



UNIVERSIDAD NACIONAL DE ASUNCIÓN  
**FACULTAD DE  
INGENIERÍA**

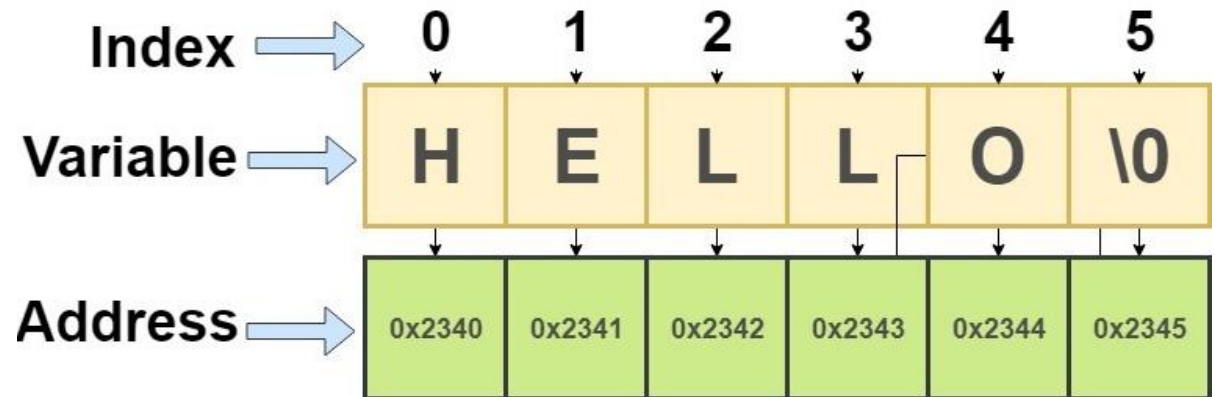
# Cadenas en C++

Cátedra de Computación

• **2020** •

# ¿Qué veremos?

- Definición de cadenas en C++
  - Lectura de cadenas
- Funciones básicas en cstring (o string.h)
- Implementación de funciones básicas
- Manipulación de caracteres
- Ejercicios



# Cadenas en C/C++

En C no existe el tipo de dato “cadena” (**string** en inglés). Pero existe una convención en la forma de representar una cadena, y es a través de un **arreglo de caracteres**. El fin de una cadena se indica con el carácter especial ‘\0’.

En C++ existe el **string**, que en realidad es una clase que internamente hace básicamente lo mismo (con características adicionales).

‘H’	‘e’	‘l’	‘l’	‘o’	‘\0’
-----	-----	-----	-----	-----	------

Esta cadena se declara de las siguientes maneras:

```
char a[6]={'h','e','l','l','o','\0'};  
char b[6]="hello";
```

# Cadenas en C/C++

El vector de caracteres puede tener un tamaño mayor que la cantidad de caracteres de la cadena. Aún así, el carácter especial ‘\0’ es el que determina el final de una cadena.

Si declaramos así:

```
char a[8]={ 'h', 'e', 'l', 'l', 'o', '\0' };  
char b[8]="hello";
```

‘h’	‘e’	‘l’	‘l’	‘o’	‘\0’	?	?
-----	-----	-----	-----	-----	------	---	---

Podrían almacenar cualquier carácter, pero como están posicionados luego del ‘\0’, son irrelevantes

# Cadenas vs Caracteres

Como se mencionó anteriormente, las cadenas son arreglos unidimensionales (vectores) de caracteres. Los caracteres se expresan con comillas simples `' '`, mientras que las cadenas con comillas dobles `" "`.

