TEMA 5

TRATAMIENTO DE CADENAS

TRATAMIENTO DE CADENAS

- ÍNDICE
  - JUEGO DE CARACTERES
  - CADENA DE CARACTERES
  - DATOS DE TIPO CARÁCTER / CADENA
  - OPERACIONES CON CADENAS

JUEGO DE CARACTERES

- Las computadoras trabajan con **distintos juegos de caracteres**, entre los que destacan:
  - **Código ASCII** (American Standard Code for Information Interchange). Muy popular.
    - ASCII básico. 128 caracteres (7 bits).
    - ASCII extendido. 256 caracteres (8 bits).
  - **Código EBCDIC** (Extended Binary Coded Decimal Interchange Code) 256 caracteres (8 bits). Empleado por IBM.
  - **UNICODE** (Universal Character Encoding). 65.536 caracteres (16 bits).
- Un **juego de caracteres** es una tabla en la que cada símbolo tiene asociado un número.

JUEGO DE CARACTERES

The ASCII code

American Standard Code for Information Interchange

ASCII control characters

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

ASCII printable characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
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	_			

Extended ASCII characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	å	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	ë	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	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	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°
153	99h	Ö	185	B9h	╣	217	D9h	┘	249	F9h	¨
154	9Ah	Ü	186	BAh	║	218	DAh	┌	250	FAh	·
155	9Bh	ø	187	BBh	╗	219	DBh	█	251	FBh	¹
156	9Ch	£	188	BCh	╝	220	DCh	▄	252	FCh	³
157	9Dh	Ø	189	BDh	¢	221	DDh	¦	253	FDh	²
158	9Eh	×	190	BEh	¥	222	DEh	Ì	254	FEh	■
159	9Fh	ƒ	191	BFh	┐	223	DFh	▀	255	FFh	

