

# Cadenas en C/C++

Mag. Ing. Nancy López

# Cadenas

- Una cadena de caracteres (string) es un conjunto de caracteres (incluido el espacio blanco) que se almacenan en localidades contiguas de memoria.
- Se representa como un vector de caracteres donde cada elemento del vector representa un caracter de la cadena.
- Ejemplo
- `char nombre [16];`

'D'	'O'	'N'	'A'	'L'	'D'	' '	'K'	'N'	'U'	'T'	'H'	'I'	'O'		
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	--	--

# Cadenas

- Tenga en cuenta que una cadena de  $n$  caracteres requerirá un vector de  $n+1$  elementos, debido al carácter nulo `'\0'` que se añade automáticamente al final de la cadena.

# Cadenas - Declaración

- `char nombre [TAM];`
- Donde TAM= cantidad máxima de caracteres +1
- Ej.
- `char direccion[100];`
- `char cedula[9];`

# Cadenas - Inicialización

- Se puede inicializar la cadena de caracteres al declararla:
- `char nombre={'M', 'a', 'r', 'i', 'a', '\0'};`
- `char saludo[10]="Hola";`
- `char palabra[]= "Hola";`
- En este último caso, el vector toma como tamaño la cantidad de caracteres de la palabra + 1.

# Cadenas - Acceso

- Para tener acceso a los elementos de una cadena se utiliza un subíndice.

- Ejemplo:

```
palabra[0] = 'H' ;  
palabra[1] = 'o' ;  
palabra[2] = 'l' ;  
palabra[3] = 'a' ;  
palabra[4] = '\0' ;
```

# Cadenas – Lectura y escritura

- Biblioteca stdio.h
- `gets(cadena);` //lee la cadena
- `puts(cadena);` //muestra la cadena
- `gets()` es específico para leer cadenas ya que si tiene algún espacio en blanco, cin lee sólo hasta el espacio, ignorando el resto.
- `gets()` lee hasta el enter inclusive (`\0`).

# Cadenas – Lectura y escritura

- Biblioteca stdio.h
- `gets(cadena);` //lee la cadena
- `puts(cadena);` //muestra la cadena
- `puts ()` muestra la cadena y hace un enter.
- Se puede mostrar con un `cout` si no se quiere un enter luego de mostrar la cadena.



# Cadenas – Matrices

- Se pueden declarar matrices de cadenas:
- `char nombres[FIL][COL];`
- FIL indica la cantidad de nombres.
- COL indica la cantidad de caracteres que tiene cada nombre como máximo -1.

# Cadenas – Matrices

- En este caso, cuando lleno una matriz de nombres, sólo indico en **qué fila** lo guardo ya que siempre comenzará a escribir en la primer columna y luego terminará cuando el usuario presione enter.

# Cadenas – Matrices

- Ejemplo de ingreso

```
char nombre[FIL][COL];  
int j;  
for(j=0; j<FIL; j++)  
{  
    cout<<"Ingrese nombre";  
    gets(nombre[j]);  
}
```

# Cadenas – Matrices

- Ejemplo de salida

```
for(j=0; j<FIL; j++)  
{  
    puts(nombre[j]);  
}
```

# Cadenas – Matrices

- Otro ejemplo:
- Ingrese el nombre de 5 alumnos y las notas de dos materias.
- Mostrar los datos por pantalla.

```
const FIL=5, COL=20, MATERIAS=2;
int main()
{
    char nombre[FIL][COL];
    int notas[FIL][MATERIAS];
    int j, k;
    for(j=0; j<FIL; j++)
    {
        cout<<"Ingrese nombre";
        gets(nombre[j]);
        for(k=0; k<MATERIAS; k++)
        {
            cout<<"Ingrese notas materia "<<k+1<<" de "<<nombre[j];
            cin>>notas[j][k];
        }
    }
}
```

```
cout<<"Nombre\tnota1\tnota2"<<endl;
for(j=0;j<FIL; j++)
{
    cout<<nombre[j]<<"\t";
    for(k=0; k<MATERIAS; k++)
    {
        cout<<notas[j][k]<<"\t";
    }
    cout<<endl;
}
return 0;
}
```

# Cadenas – String.h

- La biblioteca string.h contiene una serie de funciones que permiten la manipulación de cadenas, entre las más usadas tenemos:

**strcpy(cadenaDestino, cadenaOrigen);**

- Copia en la cadena Destino, la cadena Origen.
- Recordar que la asignación es siempre de derecha a izquierda.



# Cadenas – String.h

`strcmp(cad1, cad2)` //es case sensitive

`stricmp(cad1, cad2)` //no es case sensitive

- Realiza una serie de operaciones a nivel de código ASCII y devuelve un valor igual a cero si ambas cadenas son alfabéticamente iguales.
- Devuelve un valor menor que cero si la cadena 1 es alfabéticamente menor que la cadena 2.
- Devuelve un valor mayor que cero si la cadena 1 es alfabéticamente mayor que la cadena 2.

# Cadenas – String.h

**strcmp(cad1, cad2)**

**stricmp(cad1, cad2)**

- Ejs.
- `stricmp("Juan","Juan")` devuelve 0
- `stricmp ("Juan", "juan")` devuelve 0
- `stricmp("Juan", "Ana")` devuelve > 0
- `stricmp("Ana", "Juan")` devuelve <0
- Es especialmente útil para ordenar cadenas alfabéticamente.

# Cadenas – String.h

**strcat(cad1, cad2);**

- Concatena la cadena 2 a continuación de la cadena 1.

# Cadenas – String.h

**strlen(cadena)**

- Devuelve la longitud de la cadena en caracteres.

# Cadenas – String.h

## **strrev(cadena)**

- Invierte una cadena excepto el carácter de terminación de la cadena (\0).

# Cadenas – String.h

## **strlwr(cadena)**

- Pasa las mayúsculas de cadena a minúsculas.
- Ej.
- `cout<<strlwr("Ana"); //muestra ana`
- `cout<<strlwr("ANA"); //muestra ana`

# Cadenas – String.h

## **strupr(cadena)**

- Pasa las minúsculas de cadena a mayúsculas.
- Ej.
- `cout<<strupr("Ana"); //muestra ANA`
- `cout<<strupr("ana"); //muestra ANA`

# Cadenas – Ctype.h

- Contiene funciones que se usan para caracteres individuales.
- Devuelven un cero para falso y un valor distinto de cero para verdadero.



# Cadenas – Ctype.h

- `isalnum()` si es alfanumérico.
- `isalpha()` si es alfabético.
- `isascii()` si es un valor ascii (0-127)
- `iscntrl()` si es un caracter de control.
- `isdigit()` si es un dígito.
- `isgraph()` si es un caracter imprimible excepto el espacio.
- `islower()` si es minúscula.

# Cadenas – Ctype.h

- `isprint()` si es imprimible, incluye el espacio.
- `ispunct()` si es un símbolo de puntuación.
- `isspace()` si es un espacio.
- `isupper()` si es mayúscula.
- `isxdigit()` si es un carácter hexadecimal.
- `toascii()` pasa el carácter a formato ascii.
- `tolower()` pasa el carácter a minúscula.
- `toupper()` pasa el carácter a mayúscula.