Un caracter:

```
char c = 'a' //ocupa 1 byte
```

Una cadena:

```
char cad[] = "a" //ocupa 2 bytes
```

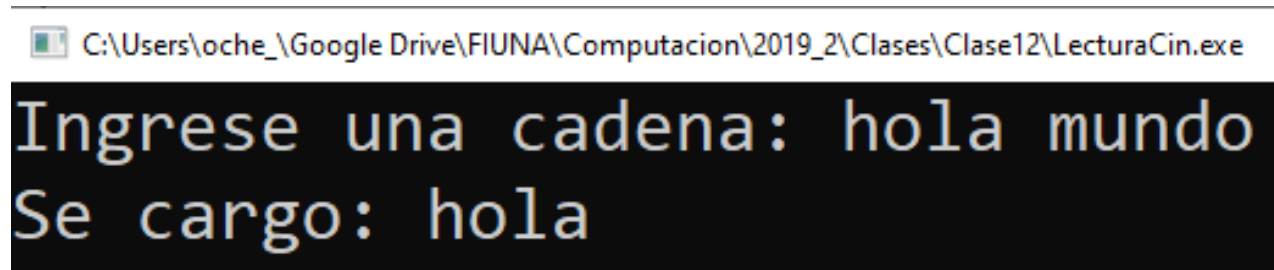
Memoria



# Cadenas – Lectura e impresión

**Se puede usar cin para leer cadenas, y cout para mostrarlas en pantalla:**

```
char mensaje[20];  
cin>>mensaje; /*Qué pasa si cargamos hola mundo?*/  
cout<<mensaje<<endl;
```



```
C:\Users\oche_\Google Drive\FIUNA\Computacion\2019_2\Clases\Clase12\LecturaCin.exe  
Ingrese una cadena: hola mundo  
Se cargo: hola
```

# Cadenas – Lectura con `cin.getline`

La función `cin.getline()` se puede usar para leer una línea completa de entrada y guardarla en una cadena (vector de caracteres).

Su sintaxis es la siguiente:

```
cin.getline(cadena, tam_max, delimitador)
```

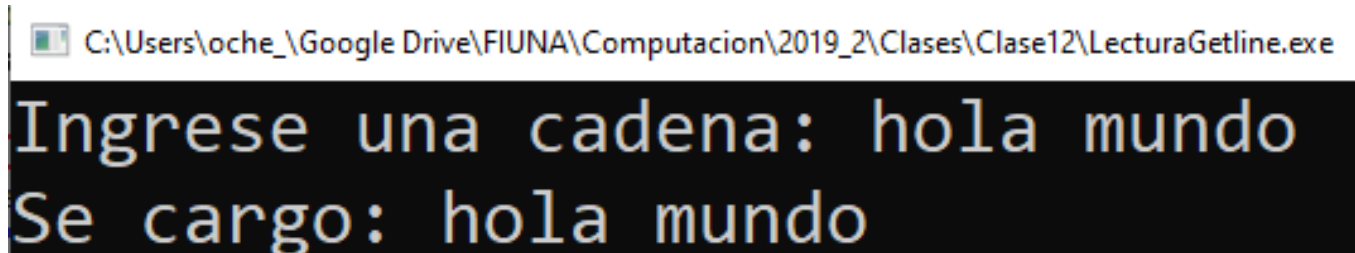
Podemos ver que tiene tres parámetros:

- `cadena`: es el identificador del vector de caracteres en donde se guardará la línea ingresada por teclado.
- `tam_max`: especifica el número máximo de caracteres, incluido el carácter de fin de cadena.
- `delimitador`: este parámetro es **opcional**. Indica el carácter de terminación; si no está incluido, `getline` se detiene en “\n” (es decir, el enter).

# Cadenas – Lectura con `cin.getline`

Para nuestro ejemplo anterior:

```
char mensaje[20];  
cin.getline(mensaje,20); /*cargamos hola mundo de vuelta*/  
cout<<mensaje<<endl;
```



```
C:\Users\oche_\Google Drive\FIUNA\Computacion\2019_2\Clases\Clase12\LecturaGetline.exe  
Ingresa una cadena: hola mundo  
Se cargo: hola mundo
```




# Cadenas – Lectura con `cin.getline`

¿Qué hace el siguiente programa? (Supongamos que ingresamos “Colbes,Jose”)

```
#include<iostream>
using namespace std;

int main(){
    char nombre[30], apellido[30];
    cout<<"Ingrese su <apellido,nombre>: ";
    cin.getline(apellido,sizeof(apellido),' ');
    cin.getline(nombre,sizeof(nombre));
    cout<<"Nombre apellido:\n|"<<nombre<<"\t"<<apellido<<"|\n";
    return 0;
}
```