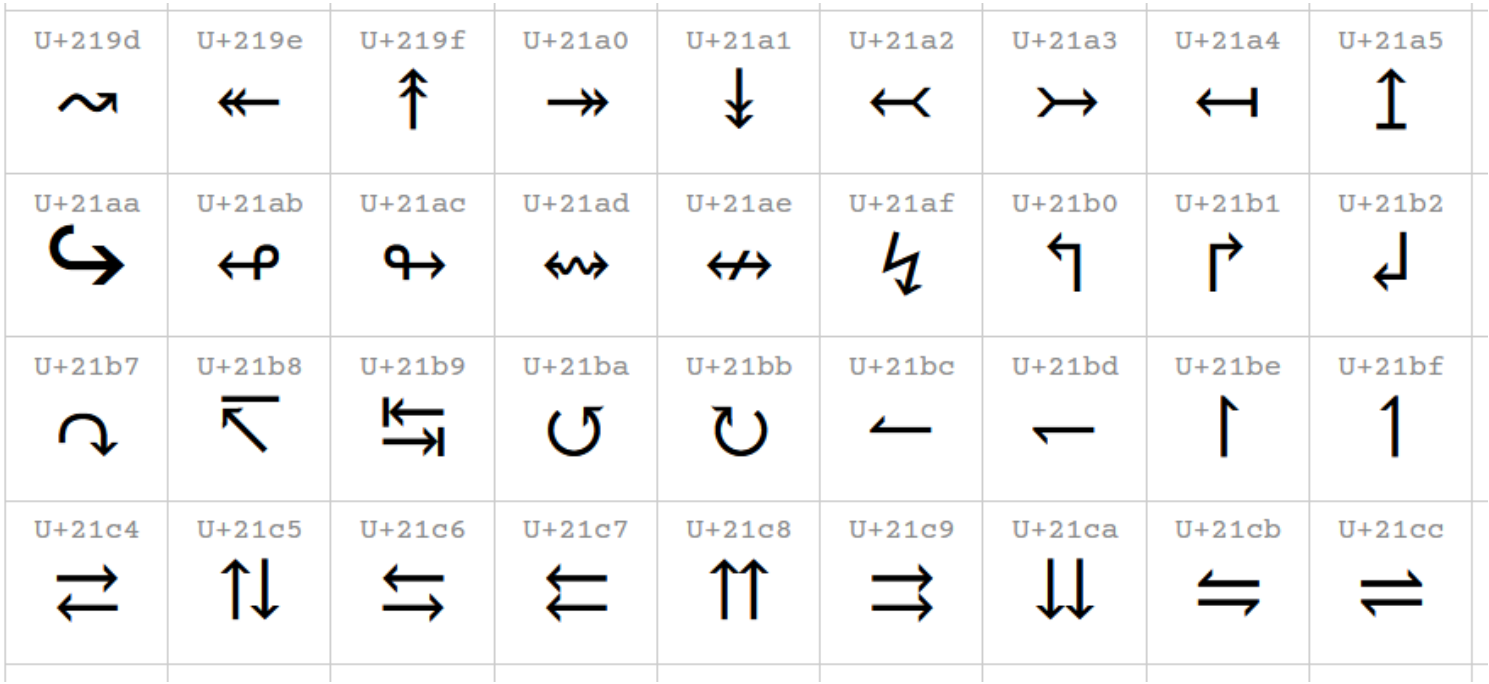
JUEGO DE CARACTERES

- UNICODE:
  - Es un estándar mantenido por el *Unicode Technical Committee* (UTC): https://unicode.org/consortium/utc.html
  - Facilita el tratamiento informático, transmisión y visualización de **textos de múltiples lenguajes y disciplinas técnicas,** además de textos clásicos de **lenguas muertas**.
  - Unicode especifica un nombre e identificador numérico único para cada carácter o símbolo, el ***code point* o *punto de código***, además de otras informaciones necesarias para su uso correcto: direccionalidad, capitalización y otros atributos.
  - Punto de código: **U+nnnn**, donde nnnn = número hexadecimal.
  - Ejemplo: letra “a” es U+0061 o en texto: “LATIN SMALL LETTER A”

JUEGO DE CARACTERES

- UNICODE

  Ejemplos de símbolos:



https://codepoints.net/

JUEGO DE CARACTERES

- Ambos juegos (ASCII – UNICODE) poseen 4 tipos de caracteres:
  - **Alfabéticos:** Letras mayúsculas y minúsculas.
  - **Numéricos:** Números.
  - **Especiales:** Todos los que no son letras y números, que vienen en el teclado (/, %, @, !, …)
  - **De control:** No se pueden imprimir y tienen asignados caracteres especiales. Sirven para órdenes: salto de línea, fin de texto, etc.

Generalmente nos referiremos a estas clases como:

- **Caracteres alfanuméricos:** que abarcan las dos primeras.
- **Caracteres de texto:** que abarcan las tres primeras categorías.

CADENA DE CARACTERES o STRING

- Es un conjunto de **cero ó más caracteres**. Entre estos caracteres puede estar incluido el blanco.
  - Las cadenas de caracteres se delimitan con dobles comillas “ “ (C, C++, Java), pero en algunos lenguajes van con ‘ ‘ (Pascal, Fortran).
    - Ej: ‘esto es una cadena de caracteres \n’
  - Las cadenas de caracteres se almacenan en posiciones contiguas de memoria.
  - Una **subcadena** es una cadena extraída de otra cadena.
  - Las que son **mutables** se pueden modificar, las **inmutables** no se pueden modificar (en Java son inmutables).

CADENA DE CARACTERES

– La **longitud de una cadena** es el número de caracteres de la misma. Si hubiese algún carácter de control, por ejemplo, para señalar el fin de cadena, no se consideraría en la longitud.

– Si una **cadena** tiene **longitud cero**, la llamamos **cadena nula o vacía,** por lo que no tiene ningún carácter de texto. Esto no quiere decir que no tenga ningún carácter válido, porque puede haber algún carácter de control que forme parte de la cadena.

DATOS DE TIPO CARÁCTER / CADENA

- **Secuencia de escape.** Permiten representar caracteres (de control) que no se pueden escribir directamente desde el teclado. **Ej: \t : Tabulación, \b: retroceso.**
- **Constantes**:
  - Una constante **de tipo cadena** es un conjunto de cero o más caracteres encerrados entre ‘ ’.
  - Una constante **de tipo carácter** es un solo carácter encerrado entre comillas.
  - Si dentro de la cadena quiero poner como parte de la cadena las ‘, las pongo 2 veces. Esto depende del lenguaje.
    - Ejemplo ‘Hola ‘’Adios’, veriamos: Hola’Adios

