

Programación 1

Tema 10

Caracteres y cadenas de caracteres



Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza





Índice

- Caracteres
- Cadenas de caracteres



El tipo carácter

- Tipos carácter
 - **Dominio** de valores
 - **Representación** de los valores
 - Externa (en C++)
 - Interna (en la memoria del computador)
 - **Operadores** asociados

Caracteres

Dominio de valores





Caracteres

Posible dominio de valores

- ❑ Letras mayúsculas del alfabeto inglés
- ❑ Letras minúsculas del alfabeto inglés
- ❑ Dígitos
- ❑ Signos de puntuación
- ❑ Signos matemáticos
- ❑ Letras con diacríticos (alfabetos latinos occidentales)
- ❑ Letras alfabetos centro-europeos
- ❑ Letras alfabeto griego
- ❑ Letras alfabeto cirílico
- ❑ Letras alfabetos asiáticos



Caracteres en C++

□ Varios tipos

■ **char**

- 1 *byte* (8 bits) en MinGW y GNU GCC

■ **wchar_t**

- 2 *bytes* (16 bits) en MinGW y GNU GCC

■ **char16_t**

- 2 *bytes* (16 bits), estándar

■ **char32_t**

- 4 *bytes* (32 bits), estándar

Caracteres

- char
- Dominio de valores
 - 95 caracteres
 - Letras del alfabeto inglés
 - Dígitos
 - Signos de puntuación
 - Otros símbolos
 - 33 *caracteres* de control

	0	@	P	`	p
!	1	A	Q	a	q
"	2	B	R	b	r
#	3	C	S	c	s
\$	4	D	T	d	t
%	5	E	U	e	u
&	6	F	V	f	v
'	7	G	W	g	w
(8	H	X	h	x
)	9	I	Y	i	y
*	:	J	Z	j	z
+	;	K	[k	{
,	<	L	\	l	
-	=	M]	m	}
.	>	N	^	n	~
/	?	O	_	o	

Caracteres

□ Representación externa en C++

- 'a' 'A' 'b' 'B' 'z' 'Z'
- '0' '1' '2' '3' '4' '5' '6' '7'
'8' '9'
- '+' '-' '*' '/' '<' '=' '>'
- '(' ')' '[' ']' '{' '}'
- '#' '\$' '%' '&' ',' '.' ':' ';'
'!' '?' '@' '^' _ ` ' | '~'
'_'
- '"' '\'



Representación interna

- Codificación arbitraria en binario
 - Código ASCII
 - *American Standard Code for Information Interchange*
 - Estandarizada por la American Standards Association en 1963
- Ejemplo: 'A' se codifica con
 - la secuencia binaria 0100 0001
 - el código numérico 65



Representación interna

Código	Carácter	Código	Carácter	Código	Carácter	Código	Carácter	Código	Carácter	Código	Carácter	Código	Carácter	Código	Carácter	Código	Carácter
0	NUL	16	DLE	32		48	0	64	@	80	P	96	`	112	p		
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q		
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r		
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s		
4	EOT	20	DC4	36	\$	52	4	68	D	84	T	100	d	116	t		
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u		
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v		
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w		
8	BS	24	CAN	40	(56	8	72	H	88	X	104	h	120	x		
9	HT	25	EM	41)	57	9	73	I	89	Y	105	i	121	y		
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z		
11	VT	27	ESC	43	+	59	;	75	K	91	[107	k	123	{		
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124			
13	CR	29	GS	45	-	61	=	77	M	93]	109	m	125	}		
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~		
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL		



Otras codificaciones de caracteres

- Longitud fija
 - 8 bits → 256 caracteres
 - Latin1 (ISO 8859-1), Latin0 (ISO 8859-15), Windows-1252
 - Página de códigos 850
 - 16 bits → 65 536 caracteres
 - UCS-2: 2-byte Universal Character Set (obsoleto)
 - 32 bits → 4 294 967 296 caracteres
 - UCS-4, UTF-32
- Longitud variable
 - UTF-8
 - UTF-16



El estándar Unicode

- Define 149 813 ***caracteres abstractos***
- Cada carácter abstracto se identifica de forma precisa por un entero único: ***punto de código*** (*code point*)
- Cada punto de código se puede codificar a través de distintas **codificaciones**:
 - UTF-8: 1, 2, 3 o 4 *bytes*
 - Compatible con los códigos ASCII de 7 bits
 - UTF-16: 2 o 4 *bytes*
 - UTF-32: 4 *bytes*
 - ...





