

Strings

Prof. Dr. Hans H. Ccacyahuillca Bejar





Cadena de caracteres (string)

- Secuencia de:
 - letras ('a', 'b', 'A', 'B'),
 - símbolos ('!', '?', '+', '=', '%', ...),
 - espacio en blanco (' ')
 - dígitos ('0', '1', ..., '9')
 - termina en el carácter '\0'.
- En lenguaje C los **strings** son almacenados en arreglos de tipo **char**.


```
char texto[500];
```

Tabla ASCII

- ASCII es un patrón donde cada carácter es utilizado en formato de **código binario**.

SIMB	DEC	BINARIO	SIMB	DEC	BINARIO
3	51	00110011	<	60	00111100
4	52	00110100	=	61	00111101
5	53	00110101	>	62	00111110
6	54	00110110	?	63	00111111
7	55	00110111	@	64	01000000
8	56	00111000	A	65	01000001
9	57	00111001	B	66	01000010
:	58	00111010	C	67	01000011
;	59	00111011	D	68	01000100

Ejemplo



```
char texto[11]; /*declaración*/
texto[0] = 'B';
texto[1] = 'e';
texto[2] = 'm';
texto[3] = '-';
texto[4] = 'v';
texto[5] = 'i';
texto[6] = 'n';
texto[7] = 'd';
texto[8] = 'o';
texto[9] = '!';
texto[10] = '\0';
printf("%s\n",texto);
```

Representación gráfica

'B'	'i'	'e'	'n'	'v'	'e'	'n'	'i'	'd'	'o'	'\0'
0	1	2	3	4	5	6	7	8	9	10



Cadena de caracteres (string)

- Para leer una entrada:
 - **scanf("%s", texto);**
 - Lee una cadena de caracteres hasta encontrar espacio en blanco, nueva línea o **EOF** (fin del archivo).
 - **gets(texto);**
 - Lee caracteres incluyendo espacios en blanco, hasta encontrar una nueva línea en blanco o **EOF**. No es recomendable su uso porque da problemas cuando el texto digitado excede el tamaño del string.
 - **fgets(texto, TAM, stdin);**
 - Igual que **gets** sin embargo es más seguro, lee como máximo **TAM** caracteres.
 - El carácter **'\0'** es insertado al final del arreglo texto después de la lectura en todos los casos.



Cadena de caracteres (string)

- Leyendo una entrada:

```
#include <stdio.h>
#define LIM 500

int main(){
    char texto[LIM];
    printf("Digite uma string: ");
    scanf("%s",texto);
    gets(texto);
    fgets(texto,LIM-1,stdin); /*Más seguro!*/

    printf("texto: %s\n",texto);

    return 0;
}
```



Cadena de caracteres (string)

- Leyendo una entrada patrón (un carácter a la vez):

```
#include <stdio.h>
int main(){
    char texto[500];
    int c,i=0;
    while(1){
        c = getchar(); /*Lee el próximo carácter.*/
        if(c==EOF || c=='\n')
            break;
        texto[i] = (char)c;
        i++;
    }
    texto[i] = '\0';
    printf("texto: %s\n",texto);
    return 0;
}
```



Cadena de caracteres (string)

- Imprimir en un string:

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

```
int main(){  
    char texto[500];  
    float media= 5.5;  
    /*Imprimindo na saída padrão.*/  
    printf("media: %.2f\n",media);  
    /*Guarda en un arreglo*/  
    sprintf(texto,"media: %.2f\n",media);  
    printf("%s",texto);  
  
    return 0;  
}
```




Cadena de caracteres (string)

- Manipulación de strings: `#include <string.h>`

```
#include <stdio.h>
#include <string.h>
int main(){
    char firstname[100]="Hans";
    char lastname[100]="Ccacyahuillca";
    char name[100];
    printf("%d\n",strlen(firstname)); /* Imprime 4. */
    printf("%d\n",strlen(lastname)); /* Imprime 13. */
    strcpy(name, firstname); /*Copia firstname.*/
    strcat(name, " "); /*Adiciona Espacio en blanco.*/
    strcat(name, lastname); /*Adiciona lastname.*/
    printf("%d\n",strlen(name)); /*Imprime 18.*/
    printf("name: %s\n",name); /*Imprime nombre completo.*/
    return 0;
}
```



Cadena de caracteres (string)

- Una posible implementación de **strlen** para ASCII:

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

```
int main(){  
    char texto[]="Hans Ccacyahuillca";  
    int i;  
    i = 0;  
    while(texto[i]!='\0')  
        i++;  
    printf("String posee %d caracteres.\n",i);  
  
    return 0;  
}
```



Cadena de caracteres (string)

- Cuenta vocales:

```
int cuentaVocales (byte s[]) {  
    byte *vocales;  
    vocales = "aeiouAEIOU";  
    int numVocales = 0;  
    for (int i = 0; s[i] != '\0'; ++i) {  
        byte ch = s[i];  
        for (int j = 0; vocales[j] != '\0'; ++j) {  
            if (vocales[j] == ch) {  
                numVocales += 1;  
                break;  
            }  
        }  
    }  
    return numVocales;  
}
```



Cadena de caracteres (string)

- Tamaño de un string:

```
unsigned int strlen (char *s) {  
    int k;  
    for (k = 0; s[k] != '\0'; ++k) ;  
    return k;  
}
```



Cadena de caracteres (string)

- Copiar string:

```
void strcpy (char *s, char *t) {  
    int i;  
    for (i = 0; t[i] != '\0'; ++i)  
        s[i] = t[i];  
    s[i] = '\0';  
}
```

- Otras formas:

```
char a[8], b[8];  
strcpy (a, "palta"); strcpy (b, "banana");
```

```
char *a = malloc(8), *b = malloc(8);  
strcpy (a, "palta"); strcpy (b, "banana");
```

```
char *a, *b;  
a = "palta"; b = "banana";
```



Cadena de caracteres (string)

- Comparar strings:

```
int strcmp (char *s, char *t) {  
    int i;  
    for (i = 0; s[i] == t[i]; ++i)  
        if (s[i] == '\0') return 0;  
    unsigned char si = s[i], ti = t[i];  
    return si - ti;  
}
```



Ejercicio resuelto sobre funciones y strings

- Invertir string:

```
void InvertirString(char str[]) {  
    int i,j;  
    char tmp;  
  
    j = strlen(str) - 1;  
    i = 0;  
  
    while (i < j) {  
        tmp = str[i];  
        str[i] = str[j];  
        str[j] = tmp;  
        i++;  
        j--;  
    }  
}
```



Ejercicio 1

- Haga un programa que reciba una línea de texto como entrada, y que lea un carácter a la vez, y que dé como resultado las **secuencias continuas** de caracteres **no blancos**, uno por línea.

Ejemplo:

Entrada:

El proveedor 123 ofrece acceso!

Salida:

**El
proveedor 123
ofrece
acceso!**



Ejercicio 2

- Haga un programa que cuente el número de palabras de un texto terminado con la palabra “end”. Ejemplo:

Entrada:

un programa que lee las
noticias en un formato especial
llamado CML end

Salida:

13 palabras



Ejercicio 3

- Haga un programa que lea una frase e imprima la frase usando sólo letras mayúsculas:

Entrada:

noticias en un formato especial

Salida:

NOTICIAS EN UN FORMATO ESPECIAL



Ejercicio 4

- Dado dos strings **a** y **b**, verifique cuántas son las repeticiones del string **b** dentro de **a** :

Entrada:

a = “una gallina pinta, piperipinta, piperialegre”

b = “pi”

Salida:

6



Ejercicio resuelto 1

Haga un programa que lea una línea de texto en un string (secuencia de caracteres terminado en '\0') y que cuente cuantas veces se repite cada letra del alfabeto.



Ejercicio resuelto 2

Implementar la función invertir string sin usar la función `<string.h>`.

Ejemplo: Si inicialmente se tiene:

`str = "PAOLO GUERRERO"`, como resultado tenemos:

`"ORERREUG OLOAP"`



Ejercicio resuelto 3

Genera un **dest** una copia de **orig** pero eliminando el exceso de espacios en blanco, de modo que todas las secuencias de espacios en blanco consecutivos son convertidos en un único carácter de espacio. Asumir que el vector **dest** es grande y suficiente para contener el resultado.

Ejemplo:

Orig = " Problema de Strings "

dest = " Problema de Strings "



Ejercicio resuelto 4

Busca en **orig** por posibles candidatos de direcciones de e-mail. Copia en **dest** la primera secuencia de caracteres no blancos encontrada que contenga el carácter “@”, o un string vacío caso contrario, o un string vacío caso contrario. Asumir que el vector **dest** es lo suficientemente grande para encontrar la palabra encontrada.

Ejemplo:

orig = “Enviar mensaje a 827393@unsaac.edu.pe o para otro asesor”

dest = “827393@unsaac.edu.pe”