DATOS DE TIPO CARÁCTER / CADENA

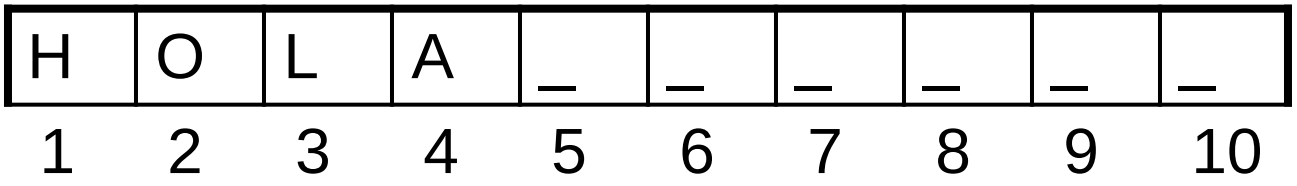
- **Variables**:
  - Hay que distinguir entre una variable de **tipo carácter** y una variable de **tipo cadena**: El contenido de una variable de tipo cadena es un conjunto de cero ó más caracteres encerrados entre ‘ ‘, mientras que una variable de tipo carácter es un solo carácter encerrado entre ‘ ‘.

```
Definir letra Como Caracter;
Definir nombre, apellido Como Cadena;
```

```
letra ← ‘a’    nombre ← ‘Jose’    apellido ←‘Lopez’
```

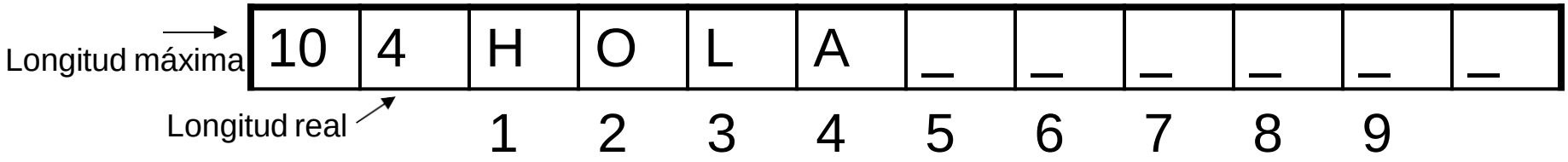
DATOS DE TIPO CADENA

- **Cadenas de longitud fija**:
  - La longitud de la cadena se tiene que definir antes de ser usada y siempre va a tener esa longitud, almacenándose en posiciones contiguas de memoria.
  - Si la cadena no ocupa todo el espacio, el resto se rellena con blancos, y esos blancos se consideran parte de la cadena.
  - Esto es muy deficiente y no se usa casi en ningún lenguaje.



DATOS DE TIPO CADENA

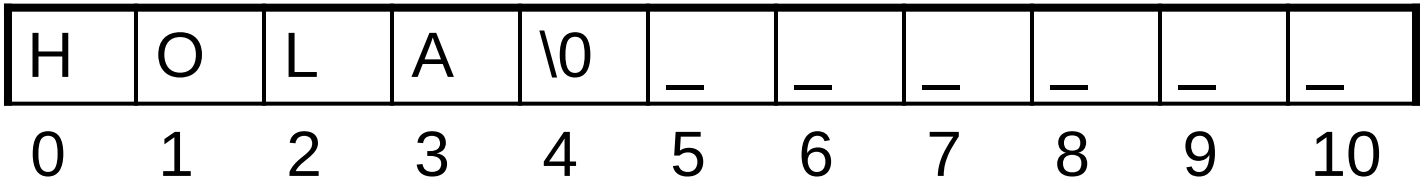
- **Cadenas de longitud variable**:
  - Antes de usar la cadena, hay que declarar la longitud máxima que puede tener. Ese es el espacio que se reserva en memoria para almacenar la cadena, siempre en posiciones contiguas en memoria.
  - La longitud real de la cadena durante la ejecución puede variar, aunque siempre tiene que ser menor que el máximo de la cadena.



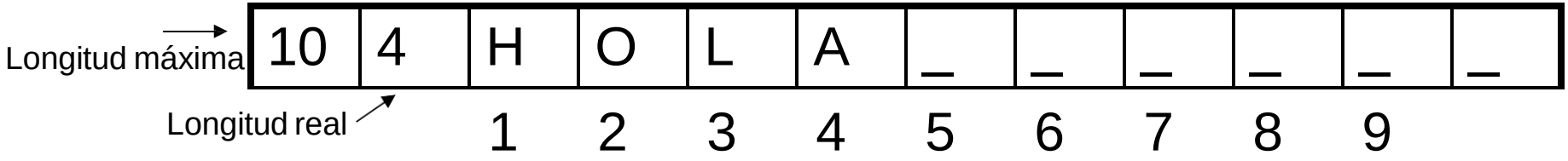
DATOS DE TIPO CADENA

- **Cadenas de longitud variable**:

  En C:

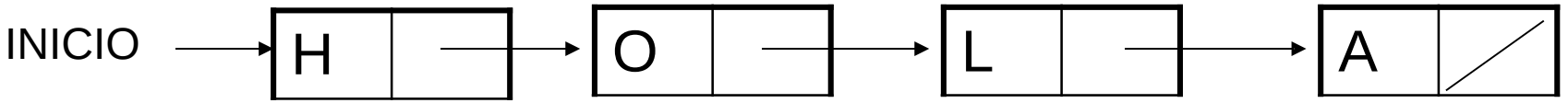


  En Pascal:



DATOS DE TIPO CADENA

- **Cadenas de longitud indefinida**:
  - No hay que definir la longitud de la cadena antes de usarla, ni siquiera la máxima. Para esto, se utiliza la memoria dinámica, y para establecer el número de elementos de la cadena usaremos **listas enlazadas**, en las que cada nodo de la lista contara un carácter de la cadena y se enlazaría mediante punteros.
  - La información no tiene que estar almacenada en posiciones contiguas de memoria.



OPERACIONES CON CADENAS

- Se pueden realizar operaciones básicas de asignación y entrada/salida (leer y escribir).
- Aparte de estas acciones, la mayor parte de los lenguajes permiten realizar operaciones especiales con las variables de tipo cadena:
  - Cálculo de la longitud.
  - Comparación.
  - Concatenación.
  - Extracción de *subcadenas*.
  - Búsqueda de información.

OPERACIONES CON CADENAS

- **ASIGNACIÓN**:

```
Definir nombre, apellidos Como Cadena;

nombre ← 'Jose'
apellidos ← 'Lopez Perez'
```

- **ENTRADA/SALIDA**:

```
Definir nombre, apellidos Como Cadena;

Leer nombre;
apellidos ← 'Lopez Perez';
Escribir apellidos;
```

OPERACIONES CON CADENAS

- **LONGITUD DE UNA CADENA:**
  - Es una función a la que se le pasa una cadena como parámetro y como resultado devuelve su longitud.

    Escribir Longitud(cadena)

  - Ejemplo:

```
nombre ← 'Jose'
apellidos ← 'Lopez Perez'
n <- Longitud (nombre)     // n = 4
m <- Longitud (apellidos) // m = 11
x <- Longitud (' ')       // x = 1
```

OPERACIONES CON CADENAS

- **COMPARACIÓN DE CADENAS:**
  - Las cadenas se pueden comparar entre sí usando los símbolos de comparación (< > =).
    - Internamente se comparan los **valores ASCII** asociados a cada carácter, uno a uno.
    - Resultado: VERDADERO o FALSO.

Ejemplos:

'A' < 'B'	'A' < 'a'
'8' < 'i'	'Jose' = 'Jose'
'pedro' < 'perez'	'luis' < 'luisito'
'Jose' < 'jose'	'jose' < 'jose '
' jose' < 'jose'	' ' < 'a'
'jose' < 'joses'	'0' < '00'

OPERACIONES CON CADENAS

- **COMPARACIÓN DE CADENAS:**
  - En la mayor parte de los lenguajes, hay una función que hace la comparación.
  - En C es la función `strcmp (C1,C2).`

```
funcion comparacion (C1:cadena;C2:cadena): entero
```

- En Java se emplea el método `compareTO().`

  - Esta función/método devuelve:
    - 0 si C1 = C2
    - Un positivo si C1 > C2
    - Un negativo si C1 < C2

Una letra mayúscula es distinta de una minúscula o de la misma letra con acento, por lo que para ordenar palabras no es demasiado efectivo.

OPERACIONES CON CADENAS

- **COMPARACIÓN DE CADENAS:**
  - Se pueden comparar cadenas sin tener en cuenta si contienen alguna letra mayúscula y/o minúscula.
  - Puede ser útil en situaciones donde la sensibilidad a la capitalización no es relevante.
  - ¿Cómo?: En los lenguajes de programación suele haber procedimientos para pasar todo el texto a letras mayúsculas o a minúsculas. A continuación se comparan.
  - En PseInt:

    Mayusculas(cadena)

    Minusculas(cadena)

OPERACIONES CON CADENAS

- **CONCATENACIÓN DE CADENAS:**
  - Permite unir varias cadenas en una sola, manteniendo el orden de los caracteres que se unen.
  - En pseudocódigo se usa el símbolo **&** o el **+**: `C1&C2` o `C1+C2`
  - Con PSeInt se emplea la función **`Concatenar().`**
    - Ejemplo:
      - C1 ← 'Hola'
      - C2 ← 'Adios'
      - C3 ← C1&C2 // C3='HolaAdios'
      - C4 ← C1&' Jose '&C2 // C4='Hola Jose Adios'
      - C5 ← C1+C4 // C5='HolaHola Jose Adios'

OPERACIONES CON CADENAS

- **SUBCADENAS:**

Consiste en extraer parte de una cadena.