 C:\Users\oche\_\Google Drive\FIUNA\Computacion\2019\_2\Clases\Clase12\LecturaGetlineEjemplo.exe

```
Ingrese su <apellido,nombre>: Colbes,Jose
Nombre apellido:
|Jose    Colbes|
```

# Cadenas en C/C++ – Funciones

```
#include<cstring>
```

ó

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

## Algunas funciones útiles de cadenas

- `strcpy(cad1, cad2);` → copia la cadena `cad2` a la `cad1`.
- `strcat(cad1, cad2);` → concatena las cadenas `cad1` y `cad2` (el resultado queda en `cad1`).
- `int a=strlen(cad1);` → devuelve la cantidad de caracteres de la cadena `cad1` (sin contar el `\0`).
- `int a=strcmp(cad1, cad2);` → devuelve un valor que representa la diferencia entre las dos cadenas.

**Ejercicios:** escribir funciones que implementen cada una de las presentadas (llamarlas de otra forma).

# Copiar cadenas - **strcpy**

Para copiar una cadena a otra, se puede hacer lo siguiente:

**strcpy**(destino, origen); //ambas son cadenas (la primera debe ser variable)

**strcpy**(name1, "Chan Tai Man");

name1

C	h	a	n		T	a	i		M	a	n	\0	?	?	?
---	---	---	---	--	---	---	---	--	---	---	---	----	---	---	---

**strcpy**(name2, "9999999999999999");

name2

9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	\0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----

**strcpy**(name2, name1);

name2

C	h	a	n		T	a	i		M	a	n	\0	9	9	\0
---	---	---	---	--	---	---	---	--	---	---	---	----	---	---	----


# Longitud de una cadena - `strlen`

La siguiente función:

`strlen(cadena)`

Devuelve el tamaño de una cadena (cantidad de caracteres, exceptuando el de finalización)

```
int longitud;  
longitud = strlen("abcde");
```



