# Cadenas de Caracteres

# ¿Qué son las cadenas de cadenas de caracteres?

-Es la forma en la que podemos ingresar un texto en programación.

-En C no existe el tipo de dato para cadenas (string), por ende una cadena de caracteres es simplemente un array de Char(Un caracter).

-Al definir una palabra en C, se debe guardar un especio más para el ‘\0’, que es un caracter especial que indica el fin de dicha cadena.

-Por ejemplo para escribir la palabra “hola” el vector deberia ser cómo mínimo [5] en vez de [4] ya que sino no habria espacio para el ‘\0’

# Ejemplos

-char saludo[10]="hola"; => Ocuparia 5 espacios de los 10 reservados

-char saludo[10]={'h','o','l','a','\0'};

-char saludo[ ]="hola";

-char menu [ ][3]={"Archivo","Nuevo","Abrir"}; => [FIL] [COL]

Cuando se declara un vector sin ningun número [ ] significa que se reserva automáticamente el número de bytes necesarios para la cadena, distinto de que si declaro [31] significa que la cadena tiene si o si 30 caracteres (Incluyendo que el nro 31 es el '\0')

# Guardar Cadena en Variable

Hasta el momento estuvimos guardando int/char/float en variables pidiendolas por un scanf (usando su respectiva mascara). En el caso de las cadenas podemos usar scanf, con la mascara %s y sin usar el & en la variable.

Pero esto tendria un efecto negativo ya que sólo podriamos guardar una sóla palabra, ya que al apretar un espacio la entrada de datos se corta y solo imprime la primer palabra.

O sea cuando a scanf se le pasa un espacio o un enter termina con su funcionamiento.

# Gets()

Para solucionar este problema usamos la función gets, que es más apta para este tipo de datos y guarda los espacios como si fuera otro caracter, en vez de terminar su funcionamiento. Una vez que ocurra un Enter, la función termina y guarda en la cadena pasada, la información que el usuario haya escrito.

```
char texto[TAM];

fflush(stdin);

printf("\nIngrese nombre: ");

gets(texto);

printf("\nNombre: %s ",texto);
```

# Fgets()

Uno de los problemas que tenemos al usar la función gets(), es que está sólo disponible para Windows, por ende para los usuarios de Linux o en GDB no podemos usarla.

Una de las soluciones es usar ***Fgets()*** que posee un funcionamiento exactamente igual a gets(), pero está creada para el manejo de archivos, pese a esto, no es necesario crear/abrir un archivo para usarla, por ende para los usuarios de Linux/GDB es una de las soluciones.

Al realizar un strlen en un Fgets, se le suma un caracter más, ya que el fgets, agrega espacio carácter más por el \n

```
char texto[TAM];

fflush(stdin);

printf("\nIngrese nombre: ");

fgets(texto, TAM, stdin);

printf("\nNombre: %s ",texto);
```

# Scanf()

Pese a lo que mostramos anteriormente, la función fgets es creada para el uso de archivos, no es un uso correcto cuando no se este manejando con ellos. Por ende existe una alternativa, ya mencionada anteriormente y es usando scanf(). ¿Pero si queremos usar espacios, cómo hacemos? , existe una solución y es ignorando el \n

***scanf("%[^\n]", cadena);***

Con esto logramos que cuando toquemos un espacio no se produzca un salto de línea, y por ende que el scanf no se corte con un espacio.

```
char texto[TAM];

fflush(stdin);

printf("\nIngrese nombre: ");

scanf("%[^\n]", texto);

printf("\nNombre: %s ",texto);
```

# Funciones usadas para Cadenas de Caracteres

# STRLEN

Lo que hace strlen es devolver la longitud de la cadena de texto que le pasemos como parametro. O sea, la cantidad de caracteres de dicha cadena, contando desde el primer caracter hasta el (\0).

