Fundamentos de Programación I

Tema 7. Cadenas de caracteres

Luís Rodríguez Baena (luis.rodriguez@upsam.net)

Universidad Pontificia de Salamanca (campus Madrid)
Escuela Superior de Ingeniería y Arquitectura

Datos alfanuméricos

- Dos tipos de datos definidos para trabajar con información no numérica ni booleana:
  - Carácter.
  - Cadena.
- Para tratarlos es necesario convertir la información no numérica con la que trabaja el ordenador.
  - Asignación de un código numérico a cada carácter.
  - Esa asignación se puede realizar de distintas formas.
  - No todos los caracteres están disponibles siempre.
  - Al conjunto de caracteres con los que puede trabajar un ordenador se le denomina **juego de caracteres.**

El juego de caracteres

- El juego de caracteres ASCII (*American Standard Code for Information Interchange*).
  - Utiliza un byte de información.
    - ✓ Se utilizan 7 bits para representar el carácter.
  - Sólo es posible representar 128 caracteres, del 0 al 127.
    - ✓ Del carácter 0 a 32: caracteres de control (retorno de carro, tabulación, final de archivo, carácter nulo, etc.).
    - ✓ Del carácter 48 al 57 los dígitos del 0 al 9.
    - ✓ Del carácter 65 al 90 los caracteres alfabéticos en mayúsculas.
    - ✓ Del carácter 97 al 122 los caracteres alfabéticos en minúsculas.
    - ✓ El resto de posiciones lo ocupan caracteres especiales como el espacio en blanco, la coma, el punto, operadores aritméticos, etc.
  - Los caracteres alfabéticos están restringidos a los que se utilizan en el idioma inglés.

El juego de caracteres (II)

- Juego de caracteres ASCII extendido.
  - Soluciona estas restricciones permitiendo la representación de caracteres regionales.
  - Utiliza los 8 bits para representar todos los caracteres.
    - ✓ Los caracteres de 32 al 126 utilizan los caracteres imprimibles ASCII estándar, así cómo los códigos de control 8 a 13.
    - ✓ Los códigos superiores a 127 se utilizan para representar los caracteres ampliados.
  - Existen distintas codificaciones para esos caracteres ampliados:
    - ✓ Página de códigos 437 (para las versiones en inglés de MS-DOS).
    - ✓ Página de códigos 850 (para las versiones de Europa occidental de MS-DOS)

El juego de caracteres (III)

- El juego de caracteres EBCDIC (*Extended Binary Coded Decimal Interchange Code*).
  - Desarrollado en los años 60 por IBM.
  - Utiliza 8 bits por carácter: 256 caracteres posibles.
  - Distribuye los caracteres de forma distinta: minúsculas, mayúsculas y dígitos.
- Juego de caracteres Unicode.
  - Puede utilizar hasta cuatro bytes para almacenar cada carácter.
  - Asigna un número a cada posible carácter de cada idioma.
    - ✓ Cubre casi todas las escrituras actuales (escrituras orientales, braille, cheroke…).
  - Admite distintas codificaciones UTF-7, UTF-8, UTF-16 y UTF-32.
    - ✓ El sistema de archivos NTFS de Windows utiliza UTF-16.
    - ✓ El juego de caracteres Unicode ISO-8859-1 o Latin-1 es el utilizado en Europa Occidental.
    - ✓ En sus primeros 127 caracteres se corresponde con el código ASCII.

Cadenas de caracteres

- Dato de tipo carácter.
  - Un único carácter del juego de caracteres utilizado.
    - ✓ En algunos lenguajes se asimila a un dato de tipo entero correspondiente al código numérico del carácter.
  - Una constante de tipo carácter estaría formada por un carácter encerrado entre comillas.
    - ✓ Por ejemplo, `'C'`, `'~'`, `'1'`.
    - ✓ La función **`aCarácter`**`()` devuelve el carácter correspondiente al código que pasamos entre paréntesis.
      - Por ejemplo, **`aCarácter`**`(13)` devolvería el carácter correspondiente al retorno de carro.
- Datos de tipo cadena.
  - Un conjunto de caracteres del juego de caracteres que se esté utilizando.
  - El lenguaje algorítmico UPSAM considera compatibles las cadenas y los caracteres.
    - ✓ Es posible asignar cadenas a caracteres o viceversa, se pueden comparar datos de tipo cadena con datos de tipo carácter, etc.
      - Esto no ocurre en otros lenguajes como Java o C.

