Programación 1

Tema 10

Representación de cadenas de caracteres

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Índice

- Caracteres
- Cadenas de caracteres
- Problemas

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

Wikimedia Commons contributors, 'File:Brother typewriter by awdean1.jpg', *Wikimedia Commons, the free media repository,* 2016, <https://commons.wikimedia.org/w/index.php?title=File:Brother_typewriter_by_awdean1.jpg&oldid=219334406>

Caracteres

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
Unicode

- Estándar de codificación de caracteres
- Dominio de valores:
  - Alfabeto latino: A b c d e f g h i j k l m n o p q r s t u v w x y z
  - Alfabeto griego: α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ω
  - Alfabeto cirílico: б в г ж з и й к л м н п с т у ф х ц ч шщ ъ ы ь
  - Alfabetos centro-europeos: Á â ă ä Ĺ ć ç č é ę ë ě í î ď đ ń ň ó ô ŕ ř
  - Alfabeto árabe: چ ژ گ ک ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د س ص ع ل م و ى
  - Alfabeto hebreo: א ב ג ד ה ו ז ח ט י כ ך ל מ ם נ ן ס ע פ ף צ ץ ק ר ש ת
  - Alfabetos asiáticos: 中文萬國碼際字出典フリ百科事典ィキペデア
  - Símbolos: £ ₧ € № ¼ ½ ¾ ⅐ ← ↑ ↗ ⇨ ∀ ∂ ∃ ∄ ⌘ ⌦
  - Emoji: ☺ 😉 😇 😛 😂 😱 💀 💩 👽 😸 🏃 👋 👶 👧 🏄 🎄 🍐 👠 🚂 🦵 🎊

Caracteres en C++

- Dos tipos
  - **`char`**
    - 1 *byte* (8 bits)
  - **`wchar_t`**
    - 2 *bytes* (16 bits) en GNU GCC

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
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

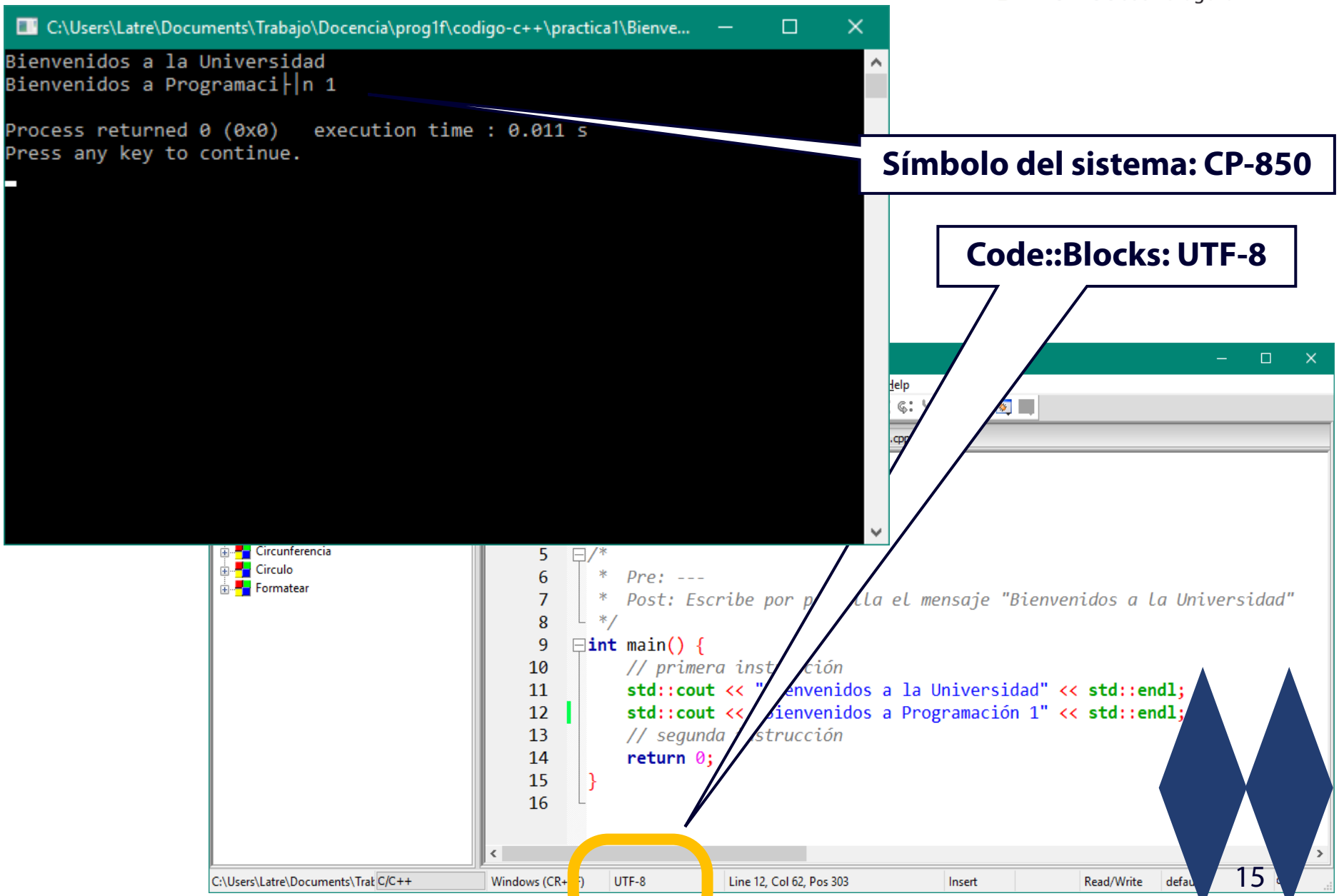
- 8 bits → 256 caracteres
  - Latin1 (ISO 8859-1), Latin0 (ISO 8859-15), Windows-1252
  - Página de códigos 850
- 16 bits → 65 536 caracteres
  - UCS-2 (2-byte Universal Character Set)
- Longitud variable → ~137 000 caracteres definidos por Unicode
  - UTF-8
  - UTF-16