Problemas con las codificaciones

- Ejemplo 1:
 - Windows con Visual Studio Code utilizando como terminal PowerShell
- Ejemplo 2:
 - Linux con Visual Studio Code usando el terminal del SO





```
File Edit Selection View Go Run Terminal Help bienvenida.cpp - practica1 - Visual Studio Code
bienvenida.cpp x
bienvenida.cpp > ...
1  /*
2  * Curso de Programación 1. Práctica 1
3  * Autores: Miguel Ángel Latre y Javier Martínez
4  * Última revisión: 20 de septiembre de 2019
5  * Resumen: Tarea 2
6  */
7  #include <iostream>
8  using namespace std;
9
10 /*
11 * Programa que escribe en la pantalla los mensajes
12 * "Bienvenidos a la Universidad de Zaragoza" y
13 * "Bienvenidos a Programación 1", cada uno en una línea distinta.
14 */
15 int main() {
16     cout << "Bienvenidos a la Universidad de Zaragoza" << endl;
17     cout << "Bienvenidos a Programación 1" << endl;
18
19     return 0;
20 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1: Code
Bienvenidos a la Universidad de Zaragoza
Bienvenidos a Programaci|n 1
PS E:\Miguel\Documentos\Trabajo\codigo-c++-privado\practica1>
```

master 0 0 Ln 1, Col 1 Spaces: 4 UTF-8 CRLF C++ Win32

Editor de Visual Studio Code: UTF-8

Terminal (PowerShell o Símbolo de sistema de Windows): CP-850

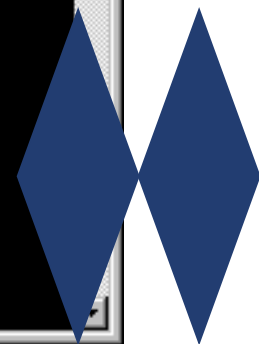




Ejemplo. Carácter ó

```
C:\ /Bienvenida
Bienvenidos a la Universidad
Bienvenidos a Programación 1
Press any key to continue.
```

```
circulo.cc  bienvenida.cc x  circunferencia.cc
1  #include <iostream>
2  using namespace std;
3
4  /*
5   * Pre: ---
6   * Post: Escribe por pantalla Los mensajes
7   *       "Bienvenidos a La Universidad" y
8   *       "Bienvenidos a Programación 1".
9   */
10 int main() {
11     cout << "Bienvenidos a la Universidad" << endl;
12     cout << "Bienvenidos a Programación 1" << endl;
13     return 0;
14 }
```





Carácter «ó»

□ Unicode:

■ «ó»

- **Descripción:** Letra latina O minúscula con acento agudo
- **Punto de código:** U+00F3 (en decimal: 243)
- **Codificación en UTF-8:** *bytes* 195 y 179





Página de códigos 850

128	Ç	129	ü	130	é	131	â	132	ä	133	à	134	å	135	ç
136	ê	137	ë	138	è	139	ï	140	î	141	ì	142	Ä	143	Å
144	É	145	æ	146	Æ	147	ô	148	ö	149	ò	150	û	151	ù
152	ÿ	153	Ö	154	Ü	155	ø	156	£	157	Ø	158	×	159	f
160	á	161	í	162	ó	163	ú	164	ñ	165	Ñ	166	a	167	o
168	¿	169	®	170	¬	171	½	172	¼	173	¡	174	«	175	»
176	☼	177	☼	178	☼	179		180	†	181	Á	182	Â	183	Ã
184	©	185	‡	186	‡	187	‡	188	‡	189	ç	190	¥	191	‡
192	L	193	⊥	194	⊥	195	†	196	—	197	†	198	ã	199	Ä
200	ℒ	201	℞	202	ℒ	203	℞	204	‡	205	=	206	‡	207	α
208	ð	209	Ð	210	Ê	211	Ë	212	È	213	ı	214	Í	215	Î
216	ÿ	217	Ј	218	Г	219	■	220	■	221	ı	222	ì	223	■
224	Ó	225	β	226	Ô	227	Ò	228	õ	229	Õ	230	μ	231	ρ
232	ρ	233	Ú	234	Û	235	Ū	236	ý	237	Ý	238	-	239	ˆ
240		241	±	242	—	243	¾	244	¶	245	§	246	÷	247	ˆ
248	°	249	¨	250	.	251	¹	252	³	253	²	254	■	255	18



Carácter «ó»

```
C:\ /Bienvenida
Bienvenidos a la Universidad
Bienvenidos a Programación 1
Press any key to continue.
```

```
circulo.cc bienvenida.cc x circunferencia.cc
1 #include <iostream>
2 using namespace std;
3
4 /*
5  * Pre: ---
6  * Post: Escribe por pantalla Los mensajes
7  *       "Bienvenidos a La Universidad" y
8  *       "Bienvenidos a Programación 1".
9  */
10 int main() {
11     cout << "Bienvenidos a la Universidad" << endl;
12     cout << "Bienvenidos a Programación 1" << endl;
13     return 0;
14 }
```