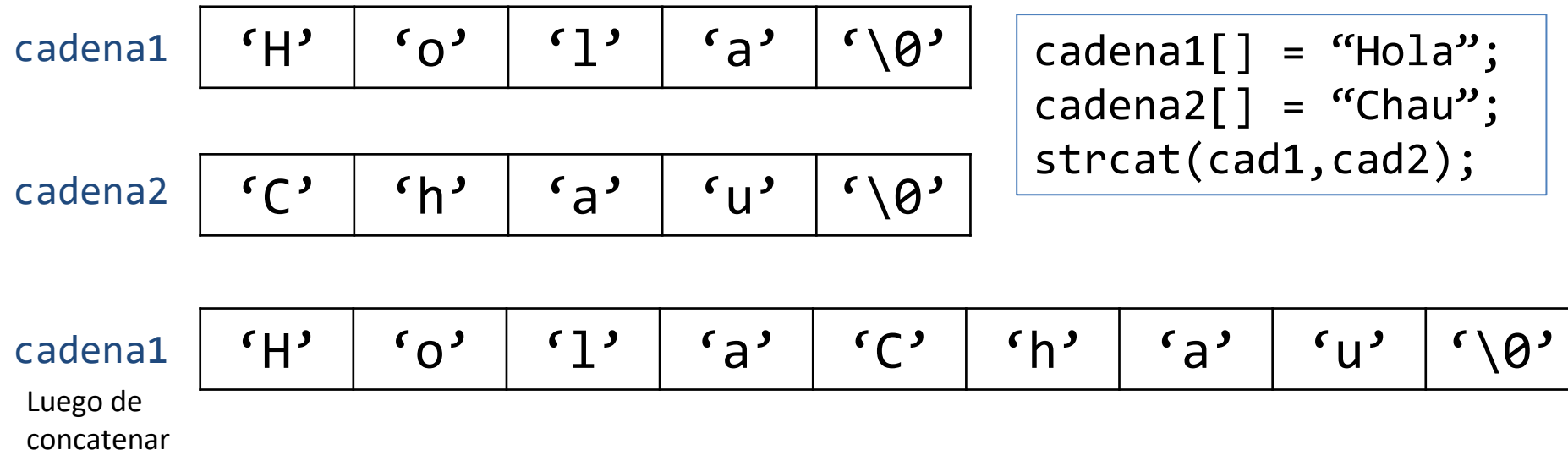
La función devuelve 5  
en este caso

# Concatenar cadenas - `strcat`

La función:

```
strcat(cadena1, cadena2);
```

Une (concatena) el contenido de la `cadena1` con el de la `cadena2` (ignorando el `'\0'` de `cadena1`); guardando el resultado en `cadena1`.



# Tipo **char**

`char b='a';` ó `char b=97;`

Nota: no hace falta memorizarse los valores de los caracteres!!

ASCII printable characters								
DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	\$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_	theASCIICode.com.ar		

Extended ASCII characters											
DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo
128	80h	Ç	160	A0h	á	192	C0h	Ł	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	ł	225	E1h	ô
130	82h	é	162	A2h	ó	194	C2h	Ť	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	Ŧ	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	—	228	E4h	ö
133	85h	à	165	A5h	Ñ	197	C5h	†	229	E5h	Õ
134	86h	á	166	A6h	ª	198	C6h	‡	230	E6h	μ
135	87h	ç	167	A7h	º	199	C7h	Ä	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	ℒ	232	E8h	ƒ
137	89h	ë	169	A9h	®	201	C9h	ℓ	233	E9h	Ů
138	8Ah	è	170	AAh	¬	202	CAh	℥	234	EAh	Ù
139	8Bh	ì	171	ABh	½	203	CBh	₣	235	EBh	Ú
140	8Ch	ï	172	ACH	¼	204	CCh	₤	236	ECh	ý
141	8Dh	ì	173	ADh	»	205	CDh	=	237	EDh	ÿ
142	8Eh	Ä	174	A Eh	«	206	CEh	₥	238	EEh	—
143	8Fh	Å	175	AFh	»	207	CFh	₦	239	EFh	·
144	90h	É	176	B0h	⋮	208	D0h	ø	240	F0h	
145	91h	æ	177	B1h	⋮	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	⋮	210	D2h	Ê	242	F2h	—
147	93h	ô	179	B3h	⋮	211	D3h	Ë	243	F3h	¾
148	94h	ò	180	B4h	⋮	212	D4h	È	244	F4h	¶
149	95h	ó	181	B5h	⋮	213	D5h	Ì	245	F5h	§
150	96h	ù	182	B6h	⋮	214	D6h	Í	246	F6h	÷
151	97h	û	183	B7h	⋮	215	D7h	Î	247	F7h	
152	98h	ÿ	184	B8h	⋮	216	D8h	Ï	248	F8h	ˆ
153	99h	Ö	185	B9h	⋮	217	D9h	Ĵ	249	F9h	˙
154	9Ah	Ü	186	BAh	⋮	218	DAh	⋮	250	FAh	˘
155	9Bh	ø	187	BBh	⋮	219	DBh	⋮	251	FBh	¹
156	9Ch	£	188	BCb	⋮	220	DCb	⋮	252	FBh	º
157	9Dh	Ø	189	BDh	⋮	221	DDh	⋮	253	FDh	²
158	9Eh	×	190	BEh	⋮	222	DEh	⋮	254	FEh	³
159	9Fh	ƒ	191	BFh	⋮	223	DFh	⋮	255	FFh	⁴

# Comparar cadenas - `strcmp`

La función:

`strcmp(cadena1, cadena2)`

Devuelve un entero, producto de la comparación de las cadenas. Sus posibles valores son:

- Si es cero, entonces las cadenas son iguales.
- Si es  $>0$ , entonces  $\text{cadena1} > \text{cadena2}$
- Si es  $<0$ , entonces  $\text{cadena1} < \text{cadena2}$