Universal Character Set (UCS)

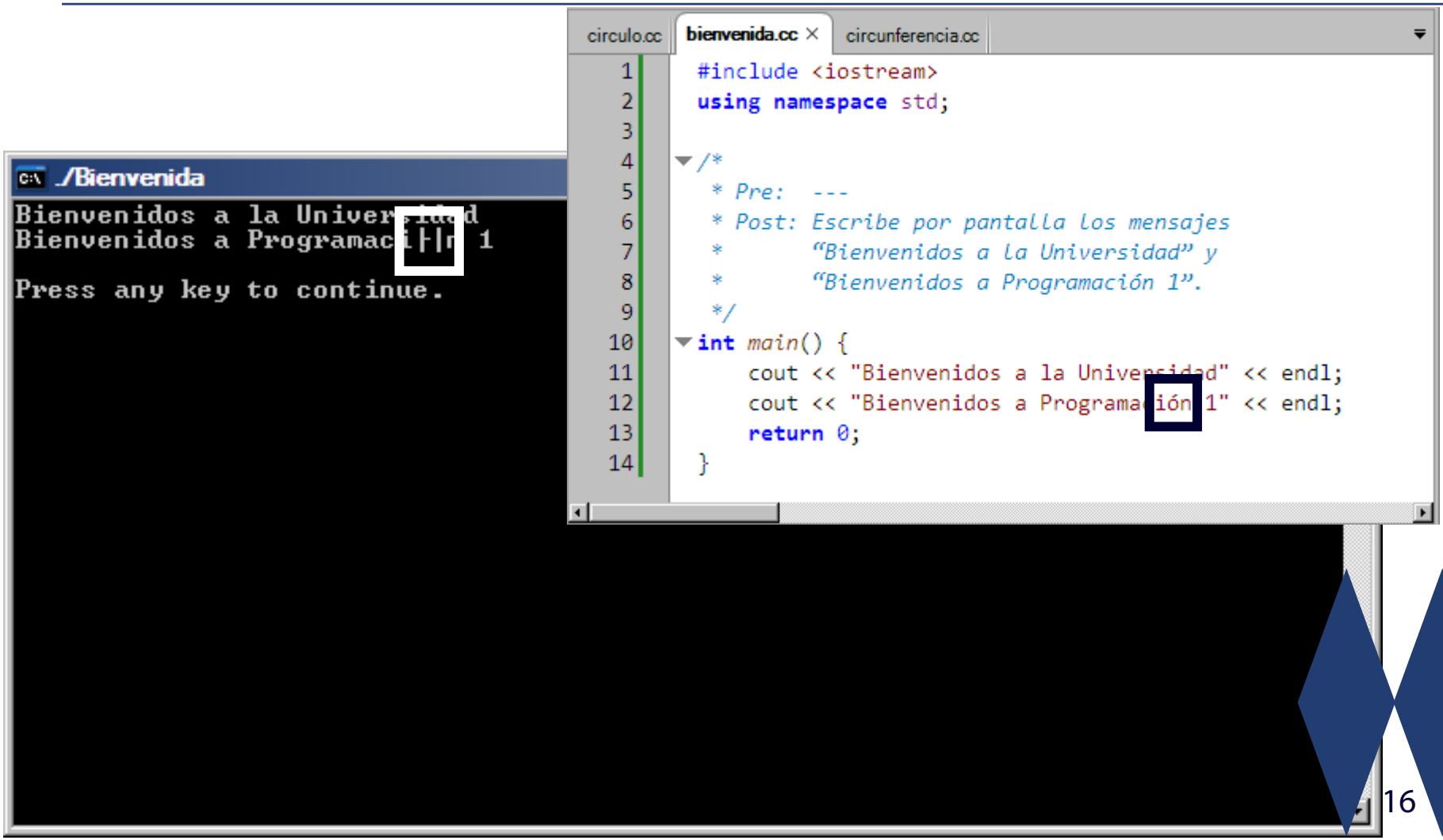
- Estándar internacional ISO/IEC 10646 (~Unicode)
  - Define 110 000 ***caracteres abstractos***
  - Cada carácter abstracto se identifica de forma precisa por un entero único: ***punto de código*** (*code point*)
  - Cada punto de código se puede codificar de acuerdo con distintas **codificaciones**:
    - UTF-8
      - 1, 2, 3 o 4 bytes
      - Compatible con los códigos ASCII de 7 bits
    - UTF-16
      - 2 o 4 bytes