Cadenas de caracteres (II)

- Constante de cadena.
  - Una serie de caracteres delimitados por el separador de cadenas que puede ser la comilla simple o doble.
- Carácter nulo.
  - El carácter correspondiente al código 0.
    - ✓ Es un carácter válido aunque no imprimible.
    - ✓ Es distinto del espacio en blanco.
      - El espacio en blanco tiene un código ASCII 32.
- Cadena nula.
  - Cadena que sólo contiene un carácter nulo.
    - ✓ Tiene longitud 0.
- Subcadena.
  - Un fragmento de una cadena principal.
  - Puede estar formado por una parte de la cadena, por toda la cadena o por ningún carácter de la cadena (se trataría de una subcadena nula).

Variables de tipo cadena

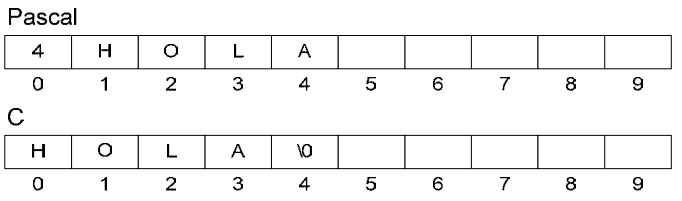
- Las cadenas se consideran un tipo de dato estándar.
  - La declaración de una variable de tipo cadena se realizaría utilizando la palabra reservada **`cadena`**.

```
var
   cadena : nombre, apellidos
```

    - ✓ En el lenguaje algorítmico UPSAM no es necesario especificar la longitud de la cadena.
- Dependiendo de la forma de almacenamiento en memoria las cadenas pueden ser:
  - Estáticas.
  - Semiestáticas.
  - Dinámicas.

Variables de tipo cadena (II)

- Cadenas estáticas.
  - Define su longitud en tiempo de compilación, es decir en la declaración de la variable.
  - Son las cadenas que utiliza C o Pascal.
    - ✓ Las cadenas estarían formadas por un array de caracteres que se trata de forma especial.
      - En Pascal, el tamaño se limita a 255 caracteres y el elemento 0 del array contendría el número de caracteres.
      - En C la longitud máxima está indefinida y el último carácter sería el carácter nulo (\0).



Declaración en Pascal

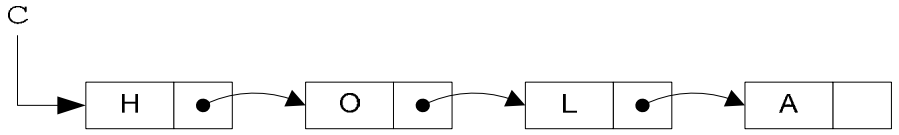
```
var
   cad : string[9];
...
   cad := 'hola';
```

Declaración en C

```
   char car[9];
   char *nombre;
...
   cad = "hola";
   nombre = "Pepe";
```

Variables de tipo cadena (III)

- Cadenas semiestáticas (o semidinámicas).
  - Cadenas de longitud variable con un tamaño máximo.
    - ✓ Su tamaño no se establece en la declaración: aumenta o disminuye según las necesidades.
  - BASIC utiliza este modelo: presenta cadenas de longitud variable con un tamaño máximo de 32.767 caracteres.
- Cadenas estáticas.
  - Cadenas de longitud totalmente variable.
    - ✓ Se almacenen en forma de lista enlazada y los caracteres estarían dispersos por la memoria.



Variables de tipo cadena (IV)

- En el lenguaje algorítmico UPSAM la forma de almacenamiento no es importante.
  - No hay que declarar el tamaño.
  - Es posible tratar la cadena:
    - ✓ Como un conjunto de caracteres mediante el nombre de la variable.
    - ✓ Los caracteres individuales como si fueran elementos de un array.
    - ✓ Ejemplo: Escribir la cadena 'HOLA' al revés:

```
var
   cadena : c
   entero : i
...
c ← 'HOLA'
desde i ← 4 hasta 1 incremento  -1 hacer
   escribir(c[i])
fin_desde
...
```

Operaciones con cadena

- Instrucciones de asignación y entrada/salida.
  - Funcionan igual que con cualquier otro tipo de dato estándar.
- Operaciones de relación.
  - Comparan el código de los caracteres que forman el carácter o la cadena.
    - ✓ La expresión `'A' < 'B'` compara el código de la A (65) con el de la B (66).
      - La expresión sería verdadera porque 65 < 66.
    - ✓ Es importante saber el orden de la codificación del juego de caracteres que se esté utilizando.
      - En ASCII
        - 1º Dígitos.
        - 2º Mayúsculas ordenadas de menor a mayor.
        - 3º Minúsculas ordenadas de menor a mayor.
        - 4º Caracteres especiales y regionales (ñ, Ñ, vocales acentuadas, etc.).
  - La comparación de cadenas compara uno por uno y de izquierda a derecha los códigos de cada uno de los caracteres de las cadenas de la operación.
    - ✓ El proceso acaba cuando se encuentra un carácter distinto.
      - 'MALAGA' > 'MADRID'; los primeros caracteres distintos son la L y la D, como L es mayor que D, la cadena MALAGA es mayor que MADRID.
      - 'MADRID' < 'MADRIDEJOS', la ausencia de carácter es menor que la presencia: MADRIDEJOS tiene los mismos caracteres que MADRID y algunos más, por lo que es mayor.