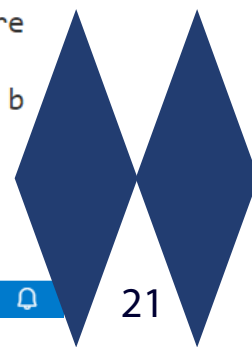
Problemas con las codificaciones

- Ejemplo 1:
 - Windows con Visual Studio Code utilizando PowerShell como terminal
- Ejemplo 2:
 - Linux con Visual Studio Code usando el terminal del SO





```
File Edit Selection View Go Run Terminal Help bienvenida.cpp - practica1 [WSL: Ubuntu] - Visual Studi...
bienvenida.cpp x
bienvenida.cpp > ...
4  * Última revisión: 20 de septiembre de 2019
5  * Resumen: Tarea 2
6  */
7  #include <iostream>
8  using namespace std;
9
10 /*
11  * Programa que escribe en la pantalla los mensajes
12  * "Bienvenidos a la Universidad de Zaragoza" y
13  * "Bienvenidos a Programación 1", cada uno en una línea distinta.
14  */
15 int main() {
16     cout << "Bienvenidos a la Universidad de Zaragoza" << endl;
17     cout << "Bienvenidos a Programación 1" << endl;
18
19     return 0;
20 }
21
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Code
cd "/home/latre/practica1/" && g++ bienvenida.cpp -o bienvenida && "/home/latre/practica1/"bienvenida
latre@Diaño:~/practica1$ cd "/home/latre/practica1/" && g++ bienvenida.cpp -o b
ienvenida && "/home/latre/practica1/"bienvenida
Bienvenidos a la Universidad de Zaragoza
Bienvenidos a Programación 1
latre@Diaño:~/practica1$
```



Más información

- Joel Spolsky, «The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)», *Joel on Software*, 8-10-2013.
 - <https://www.joelonsoftware.com/2003/10/08/the-absolute-minimum-every-software-developer-absolutely-positively-must-know-about-unicode-and-character-sets-no-excuses/>





Operadores asociados

- Los de los tipos enteros
 - Aritméticos: +, -, ...
 - Relación: ==, !=, <, <=, >, >=
- Conversión con enteros pueden ser explícitas:
 - `int('A')` se evalúa como 65
 - `char(66)` se evalúa como 'B'



Propiedades

- Hay secuencias de caracteres con códigos consecutivos crecientes:
 - Mayúsculas del alfabeto inglés: 'A', 'B', 'C', ..., 'X', 'Y' y 'Z'
 - Minúsculas del alfabeto inglés : 'a', 'b', 'c', ..., 'x', 'y' y 'z'
 - Dígitos: '0', '1', '2', '3', '4', '5', '6', '7', '8' y '9'



Expresiones con caracteres

- `char c = 'E';`
- 1. `c == 'A'`
- 2. `c != 'e'`
- 3. `c >= 'A'`
- 4. `c <= 'Z'`
- 5. `c >= 'A' && c <= 'Z'`
- 6. `c >= 'a'`
- 7. `c <= 'z'`
- 8. `c >= 'a' && c <= 'z'`
- 9. `char(c + 1)`
- 10. `char(c + 32)`
- 11. `char(c - 'A' + 'a')`

65	A	97	a
66	B	98	b
67	C	99	c
68	D	100	d
69	E	101	e
70	F	102	f
71	G	103	g
72	H	104	h
73	I	105	i
74	J	106	j
75	K	107	k
76	L	108	l
77	M	109	m
78	N	110	n
79	O	111	o
80	P	112	p
81	Q	113	q
82	R	114	r
83	S	115	s
84	T	116	t
85	U	117	u
86	V	118	v
87	W	119	w
88	X	120	x
89	Y	121	y
90	Z	122	z

Biblioteca estándar <ctype>

- Character handling functions. This header declares a set of functions to classify and transform individual characters.
 - `isalnum`: Check if character is alphanumeric
 - `isalpha`: Check if character is alphabetic
 - `islower`: Check if character is lowercase letter
 - `isupper`: Check if character is uppercase letter
 - `isdigit`: Check if character is decimal digit
 - `isspace`: Check if character is a white-space
 - `tolower`: Convert uppercase letter to lowercase
 - `toupper`: Convert lowercase letter to uppercase



Índice

- Caracteres
- Cadenas de caracteres



Cadenas de caracteres

- Secuencias de 0, 1 o más caracteres
- Representación literal entre comillas
 - ""
 - "A"
 - "Programación 1"



Cadenas de caracteres

- Posibilidades de representación
 - Vector de datos de tipo **char** finalizadas con el carácter ' $\backslash 0$ '
 - Conocidas como *null-terminated strings* o *C strings*.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
U	n	i	z	a	r	NUL	?	?	?	?	?	?	?	?	?	?	?	?	?