Problemas con las codificaciones

- Ejemplo 1:
  Windows con Code::Blocks usando UTF-8
- Ejemplo 2:
  Windows con Code::Blocks usando la codificación predeterminada de Windows (CP-1252)


C:\Users\Latre\Documents\Trabajo\Docencia\prog1f\codigo-c++\practica1\Bienve...
Bienvenidos a la Universidad
Bienvenidos a Programaci├│n 1
Process returned 0 (0x0) execution time : 0.011 s
Press any key to continue.
Símbolo del sistema: CP-850
Code::Blocks: UTF-8
Circunferencia
Circulo
Formatear
5 /*
6 * Pre: ---
7 * Post: Escribe por p lla el mensaje "Bienvenidos a la Universidad"
8 */
9 int main() {
10 // primera inst ción
11 std::cout << " envenidos a la Universidad" << std::endl;
12 std::cout << ienvenidos a Programación 1" << std::endl;
13 // segunda strucción
14 return 0;
15 }
16
C:\Users\Latre\Documents\Trak C/C++
Windows (CR+
UTF-8
Line 12, Col 62, Pos 303
Insert
Read/Write

Carácter «ó»


circulo.cc
bienvenida.cc
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 }
./Bienvenida
Bienvenidos a la Universidad
Bienvenidos a Programación 1
Press any key to continue.

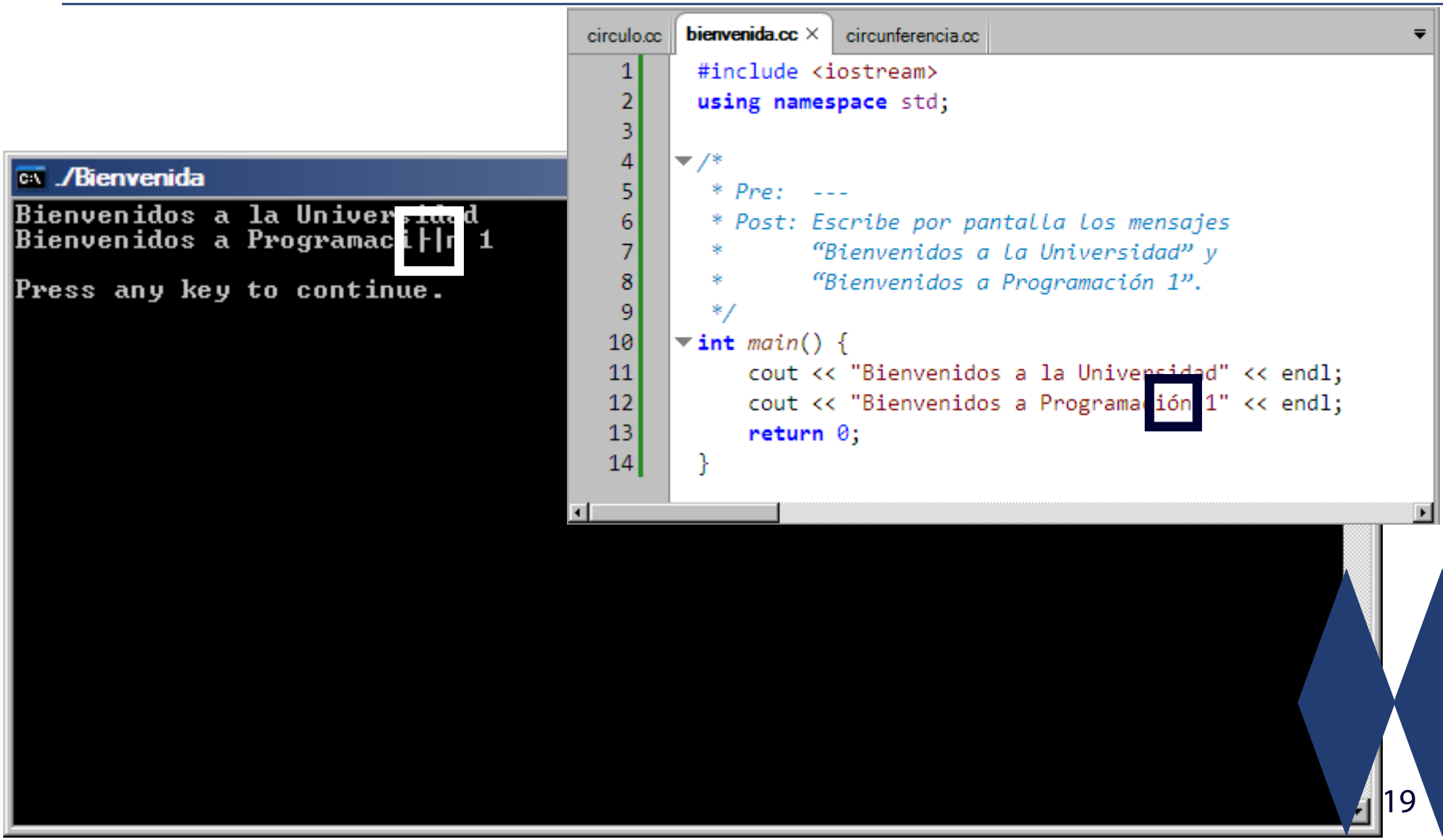
Carácter «ó»