Operaciones con cadena (II)

- Concatenación.
  - Consiste en unir dos cadenas para obtener otra cadena.
    - ✓ En el lenguaje UPSAM se utiliza el operador + o, preferiblemente, el operador &.
  - Se pueden utilizar acumuladores de caracteres de forma similar a los acumuladores numéricos.

```
//Lee caracteres hasta pulsar la tecla Intro y devuelve una
//cadena con la unión los caracteres leídos.
cadena función LeerCadenaCarácterACarácter()
var
   cadena : cad
   carácter : car
inicio
   cad ← ''    //Se inicializa la cadena a nulo
   leer(car)
   mientras carácter(13) <> car hacer //Mientras que car no sea Intro
      cad ← cad & car
      leer(car)
   fin_mientras
   devolver(cad)
fin_función
```

Funciones de cadena

- Se consideran tres funciones básicas para trabajar con cadenas:
  - Función `longitud()`.
  - Función `posición()`.
  - Función `subcadena()`.

Función longitud()

- Devuelve un número entero con el número de caracteres de la cadena.
- Formato:

  `longitud``(`*`expresiónDeCadena`*`)`

  - ✓ *ExpresiónDeCadena* es cualquier expresión que devuelva como resultado una dadena.
  - ✓ La función devuelve un entero positivo con el número de caracteres, o 0 si la expresión es una cadena nula.

Función longitud() (II)

- Ejemplos:

longitud('Hola')	Devuelve 4
longitud('Hola' & 'Adios')	Devuelve 9
cad ← 'Cocodrilos' **longitud**(cad)	Devuelve 10
cad1 ← 'ABCD' cad2 ← ' ' //espacio en blanco **longitud**(cad1 & cad2)	Devuelve 5
cad ← '' //cadena nula **longitud**(cad)	Devuelve 0

Función longitud() (III)

```
//Recorre una cadena carácter a carácter
var
   cadena : cad
   entero : i, numMayúsculas
...
   leer(cad)
   numMayúsculas ← 0
   desde i ← 1 hasta longitud(cad) hacer
      si (cad[i] >= 'A') y (cad[i] <= 'Z') entonces
         numMayúsculas ← numMayúsculas + 1
      fin_si
   fin_desde
   escribir(numMayúsculas)
...
```

```
//Verifica si se ha introducido algún carácter distinto de nulo
//en alguna operación de entrada.
repetir
   leer(cad)
hasta_que  longitud(cad) <> 0
```

Función posición()

- Busca una subcadena dentro de una cadena principal.
  - Devuelve un número entero positivo con la posición de la subcadena en la cadena principal.
  - Devuelve 0 si la subcadena no se encuentra.
  - Formato:

    posición(*cadenaPrincipal*, *cadenaBuscada*)

    - *cadenaPrincipal* y *cadenaBuscada* son dos expresiones de tipo cadena.

cad ← 'cocodrilo' **posición**(cad,'o')	Devuelve 2
posición(cad,'co')	Devuelve 3
posición(cad,'drilo')	Devuelve 5
posición(cad, 'a')	Devuelve 0

Función posición() (II)

- Ejemplos:

```
//Validar una entrada de usuario.
//Pide caracteres hasta que el usuario introduce una s o una n.
...
repetir
   escribir('¿Desea continuar (S/N)?')
   leer(opción)
hasta_que posición('SsNn',opción) <> 0
...
```

```
//Cuenta el número de vocales de la cadena cad
...
desde i ← 1 hasta longitud(cad) hacer
   si posición('AEIOUaeiouáéíóú',cad[i]) <> 0 entonces
      contadorVocales ← contadorVocales + 1
   fin_si
fin_desde
...
```

Función subcadena()

- Devuelve una subcadena a partir de una cadena principal.

  subcadena(*cadenaPrincipal*,*posiciónInicial*,
  [*longitudSubcadena*])

  - ✓ *cadenaPrincipal* es cualquier expresión de cadena.
  - ✓ *posiciónInicial* es un entero positivo con el carácter a partir del cual se obtiene la subcadena.
  - ✓ *longitudSubcadena* es un número entero mayor o igual que 0 con la longitud de la subcadena.