- **SubCadena** (*cadena*, *inicio*, final)
  - *cadena* es la cadena original, de la que se extraerá la subcadena.
  - *inicio* es un entero que marca la posición a partir de la cual se extrae la subcadena.
  - *final* es un entero que marca la posición final.

- Ej: `Subcadena ('longitud', 3,6)  //devuelve gitu`
- `Subcadena ('longitud', 2,2)  //devuelve n`
- `Subcadena ('longitud', 1,3)  //devuelve ong`

Las posiciones utilizan la misma base que los arrays, por lo que la primera letra será la 0 o la 1 de acuerdo al perfil utilizado en PSeInt.

Ejercicio nº 16

Ejercicio nº 17 para casa

Ejercicio completo de operaciones con cadenas

OPERACIONES CON CADENAS

- **BÚSQUEDA EN CADENA:**
  - Consiste en:
    - Localizar si una determinada subcadena forma parte de otra cadena más grande y
    - Buscar la posición en la que aparece una subcadena.
  - Para ello basta definir una función denominada indice o posición:

```
Funcion resultado <- indice (cadena,subcadena)
```

    - Si el resultado es >= 0 entonces indica la posición del primer carácter de la primera coincidencia de la subcadena en la cadena original
    - Si el resultado es -1, la subcadena no aparece en la cadena original o la subcadena es vacía.

OPERACIONES CON CADENAS

- **INSERTAR CADENAS:**
  - **insertar** (*cadena*, *posición*, *subcadena*)
    - *cadena* es la cadena donde se insertará la subcadena
    - *posición* es la posición, comenzando por cero (0) a partir de la cual (se incluye) se va a insertar la subcadena
    - *subcadena* es la cadena que se va a insertar.
      - Ej: insertar (‘longitud’, 2,’Jose’) //devuelve ‘loJosengitud’

OPERACIONES CON CADENAS

- **INSERTAR CADENAS:**
  - **Algoritmo:**

```
Funcion insertar (cad Por Referencia, posicion, subcad)
    Definir aux Como Cadena;
    aux <- "";
    aux <- Subcadena(cad, 0, posicion-1);
    aux <- Concatenar(aux, subcad);
    aux <- Concatenar(aux, subcadena(cad, posición, Longitud(cad)));

    cad <- aux;
FinFuncion
```

OPERACIONES CON CADENAS

- **BORRAR CADENAS:**
  - **borrar** (*cadena*, *posición*, *longitud*)
    - *cadena* es la cadena donde se va a borrar
    - *posición* es la posición a partir de la cual (ésta se incluye) se va a borrar
    - *longitud* es la cantidad de caracteres a borrar.
      - Ej: borrar (‘longitud’, 2,3) //devuelve ‘lotud’

OPERACIONES CON CADENAS

- **BORRAR CADENAS:**
  - **Algoritmo:**

```
Funcion borrar (cad Por Referencia, pos , longitud)
    Definir aux Como Cadena;

    aux ← Subcadena(cad,0,pos-1);
    aux ← Concatenar (aux, Subcadena(cad,pos+long,
    Longitud(cad)-1));

    cad <- aux;

FinFuncion
```

OPERACIONES CON CADENAS

- **CAMBIAR CADENAS:**
  - **cambiar** (*cadena*, *subcadena1*, *subcadena2*)
    - *cadena* es la cadena donde se producirán los cambios
    - *subcadena1* es la subcadena a sustituir
    - *subcadena2* es la subcadena nueva

      - Ej: cambiar (‘longitud’, ‘tu’,’vida’) //devuelve ‘longividad’

OPERACIONES CON CADENAS

- **CAMBIAR CADENAS:**
  - **Algoritmo:**

```
Funcion cambiar(cad Por Referencia, sub1, sub2)
    Definir aux Como Cadena;
    Definir pos Como Entero;
    Aux <- "";
    pos ← indice(cad, sub1);

    borrar(cad, pos, Longitud(sub1));
    insertar(cad, pos, sub2);

FinFuncion
```

OPERACIONES CON CADENAS

- **CONVERSIÓN CADENA/NÚMERO:**

  Estas funciones están en PseInt:

  – **ConvertirANumero** (*cadena)*

    - Convierte la cadena en un número, si es posible.

      Ej: ConvertirANumero(‘2341’) devuelve el 2341

  – **ConvertirATexto** (*numero)*

    - Convierte un número en una cadena.

      Ej: ConvertirATexto(2341) devuelve la cadena ‘2341’