- 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

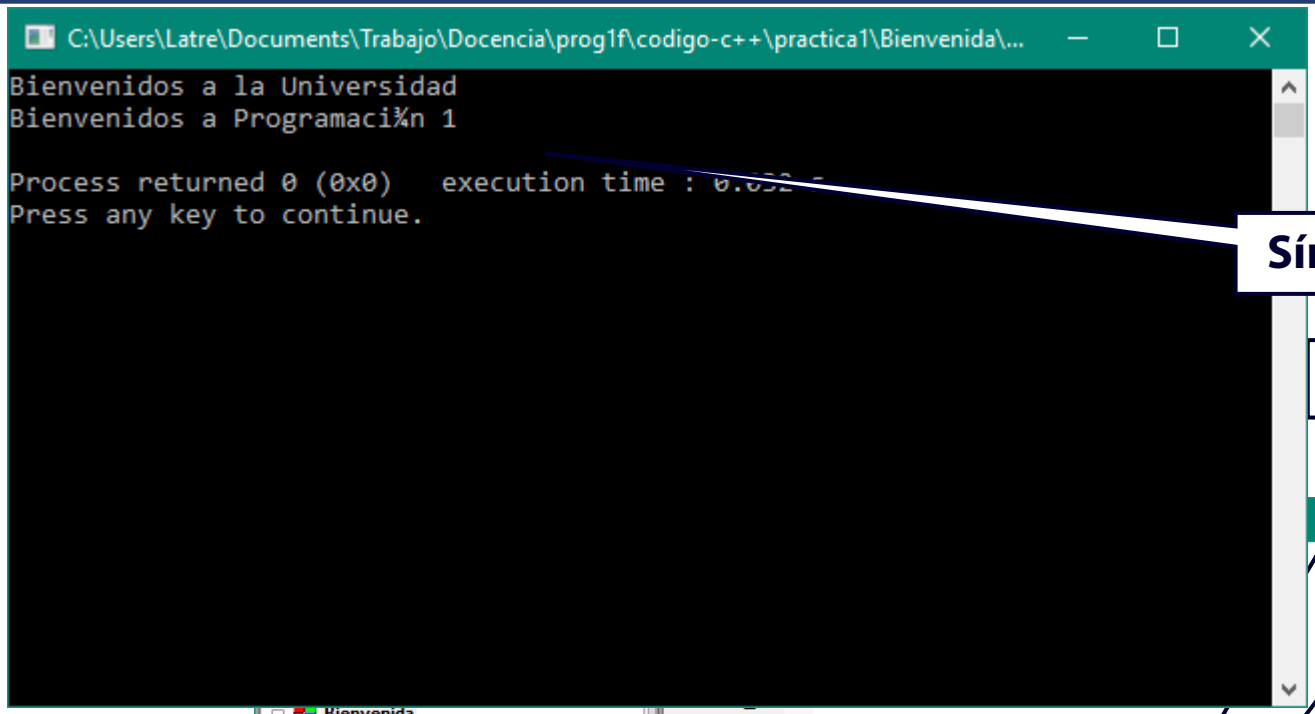
128	Ç	129	ü	130	é	131	â	132	ä	133	à	134	å	135	ç
136	ê	137	ë	138	è	139	ï	140	î	141	ì	142	Ä	143	Å
144	É	145	æ	146	Æ	147	ô	148	ö	149	ò	150	û	151	ù
152	ÿ	153	Ö	154	Ü	155	ø	156	£	157	Ø	158	×	159	ƒ
160	á	161	í	162	ó	163	ú	164	ñ	165	Ñ	166	ª	167	º
168	¿	169	®	170	¬	171	½	172	¼	173	¡	174	«	175	»
176	▒	177	▒	178	▓	179	│	180	┤	181	Á	182	Â	183	À
184	©	185	╣	186	║	187	╗	188	╝	189	¢	190	¥	191	┐
192	└	193	┴	194	┬	195	├	196	─	197	┼	198	ã	199	Ã
200	╚	201	╔	202	╩	203	╦	204	╠	205	═	206	╬	207	¤
208	ð	209	Ð	210	Ê	211	Ë	212	È	213	ı	214	Í	215	Î
216	Ï	217	┘	218	┌	219	█	220	▄	221	¦	222	Ì	223	▀
224	Ó	225	ß	226	Ô	227	Ò	228	õ	229	Õ	230	µ	23	Þ
232	þ	233	Ú	234	Û	235	Ú	236	ý	237	Ý	238	¯	2	´
240		241	±	242	_	243	¾	244	¶	245	§	246	÷	24	¸
248	°	249	¨	250	·	251	¹	252	³	253	²	254	■	255	18

Carácter «ó»


circulo.cc
bienvenida.cc
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 }
./Bienvenida
Bienvenidos a la Universidad
Bienvenidos a Programación 1
Press any key to continue.