Por lo general la usamos para validar el largo de la cadena

int cantidadCaracteres=**strlen**(“Hola Mundo”);

cantidadCaracteres valdra 10 ya que los espacios son contados.

# STRCPY

Lo que hace strcpy es copiar una cadena de un lugar a otro.

**strcpy**(destino,origen)

La función copia la cadena de caracteres que se encuentra guardada en un vector (origen) hacia otro vector (destino).

El destino tiene que ser si o si una variable, y tiene que tener la misma capacidad como minimo

# STRCPY

Imaginemos que tenemos dos variables, una llamada texto1 y otra texto2.

A **texto1** la cargue con el valor “**Hola**” y a **texto2** sin nada, para copiar el valor de **texto1** a **texto2** hacemos lo siguiente

**strcpy**(texto2 , texto1);

Al llamar a **strcpy** texto2 va a valer lo mismo que texto1 o sea “Hola”.

# STRCMP

**strcmp** compara cadenas de caracteres retornando un valor entero segun su comparación.

**int strcmp (cadena1 , cadena2)**

-Si la función devuelve menor a cero, significa que **cadena1** es menor a **cadena2**

-Si la función devuelve cero significa que **cadena1** y **cadena2** son la misma cadena.

-Si la función devuelve mayor a cero significa que la **cadena1** es mayor a la **cadena2**

# STRICMP

**Su función es la misma que STRCMP sólo que está no respeta las mayúsculas ni minúsculas.**

**int stricmp (cadena1 , cadena2)**

-Si la función devuelve menor a cero, significa que **cadena1** es menor a **cadena2**

-Si la función devuelve cero significa que **cadena1** y **cadena2** son la misma cadena.

-Si la función devuelve mayor a cero significa que la **cadena1** es mayor a la **cadena2**

# STRLWR/STRUPR

-Lo que hace **strlwr** es convertir todas las mayúsculas de una cadena en minúsculas.

**strlwr(cadena)**

-Lo que hace **strupr** es convertir todas las minúsculas de una cadena en mayúsculas .

**strupr(cadena)**

# STRCAT

**strcat** concatena en la cadena1 la cadena2.

***char* strcat(cadena1 , cadena2)***

El valor de **cadena1** luego del **strcat** es la cadena concatenada.

# Algunas Funciones importantes para Char

# TOLOWER/TOUPPER

-Lo que hace **tolower** es convertir un caracter a minúscula

***tolower(letra)***

-Lo que hace **toupper** es convertir un caracter a mayúscula

***toupper(letra)***

# ISDIGIT/ISALPHA/ISSPACE

Existen unas funciones para verificar que el caracter le estoy mandando concuerde con algún patrón en especifico.

Para poder usarlas debemos incluir la libreria <ctype.h>

-***Isdigit()*** me indica si el char que le paso es un númerico o no retornandome 1 en caso de que sea y 0 si no lo es, por ejemplo.

***isdigit('a')*** => Me retorna 0

***isdigit('+')*** => Me retorna 0

***isdigit('4')*** => Me retorna 1

# ISDIGIT/ISALPHA/ISSPACE

-***Isalpha()*** me indica si el char que le paso es una letra (a/z A/Z) o no retornandome 1 en caso de que sea y 0 si no lo es, por ejemplo.

***Isalpha('a')*** => Me retorna 1

***Isalpha('5')*** => Me retorna 0

-isspace***()*** me indica si el char que le paso es un espacio en blanco o no retornandome 1 en caso de que sea y 0 si no lo es, por ejemplo.

***isspace(' ')*** => Me retorna 1

***isspace('p')*** => Me retorna 0

***isspace('\n')*** => Me retorna 1

# ISDIGIT/ISALPHA/ISSPACE

Se denominan espacios en blanco a los siguientes caracteres.

' ' = espacio

'\n' = salto de linea

'\t' = tab horizontal

'\v = tab vertical

'\f' Salto de página

'\r' Retorno de carro