

Caracteres y cadenas de catacteres: Char, string y código ascii

OMI Yucatán - Alonso Huerta

¡Tengan su editor abierto para experimentar!

Tipos primitivos

- Aritméticos
 - Enteros (int, long long, etc.)
 - Flotantes (double, float, etc.)
- Caracteres y cadenas de caracteres
- Booleanos

'a' // character

"Hello, World!" // cadena de caracteres

```
char car1 = '1';
```

```
char car2 = 'a';
```

```
char car3 = 'A';
```

```
char car4 = '@';
```

```
char car5 = car1;
```

```
char car6 = '\n';
```

Escape Sequences

- **newline** \n
- **horizontal tab** \t
- alert (bell) \a
- vertical tab \v
- backspace \b double quote \"
- **backslash** \\
- **question mark** \?
- **single quote** \'
- carriage return \r
- formfeed \f

```
char single_quote = '''; // error
```

```
char single_quote = '\\''; // el caracter '
```

```
cout << single_quote; // '
```

```
cout << '\n'; // Imprime nueva línea
```

```
cout << "\tHi!\n"; // imprime un tab seguido de "Hi!" y una nueva línea
```


“El código ASCII (siglas en ingles para American Standard Code for Information Interchange, es decir Código Americano) [está hecho para] reordenar y expandir el conjunto de símbolos y caracteres.”

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

```
char car1 = 65;
```

```
cout << car1; // imprime 'A'
```

```
char car2 = 'A' + 3;
```

```
cout << car2; // Imprime 'D'
```

```
#include <iostream>

using namespace std;

int main() {

    char car1 = 'A' + 'a' - '@' + '\t' + 4;
    cout << "El character es: " << car1 << endl;

    return 0;
}
```

```
// Para obtener el valor en decimal de un caracter
```

```
char car = 'A';
```

```
// Almacenalo en una variable tipo entero
```

```
int entero = car;
```

```
cout << entero; // 65
```

```
// Agrega (int) antes de la variables
```

```
cout << (int)car; // 65
```

¿Cómo represento una palabra?

```
char cadena_de_caracteres[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

```
char otra_cadena_de_caracteres[] = ", World!";
```

```
cout << cadena_de_caracteres << otra_cadena_de_caracteres; // imprime "Hello, World!"
```

Strings

```
#include <string>
```


string me permite...

- Juntar palabras
- Saber el tamaño de la palabra
- Poder cambiar caracteres únicos
- Comparar palabras (con operadores como ==, <= y !=)
- ¡Muchas cosas más!

```
string s1; // una palabra vacía  
string s2 = "Hello, World!"; // la palabra "Hello, World!"  
string s3 = s2; // una copia de la palabra s2  
string s4 = "Hey! " + s3 + '!'; // "Hey! Hello, World!!"
```

```
string s1 = "esto es una palabra";  
s1[0] = 'E';  
s1[12] = 'P';  
cout << s1; // "Esto es una Palabra"
```

```
#include <iostream>
#include <string>

using namespace std;

int main(){

    string pato = "P470";
    string pez = "P32";

    pato[1] = 'a';
    pez[2] = 'z';

    string oracion = pato + " tiene un " + pez;

    cout << oracion << endl;

    return 0;
}
```

Funcionalidades de string

```
str.size(); // Regresa el tamaño de str
str.length(); // Regresa el tamaño de str
str.clear(); // Vacía str
str.empty(); // Regresa true si str es una palabra vacía, false si no lo es
str.front(); // Regresa el primer caracter de str
str.back(); // Regresa el último caracter de str
str += otro_str; // Agrega el valor de otro_str al final de str
str.append(otro_str); // Agrega el valor de otro_str al final de str
str.push_back(caracter); // Agrega el valor de caracter al final de str
// Más en https://cplusplus.com/reference/string/string/
```