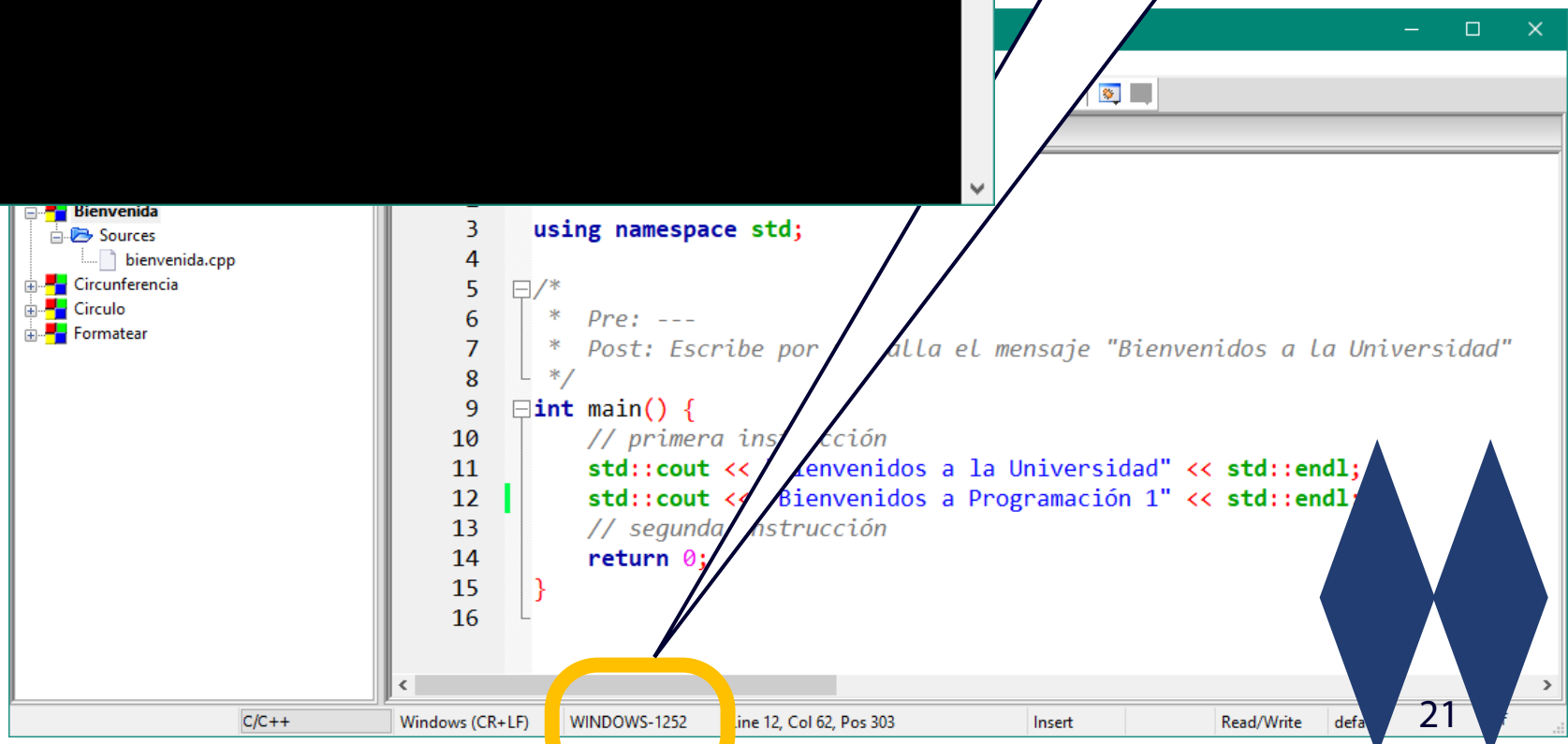
Problemas con las codificaciones

- Ejemplo 1:
  Windows con Code::Blocks usando UTF-8
- Ejemplo 2:
  Windows con Code::Blocks usando la codificación predeterminada de Windows (CP-1252)



Símbolo del sistema: CP-850

Code::Blocks: CP-1252



Carácter «ó»

circulo.cc | bienvenida.cc | 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 }
```

```
C:\Users\Latre\Documents\Trabajo\Docencia\pr
Bienvenidos a la Universidad
Bienvenidos a Programaci¾n 1

Process returned 0 (0x0)   execution
Press any key to continue.
```

Carácter «ó»

- Windows-1252 o CP-1252:
  - «ó»
    - Código entero 243 (00F3 en hexadecimal)

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	ƒ
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	À
184	©	185	╣	186	║	187	╗	188	╝	189	¢	190	¥	191	┐
192	└	193	┴	194	┬	195	├	196	─	197	┼	198	ã	199	Ã
200	╚	201	╔	202	╩	203	╦	204	╠	205	═	206	╬	207	¤
208	ð	209	Ð	210	Ê	211	Ë	212	È	213	ı	214	Í	215	Î
216	Ï	217	┘	218	┌	219	▉	220	▄	221	¦	222	Ì	223	▀
224	Ó	225	ß	226	Ô	227	Ò	228	õ	229	Õ	230	µ	23	Þ
232	þ	233	Ú	234	Û	235	Ú	236	ý	237	Ý	238	¯	2	´
240		241	±	242	_	243	¾	244	¶	245	§	246	÷	24	¸
248	°	249	¨	250	·	251	¹	252	³	253	²	254	■	255	24

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')` devuelve 65
  - `char(66)` devuelve 'B'
- 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';`
- `c == 'A'`
- `c != 'e'`
- `c >= 'A'`
- `c <= 'Z'`
- `c >= 'A' && c <= 'Z'`
- `c >= 'a'`
- `c <= 'z'`
- `c >= 'a' && c <= 'z'`
- `char(c + 1)`
- `char(c + 32)`
- `char(c - 'A' + 'a')`

Ejemplos

