" en la posición %ld\n", str, ptr - str);
    }
    else
    {
        printf("El caracter 'o' no se encuentra en la cadena \"%s\"\n", str);
    }
    return 0;
}
```

La pantalla muestra

El caracter 'o' se encuentra en la cadena "Hola mundo" en la posici¾n 9

Por último la función strstr() se utiliza en lenguaje de programación C para buscar la primera aparición de una subcadena en una cadena de caracteres. Devuelve un puntero al inicio de la subcadena encontrada, o un puntero nulo si la subcadena no se encuentra en la cadena.

```
#include <stdio.h>

#include <string.h>

int main()

{

   char str[] = "Hola mundo";

   char sub[] = "mun";

   char *ptr;

   ptr = strstr(str, sub);

   if (ptr != NULL)

   {

  printf("La subcadena \"%s\" se encuentra en la cadena \"%s\" en la posicion %ld\n", sub, str, ptr - str);

   }

   else

   {

      printf("La subcadena \"%s\" no se encuentra en la cadena \"%s\"\n", sub, str);

   }


   return 0;

}
```

La pantalla muestra

La subcadena "mun" se encuentra en la cadena "Hola mundo" en la posicion 5

Es una función que es muy útil para buscar palabras en un texto largo

La función memset() se utiliza en lenguaje de programación C para asignar un valor específico a una sección de memoria. Se utiliza principalmente para inicializar variables y arreglos con un valor específico.

Ejemplo

```
#include <stdio.h>
#include <string.h>
int main()

{
   char str[] = "Hola mundo";
   memset(str, '-', 5);      // Inicializa los primeros 5 bytes de str con el valor '-'
   printf("La cadena inicializada es: %s\n", str);

   return 0;
}
```

En este ejemplo, se inicializan los primeros 5 bytes de la cadena "Hola mundo" con el valor '-' utilizando la función memset(). El primer argumento es un puntero a la sección de memoria que se desea inicializar, el segundo argumento es el valor que se desea asignar y el tercer argumento es el número de bytes que se desea inicializar.

Después de llamar a la función memset(), la cadena "Hola mundo" queda inicializada como "----- mundo". Es importante tener en cuenta que la función memset() asigna el valor especificado a cada byte en la sección de memoria, por lo que se debe tener cuidado al utilizar esta función para inicializar variables y arreglos.

# Librería string.h

La función memcpy() se utiliza en lenguaje de programación C para copiar un bloque de memoria de una ubicación a otra. Es útil cuando se necesita copiar grandes cantidades de datos de una ubicación a otra en la memoria.

```
#include <stdio.h>
#include <string.h>
int main()
{
 char src[] = "Hola mundo";
 char dest[5];
 memcpy(dest, src, strlen(src) + 1); // Copia src en dest sin importar el tamaño original de dest

printf("La cadena copiada es: %s\n", dest);

int longitud = strlen(dest);   // Obtener la longitud de la cadena
printf("La longitud de la cadena es: %d\n", longitud); // podemos ver que la longitud de la cadena original cambio
return 0;
}
```

En este ejemplo, se copia la cadena "Hola mundo" en la variable dest utilizando la función memcpy(). El primer argumento es un puntero a la ubicación de memoria de destino, el segundo argumento es un puntero a la ubicación de memoria de origen y el tercer argumento es el número de bytes que se desea copiar.

En este caso, el tercer argumento es la longitud de la cadena src más uno para copiar el carácter nulo al final de la cadena. Después de llamar a la función memcpy(), la cadena "Hola mundo" se copia en la variable dest.

Es importante tener en cuenta que la función memcpy() no comprueba si la ubicación de memoria de origen y destino se superponen, por lo que se debe tener cuidado al utilizar esta función para copiar datos.

```
#include <stdio.h>
#include <string.h>
int main()

{
   char str1[] = "Hola mundo";
   char str2[] = "Hola";
```

```
    int result;

    result = memcmp(str1, str2, strlen(str2)); //siempre que str2 > str1
    if (result < 0)
    {       printf("La cadena \"%s\" es menor que la cadena \"%s\"\n", str1, str2);    }
    else if (result > 0)
    {       printf("La cadena \"%s\" es mayor que la cadena \"%s\"\n", str1, str2);    }
    else
    {       printf("Ambas cadenas son iguales hasta la longitud de \"%s\"\n", str2);    }
    return 0;
}
```

En este ejemplo, se comparan las cadenas "Hola mundo" y "Hola" utilizando la función memcmp(). El primer argumento es un puntero al primer bloque de memoria, el segundo argumento es un puntero al segundo bloque de memoria y el tercer argumento es el número de bytes que se desea comparar.

En este caso, se compara la longitud de la cadena "Hola" (4 bytes). Después de llamar a la función memcmp(), se comprueba el valor de retorno para determinar la relación entre las dos cadenas. En este caso, la cadena "Hola mundo" es mayor que la cadena "Hola", por lo que se imprime el mensaje correspondiente.

Es importante tener en cuenta que la función memcmp() compara bloques de memoria byte a byte y no tiene en cuenta el significado de los datos.