`strcmp("horca", "horario")`

Esto dará un valor igual mayor que cero, pues 'c' está después que 'a' en la tabla ASCII

# Comparar cadenas - `strcmp`

Algunos ejemplos:

<code>cad1</code>	<code>cad2</code>	<code>retorna</code>	<code>motivo</code>
"AAAA"	"ABCD"	<0	'A' < 'B'
"B123"	"A089"	>0	'B' > 'A'
"127"	"409"	<0	'1' < '4'
"abc888"	"abc888"	=0	Cadenas iguales
"abc"	"abcde"	<0	cad1 es subcadena de cad2
"3"	"12345"	>0	'3' > '1'



# Implementación de estas funciones

```
void copia(char cad1[], char cad2[]){
    int i=0;
    while(cad2[i]!='\0'){
        cad1[i]=cad2[i];
        i++;
    }
    cad1[i]='\0';
}
```

```
void concatenar(char cad1[], char cad2[]){
    int i=0;
    while(cad1[i]!='\0') i++;
    int j=0;
    while(cad2[j]!='\0'){
        cad1[i]=cad2[j];
        i++;j++;
    }
    cad1[i]='\0';
}
```

# Implementación de estas funciones

```
int longitud(char cad1[]){  
    int i=0;  
    while(cad1[i]!='\0') i++;  
    return i;  
}
```

```
int comparar(char cad1[], char cad2[]){  
    int compararCadenas(char cad1[], char cad2[]){  
        int i=0;  
        while((cad1[i]!='\0')&&(cad1[i]==cad2[i])) i++;  
        if(cad1[i]>cad2[i]) return 1;  
        else if (cad1[i]<cad2[i]) return -1;  
        else return 0;  
    }  
}
```

# Algunos errores comunes

- No se puede asignar un valor a una variable de cadena (excepto en la declaración).

```
char A_string[10];  
A_string = "Hola" // da error
```

```
//Se debe usar strcpy  
strcpy (A_string, "Hello");
```

# Algunos errores comunes

- El operador == no funciona para comprobar la igualdad de cadenas

```
if (string1 == string2) //error lógico!
```

```
    cout << "Son iguales!";
```

```
// Debería usarse:
```

```
if (strcmp(string1,string2)==0)
```

```
    cout << "Son iguales!";
```

```
// Recordar que strcmp retorna 0 si las cadenas son iguales
```

# Cadenas – Conversión

```
#include<cstdlib>
```

- Cadena a entero (atoi):

```
char cad[] = "12345";  
int a = atoi(cad); // a <-- 12345
```

- Entero a cadena (itoa):

```
int a = 12345;  
char cad[10];  
itoa(a,cad,10); //cad <-- "12345"
```

Obs: lleva el 10 porque está en base decimal

# Cadenas – Conversión

```
int cadAnum(char s[])
{
    int i=0, n=0, signo=1;
    while(s[i]==' ') i++; //se eliminan los espacios en blanco
    if ((s[i]=='+' || (s[i]=='-')){ //se toma el signo
        if(s[i] == '-') signo=-1;
        i++;
    }
    while(s[i]!='\0'){
        n=10*n +(s[i]-'0');
        i++;
    }
    return (signo*n);
}
```

'1'	'2'	'3'	'4'	'\0'
-----	-----	-----	-----	------

→ 1234

# Cadenas – Conversión

```
void numAcad(char s[], int n){
    int i=0,d=1,j;
    if(n<0){
        s[i]='-';
        i++;
        n=n*(-1);}
    int aux=n;
    while(aux>=10){ //se cuenta el numero de cifras
        d++;
        aux/=10;}
    for(j=(d+i-1);j>=i;j--){
        s[j]=(n%10)+'0';
        n/=10;}
    s[i+d]='\0';
}
```

1234



'1'	'2'	'3'	'4'	'\0'
-----	-----	-----	-----	------

# Cadenas – Conversión

**Ejercicio 1:** Pasar todas las letras de una cadena a mayúsculas.

**Desafío:** convertir una cadena a un punto flotante (número decimal) y viceversa.