```
/*
 * Pre:  ---
 * Post: Si «c» es un carácter que representa
 *       una letra en mayúscula entonces
 *       devuelve true; en otro caso devuelve
 *       false.
 */
bool esMayuscula(const char c) {
    return c >= 'A' && c <= 'Z';
}
```

Ejemplos

```
/*
 * Pre:  ---
 * Post: Si «c» es un carácter que
 *       representa una letra minúscula
 *       entonces devuelve true; en otro
 *       caso devuelve false.
 */
bool esMinuscula(const char c) {
    return c >= 'a' && c <= 'z';
}
```

Ejemplos

```
/*
 * Pre:  ---
 * Post: Si «c» es un carácter que
 *       representa un dígito entonces
 *       devuelve true; en otro caso
 *       devuelve false.
 */
bool esDigito(const char c) {
    return c >= '0' && c <= '9';
}
```

Ejemplos

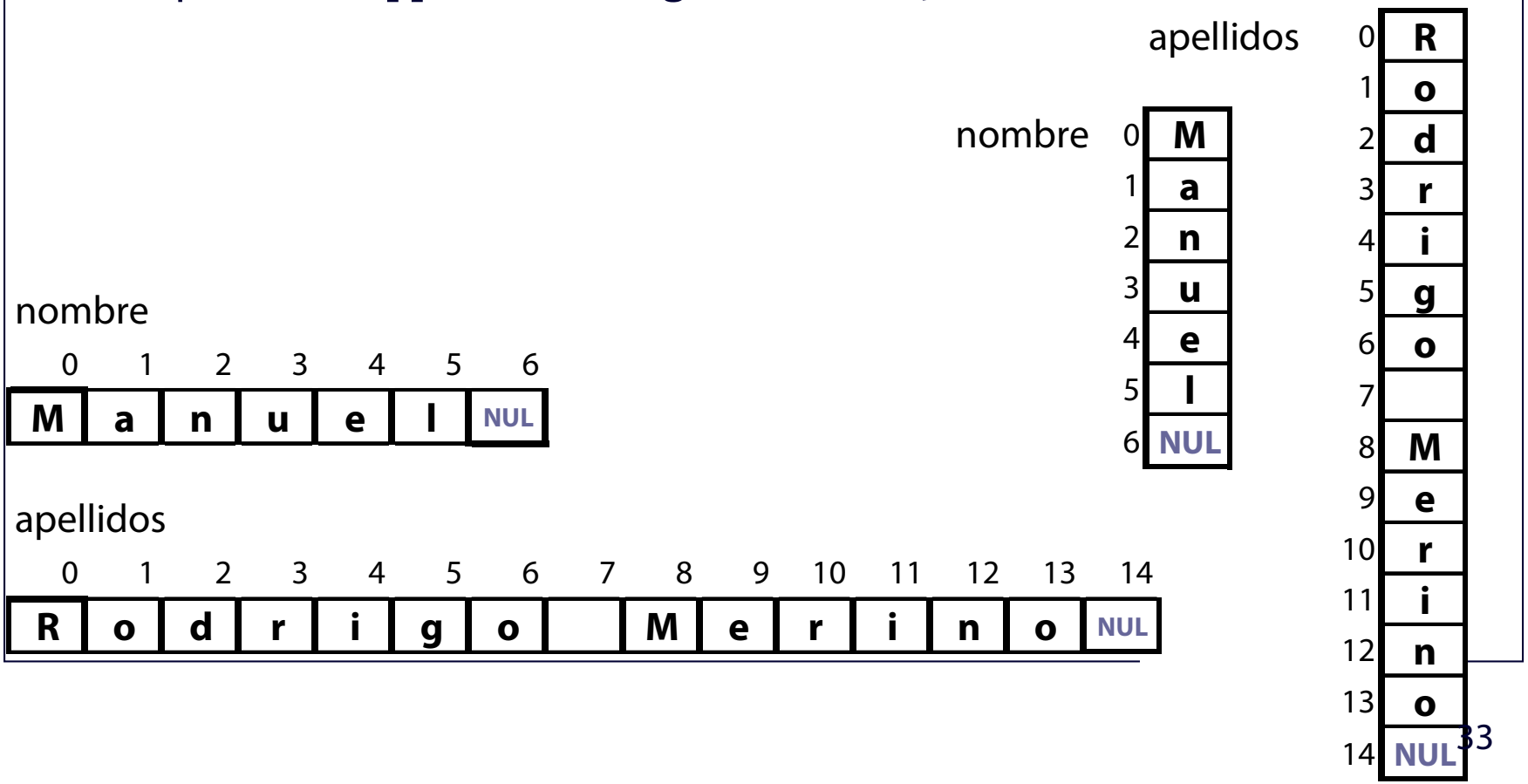
```
/*
 * Pre:  ---
 * Post: Si «c» es un carácter que representa un dígito entonces
 *       devuelve el valor numérico comprendido entre 0 y 9
 *       representado por c;
 *       en otro caso devuelve un valor negativo.
 */
int valorDigito(const char c) {
   if (esDigito(c)) {
      return c - '0';
   }
   else {
      return -1;
   }
}
```

Cadenas de caracteres

- Secuencia de 0, 1 o más caracteres
- Representación literal entre comillas
  - `""`
  - `"A"`
  - `"Programación 1"`
- Tipos de datos para su representación
  - Tabla de datos de tipo **`char`**
  - Clase predefinida `string`
    - No la usaremos en este curso

Cadenas de caracteres