```
#include <iostream>
#include <string>

using namespace std;

int main(){

    string str = "OMIYUC";
    str.push_back('!');

    for(int i = 0; i < str.size(); i++){
        cout << str[i] << '_';
    }

    return 0;
}
```

**Leyendo caracteres y strings
como input**

```
cin >> character; // leerá el siguiente caracter no vacío
```

```
cin >> str; // leerá todos los caracteres hasta el primer caracter vacío
```

```
// un carater vacío podría ser un salto de línea, un TAB, o un espacio
```



```
// Input
// -----
// Este es
// mi input !
// -----

string str;
char character;

cin >> str >> character;
cout << str << character << endl; // Primero imprimirá "Este" y luego 'e'

cin >> str >> character;
cout << str << character << endl; // Primero imprimirá "s" y luego 'm'

cin >> str >> character;
cout << str << character << endl; // Primero imprimirá "i" y luego 'i'

cin >> str >> character;
cout << str << character << endl; // Primero imprimirá "nput" y luego '!'

// Ouput
// -----
// Estee
// sm
// ii
// nput!
// -----
```

¿Y si quiero leer una línea completa?

¡Fácil!

```
string line;
```

```
// Lee toda la línea hasta el primer salto de línea ('\n')  
getline(cin, line);
```

```
// Input
// -----
// Esta es una línea con espacios
// Palabra
// -----

string line;
string palabra;

// Lee toda la línea hasta el primer salto de línea ('\n')
getline(cin, line);
cin >> palabra;

// Output
// -----
// Esta es una línea con espacios
// Palabra
// -----
```

```
// 0J0!  
  
// Input  
// -----  
// 5  
// Hola, Jorge!  
// -----  
  
int entero;  
string str;  
  
cin >> entero;  
getline(cin, str);  
  
cout << entero << endl;  
cout << str;  
  
// Input  
// -----  
// 5  
//  
// -----
```

```
// OJO!
```

```
// Input
```

```
// -----
```

```
// 5
```

```
// Hola, Jorge!
```

```
// -----
```

```
int entero;
```

```
string str;
```

```
cin >> entero;
```

```
cin.ignore(); // Ignora todos los espacios vacíos
```

```
// hasta llegar al siguiente caracter
```

```
getline(cin, str);
```

```
cout << entero << endl;
```

```
cout << str;
```

```
// Input
```

```
// -----
```

```
// 5
```

```
// Hola, Jorge!
```

```
// -----
```

Table 3.3: ctype Functions

<code>isalnum(c)</code>	true if <code>c</code> is a letter or a digit.
<code>isalpha(c)</code>	true if <code>c</code> is a letter.
<code>iscntrl(c)</code>	true if <code>c</code> is a control character.
<code>isdigit(c)</code>	true if <code>c</code> is a digit.
<code>isgraph(c)</code>	true if <code>c</code> is not a space but is printable.
<code>islower(c)</code>	true if <code>c</code> is a lowercase letter.
<code>isprint(c)</code>	true if <code>c</code> is a printable character (i.e., a space or a character that has a visible representation).
<code>ispunct(c)</code>	true if <code>c</code> is a punctuation character (i.e., a character that is not a control character, a digit, a letter, or a printable whitespace).
<code>isspace(c)</code>	true if <code>c</code> is whitespace (i.e., a space, tab, vertical tab, return, newline, or formfeed).
<code>isupper(c)</code>	true if <code>c</code> is an uppercase letter.
<code>isxdigit(c)</code>	true if <code>c</code> is a hexadecimal digit.
<code>tolower(c)</code>	If <code>c</code> is an uppercase letter, returns its lowercase equivalent; otherwise returns <code>c</code> unchanged.
<code>toupper(c)</code>	If <code>c</code> is a lowercase letter, returns its uppercase equivalent; otherwise returns <code>c</code> unchanged.

```
isdigit('0'); // true  
isdigit('#'); // true
```

```
isalpha('A'); // true  
isalpha('a'); // true  
isalpha('0'); // false
```

```
isupper('Z'); // true  
isupper('z'); // false  
isupper('@'); // false
```

```
toupper('a'); // 'A'  
tolower('A'); // 'a'
```


¡Vamos a practicar!