Cadenas de caracteres

- Posibilidades de representación
 - Vector de datos de tipo **char** finalizadas con el carácter '`\0`'
 - Conocidas como *null-terminated strings* o *C strings*.
 - No trabajaremos con ellas en este curso
 - Clase predefinida **string**



Clase string

- Dominio de valores
 - Secuencias de 0, 1 o más caracteres
 - Longitud máxima de la secuencia limitada por la memoria disponible
- Representación externa
 - Secuencia de caracteres entre comillas
 - ""
 - "A"
 - "Programación 1"



Clase `string`

- Representación interna
 - Objetos
 - En último término, un vector de datos de tipo `char`.
- Operaciones
 - Las definidas en el módulo predefinido `<string>` para la clase **`string`**:
 - <http://www.cplusplus.com/reference/string/string/>



Clase string

Operaciones

Operación	Operador o método
Asignación	=
Longitud de la cadena	length()
Acceso a caracteres	[] at()
Comparación	compare() == != < <= > >=
Concatenación	+ +=
Extracción de teclado	>>
Inserción en pantalla	<<

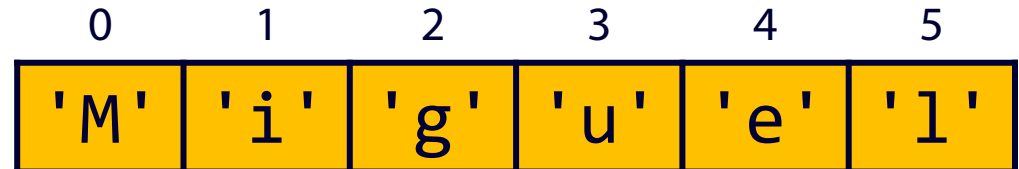


Clase string

Ejemplo 1: declaración, asignación y escritura en pantalla

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

int main() {
    string nombre;
    nombre = "Miguel";
    cout << nombre << endl;
    return 0;
}
```





Clase string

Ejemplo 2: lectura de teclado y acceso a un carácter

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

int main() {
    cout << "Escribe tu nombre de pila: ";
    string nombre;
    cin >> nombre;

    cout << "Hola, " << nombre << endl;
    cout << "Tu nombre empieza por la letra "
         << nombre[0] << "." << endl;
    return 0;
}
```



Clase string

Ejemplo 3: acceso a un carácter con el método «at()»

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

int main() {
    cout << "Escribe tu nombre de pila: ";
    string nombre;
    cin >> nombre;

    cout << "Hola, " << nombre << endl;
    cout << "Tu nombre empieza por la letra "
         << nombre.at(0) << "." << endl;
    return 0;
}
```



Clase string

Ejemplo 4: «length()» y operadores de comparación

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

int main() {
    cout << "Escribe tu nombre de pila: ";
    string nombre;
    cin >> nombre;
    cout << "Tu nombre tiene " << nombre.length() << " letras." << endl;
    if (nombre == "Miguel") {
        cout << "Te llamas como yo." << endl;
    } else if (nombre < "Miguel") {
        cout << "Tu nombre va alfabéticamente antes que el mío." << endl;
    } else {
        cout << "Tu nombre va alfabéticamente después del mío." << endl;
    }
    return 0;
}
```



Clase string

Ejemplo 5: concatenación

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

int main() {
    string nombre, apellido;
    cout << "Escribe tu nombre de pila: ";
    cin >> nombre;
    cout << "Escribe tu primer apellido: ";
    cin >> apellido;

    string nombreCompleto = nombre + " " + apellido;

    cout << "Tu nombre completo es " << nombreCompleto << endl;
    return 0;
}
```

Vectores de cadenas de caracteres

```
/*  
 * Pre: 1 ≤ mes ≤ 12  
 * Post: Escribe en la pantalla el nombre (en  
 * mayúsculas) del mes correspondiente al valor del  
 * parámetro «mes».  
 */  
void escribirNombreMes(const unsigned mes) {  
    const string NOMBRES_MES[NUM_MESES] = { "ENERO",  
        "FEBRERO", "MARZO", "ABRIL", "MAYO", "JUNIO",  
        "JULIO", "AGOSTO", "SEPTIEMBRE", "OCTUBRE",  
        "NOVIEMBRE", "DICIEMBRE" };  
    cout << NOMBRES_MES[mes - 1];  
}
```



¿Cómo se puede estudiar este tema?

- Repasando estas transparencias
- Trabajando con el código de estas transparencias
 - <https://github.com/prog1-eina/tema-10-cadenas>
- Leyendo material adicional publicado en Moodle
- Trabajando con los problemas del tema