```
char nombre[] = { 'M', 'a', 'n', 'u', 'e', 'l', '\0' };
char apellidos[] = "Rodrigo Merino";
```

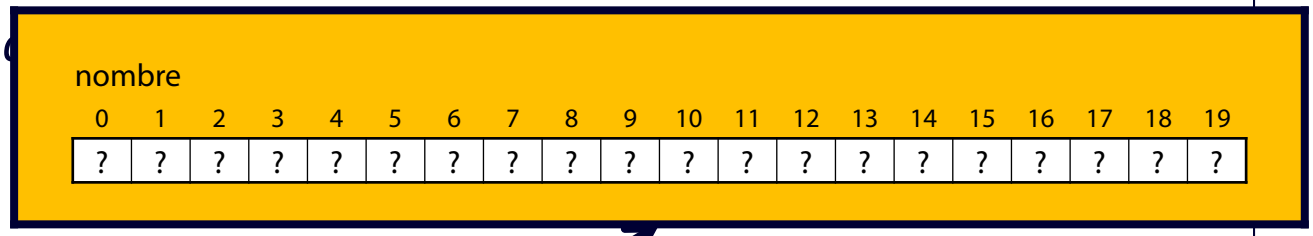


Cadenas de caracteres

```
#include <iostream>
using namespace std;
const int MAX_LONG_NOMBRE = 20;

/*
 * Pre:  ---
 * Post: He pedido el nombre del usuario, lo ha leído del
 *       teclado y lo ha vuelto a escribir en la pantalla.
 */
int main() {
    char nombre[MAX_LONG_NOMBRE];
    cout << "Escriba su nombre: " << flush;
    cin >> nombre;
    cout << "Su nombre es: " << nombre << endl;
    return 0;
}
```

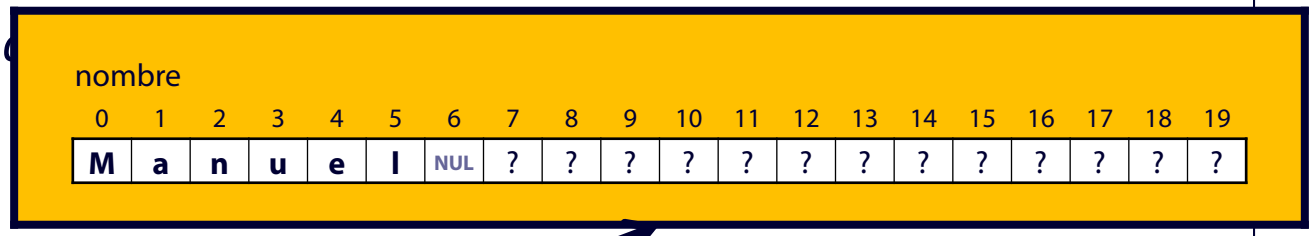
Cadenas de caracteres

```
#include <iostream>
using namespace std;
const int MAX_LONG_NOMBRE = 20;

/*
 * Pre:  ---
 * Post: He pedi
 *       teclado
 */
int main() {
    char nombre[MAX_LONG_NOMBRE];
    cout << "Escriba su nombre: " << flush;
    cin >> nombre;
    cout << "Su nombre es: " << nombre << endl;
    return 0;
}
```



Cadenas de caracteres

```
#include <iostream>
using namespace std;
const int MAX_LONG_NOMBRE = 20;

/*
 * Pre:  ---
 * Post: He pedi
 *       teclado
 */
int main() {
    char nombre[MAX_LONG_NOMBRE];
    cout << "Escriba su nombre: " << flush;
    cin >> nombre;
    cout << "Su nombre es: " << nombre << endl;
    return 0;
}
```



Cadenas de caracteres

- En la biblioteca predefinida `<cstring>`
  - `strcpy(cad1, cad2)`
    - copia la cadena `cad2` en `cad1`
  - `strcat(cad1, cad2)`
    - modifica `cad1` concatenándole la cadena `cad2`
  - `strlen(cad)`
    - devuelve la longitud de la cadena `cad`
  - `strcmp(cad1, cad2)`
    - compara las cadena `cad1` y `cad2`
    - devuelve 0 si el contenido de ambas es idéntico
    - devuelve un valor positivo si `cad1` es alfabéticamente posterior a `cad2`
    - devuelve un valor negativo si `cad1` es alfabéticamente anterior a `cad2`

Ejemplo

```
char letras[] = "ABCD";
char alfabeto[30] = "";
cout << "[" << alfabeto << "] tiene " << strlen(alfabeto)
     << " caracteres" << endl;

strcpy(alfabeto, letras);
cout << "[" << alfabeto << "] tiene " << strlen( alfabeto )
     << " caracteres" << endl;

strcat(alfabeto, "EFGHIJ");
cout << "[" << alfabeto << "] tiene " << strlen( alfabeto )
     << " caracteres" << endl;

strcat(alfabeto, "KLMNOPQRSTUVWXYZ");
cout << "[" << alfabeto << "] tiene " << strlen( alfabeto )
     << " caracteres" << endl;
```

Ejemplo

