**Pasar un string para modificarlo en una función**

Hay que tener en cuenta cómo ha sido declarado, ya que un string puede ser declarado como array de caracteres o como puntero:

* Como array: str[] = “hola”; -> 5 bytes -> h o l a \0

str[10] = “hola”; -> 10 bytes -> h o l a \0 \_ \_ \_ \_ \_

* Como puntero \*str = “hola”; -> 4 bytes -> 0x24844484844

Depende de cómo se declare se guarda en una parte de la memoria o en otra y afecta a lo que se puede hacer o no.

Cuando se declara como array puede ser una variable automática (variable local), o una variable global (accesible a todo el programa) o estática (mantiene su valor al salir de la función). La primera se guarda en la pila (*stack memory*), que es una memoria de lectura y escritura. Por tanto, podemos acceder y realizar cambios. Las otras dos se guardan en “data segment” que también es de lectura/escritura.

Cuando se declara como puntero se guarda en un tipo de memoria de solo lectura, por lo que no nos va a permitir hacer cambios en el string. Aunque podemos salvar este problema utilizando malloc para guardarlo en la *heap memory*, y entonces podremos realizar cambios.

**Pasar un string declarado como array**

Simplemente hay que pasar el nombre del array y **recibirlo como array** en la función:

void prueba(char str[]) **// recibido como array**

{

str[3] = 'c';

}

int main()

{

char s1[] = "Adios";

prueba(s1); **// pasado como array**

printf("%s\n", s1);

}

Adics

O crear un puntero que apunte al array, pasarlo a la función y **recibirlo como array**:

void prueba(char str[]) **// recibido como array**

{

str[3] = 'c';

}

int main()

{

char s1[] = "Adios";

char \*ptr;

ptr = s1;

prueba(ptr); **// pasado como puntero**

printf("%s\n", ptr);

}

Adics

O crear un puntero que apunte al array, pasarlo a la función y **recibirlo como puntero**:

void prueba(char \*str) **// recibido como puntero**

{

str[3] = 'c';

}

int main()

{

char s1[] = "Adios";

char \*ptr;

ptr = s1;

prueba(ptr); **// pasado como puntero**

printf("%s\n", ptr);

}

Adics

**Pasar un string declarado como puntero**

Como ya hemos dicho, al declarar una variable que contiene un puntero a string, lo que contiene esa variable es una dirección de memoria. Y no vas a poder modificar el contenido del string, porque está guardado en una zona de memoria de solo lectura. Pero sí puedes asignar a esa variable otro string diferente que estará en otra dirección de memoria.

int main()

{

char \*str = "hola"; **// str apunta a la dirección 0x55779c0bc774**

printf("%p\n", str);

str = "adios"; **// ahora str apunta a la dirección 0x55779c0bc77d**

printf("%p\n", str);

printf("%s\n", str);

}

0x55779c0bc774

0x55779c0bc77d

adios

Si quieres hacer lo mismo que hemos hecho arriba pero en una función aparte, debes pasar a la función la dirección de memoria donde se guarda la variable (&str), en lugar de pasar la dirección a la que está apuntando (str).

void prueba(char \*\*str) **// se recibe como puntero a puntero**

{

\*str = "adios"; **// se dereferencia para usarlo**

}

int main()

{

char \*str = "hola";

prueba(&str); **// se pasa la dirección de str**

printf("%s\n", str);

}

adios

<https://www.geeksforgeeks.org/storage-for-strings-in-c/>