```
[] tiene 0 caracteres
[ABCD] tiene 4 caracteres
[ABCDEFGHIJ] tiene 10 caracteres
[ABCDEFGHIJKLMNOPQRSTUVWXYZ] tiene 26 caracteres
```

Conversión de cadena a entero

```
/*
 * Pre:  «cadena» almacena una secuencia de caracteres
 *       que representan literalmente un entero con la
 *       siguiente sintaxis:
 *          <literal_entero> ::= [ <signo> ]
 *                               <digito> { <digito> }
 *          <signo> ::= "+"  | "-"
 *          <digito> ::= "0" | "1" | "2" | "3" | "4"
 *                     | "5" | "6" | "7" | "8" | "9"
 * Post: Devuelve el valor entero representado en
 *       «cadena».
 */
int valorEntero(const char cadena[]);
```

Conversión de cadena a entero

```
int valorEntero(const char cadena[]) {
   int i = 0;
   bool negativo = false;

   // cadena[0] es un dígito,
   // un signo más o un signo menos
   if (cadena[0] == '+') {
      i++;
   }
   else if (cadena[0] == '-') {
      i++;
      negativo = true;
   }

   ...
}
```

Conversión de cadena a entero

```
int valorEntero(const char cadena[]) {
  ...
  // cadena[i] es un dígito
  int valor = 0;
  while (cadena[i] != '\0') {
    int valorDigito = cadena[i] - '0';
    valor = 10 * valor + valorDigito;
    i++;
  }
  if (negativo) {
    valor = -valor;
  }
  return valor;
}
```

Conversión de entero a cadena

```
/*
 * Pre:  ---
 * Post: Asigna a «literal» una secuencia de caracteres
 *       que representa literalmente el entero «valor»
 *       con la siguiente sintaxis:
 *          <literal_entero> ::= [ <signo> ]
 *                               <digito> { <digito> }
 *          <signo> ::= "+"  | "-"
 *          <digito> ::= "0" | "1" | "2" | "3" | "4"
 *                     | "5" | "6" | "7" | "8" | "9"
 */
void literalEntero(const int valor, char literal[]);
```

Conversión de entero a cadena (versión apuntes)

```
void literalEntero(const int valor, char literal[]) {
   // Determinación de si valor es negativo o no

   // Cálculo de las unidades. Fuera del bucle para
   // que la representación de 0 sea "0".

   // Cálculo del resto de las cifras de menos
   // a más significativas

   // Se pone el signo si es preciso

   // Inversión de los dígitos almacenados en
   // «literal»

   // Inserta el carácter de final de la cadena
}
```

Conversión de entero a cadena (versión apuntes)

```
void literalEntero(const int valor, char literal[]) {
   // Determinación de si valor es negativo o no
   bool negativo = false;
   if (valor < 0) {
      negativo = true;
      valor = -valor;
   }

   // Cálculo de las unidades. Fuera del bucle
   // para que la representación de 0 sea "0".
   literal[0] = '0' + valor % 10;
   valor = valor / 10;

   ...
}
```

Conversión de entero a cadena (versión apuntes)

```
void literalEntero(const int valor, char literal[]) {
   ...
   // Cálculo del resto de las cifras de menos a más
   // significativas. El cursor «i» indica la siguiente
   // componente de «literal» a almacenar.
   int i = 1;
   while (valor != 0) {
      literal[i] = '0' + valor % 10;
      valor = valor / 10;
      i++;
   }

   // Se pone el signo si es preciso
   if (negativo) {
      literal[i] = '-';
      i++;
   }
   ...
}
```

Conversión de entero a cadena (versión apuntes)

```
void literalEntero(const int valor, char literal[])
{
   ...

   // Inversión de los dígitos almacenados
   // en «literal»
   for (int j = 0; j < i / 2; j++) {
      char aux = literal[j];
      literal[j] = literal[i - j - 1];
      literal[i - j - 1] = aux;
   }

   // Inserta el carácter de final de la cadena
   literal[i] = '\0';
}
```

Conversión de entero a cadena (otra versión)

```
void literalEntero(const int valor,
                   char literal[]) {
  // Cálculo del número de cifras.
  // Determinación de si valor es negativo o no.
  // Inserción del carácter de final de la
  //     cadena.
  // Cálculo de las cifras de menos a más
  //     significativas, almacenándose en las
  //     componentes finales.
}
```

Conversión de entero a cadena (otra versión)

```
void literalEntero(const int valor, char literal[]) {
   // Cálculo del número de cifras
   int i = numCifras(valor);
   int n = valor;

   // Determinación de si valor es negativo o no
   if (valor < 0) {
      literal[0] = '-';
      n = -valor;
      i++;
   }
   // Inserta el carácter de final de la cadena
   literal[i] = '\0';
   ...
}
```

Conversión de entero a cadena (otra versión)

```
void literalEntero(const int valor, char literal[]) {
   ...

   if (n == 0) {
      // Tratamiento específico del caso del 0;
      literal[0] = '0';
   }
   else {
      // Cálculo de las cifras de «n» de menos a más
      // significativas
      while (n > 0) {
         i--;
         literal[i] = '0' + n % 10;
         n = n / 10;
      }
   }
}
```