- Devuelve una cadena:
  - Si *longitudSubcadena* se omite, la subcadena estará formada por todos los caracteres hasta el final de la cadena principal.
  - Si *longitudSubcadena* es igual a 0, devuelve una cadena nula.
  - Si *longitudSubcadena* sobrepasa la longitud total de la cadena, devolverá todos los caracteres desde *posiciónInicial*.
  - Si *posiciónInicial* es mayor que la longitud total de la cadena devuelve una cadena nula.

Función subcadena() (II)

`cad ← 'cocodrilo'` **subcadena**`(cad,3)`	Devuelve 'codrilo'
subcadena`(cad,3,4)`	Devuelve 'codr'
subcadena`(cad,5,8)`	Devuelve 'drilo'
subcadena`(cad, 5,0)`	Devuelve una cadena nula
subcadena`(cad,10)`	Devuelve una cadena nula

Otras funciones de cadena

- insertar.
  - Devuelve una cadena en la que se inserta una subcadena dentro de la cadena principal a partir de una posición determinada.

    insertar(*cadenaPrincial*, *posición*, *subcadenaAInsertar*)

    ✓ **insertar**('Miguel Saavedra', 8, 'de Cervantes') devuelve la cadena 'Miguel de Cervantes Saavedra'.
  - Si el lenguaje utilizado no dispone de esta función se puede implementar utilizando las funciones básicas.

```
cadena función Insertar(valor cadena : c; valor entero : p; valor cadena : sc)
inicio
   si p <= 0 entonces
      devolver(c)
   si_no
      devolver(subcadena(c,1,p-1) & sc & subcadena(c,p))
   fin_si
fin_función
```

Otras funciones de cadena (II)

- Borrar.
  - Devuelve una cadena en la que se ha eliminado una subcadena a partir de una posición dada y de una longitud determinada.
    - ✓ **borrar**(*cadenaPrincipal*, *posiciónInicio*, *longitud*)
    - ✓ **borrar**('La Casa Blanca', 4, 5) devolvería la cadena 'La Blanca'.
  - Si el lenguaje utilizado no dispone de esta función se puede implementar con las funciones básicas.

```
cadena función Borrar(valor cadena : c ; valor entero : p,n)
inicio
   si (p <= 0) o (n <= 0) entonces
      devolver(c)
   si_no
      devolver(subcadena(c,1,p-1) & subcadena(c, p+n))
   fin_si
fin_función
```

Otras funciones de cadena (III)

- Cambiar.
  - Sustituye la primera aparición de una subcadena por otra.
    cambiar(*cadenaPrincipal*,*cadenaBuscada*,*cadenaASustituir*)
    - ✓ **cambiar**('La Casa Blanca', 'Casa', 'Cabaña') devolverá la cadena 'La Cabaña Blanca'.
    - ✓ Si la cadena buscada no se encuentra devuelve la cadena principal.
  - Si el lenguaje utilizado no dispone de esta función, se puede implementar mediante las funciones básicas.

```
cadena función Cambiar(valor cadena:c,borrada,sustituida )
var
   entero : p
inicio
   p ← posición(c,borrada)
   si p = 0 entonces
      devolver(c)
   si_no
      devolver(Insertar(Borrar(c,p,longitud(borrada)),p,sustituida)
   fin_si
fin_función
```

Funciones de conversión

- Función **`valor()`.**
  - Recibe como argumento una expresión de cadena y la convierte en un valor numérico.
    - ✓ Si la conversión no es posible devuelve 0.

valor('2345')	Devuelve el dato entero 2345
valor('12.67')	Devuelve el dato real 12.67
valor('Hola')	Devuelve 0
valor('12A34B')	Devuelve 0

- Función **`aCadena()`.**
  - Recibe como argumento una expresión numérica y la convierte en una cadena.

aCadena(12678)	Devuelve la cadena '12678'
aCadena(12.67)	Devuelve la cadena '12.67'
aCadena(3+6.5)	Devuelve la cadena '9.5'

Funciones de conversión (II)

- Función **`aCarácter`**`()`**.**
  - Devuelve el carácter correspondiente al código numérico que se pasa como argumento según el juego de caracteres utilizado.
- Función **`código`**`()`**.**
  - Devuelve el código numérico asociado al carácter que se pasa como argumento.

`aCarácter``(65)`	Devuelve el carácter A
`código``('B')`	Devuelve el valor entero 66

Ejercicios

1. Complete la biblioteca de funciones de cadena implementando funciones que realicen las siguientes operaciones:
   - Extraer los n primeros caracteres de una cadena.
   - Contar el número de veces que aparece una cadena dentro de otra
   - Borrar todas las apariciones de una cadena dentro de otra.
   - Sustituir todas las apariciones de una cadena dentro de otra, por una tercera.
   - Convertir una cadena en mayúsculas. En el código ASCII, la diferencia entre un carácter estándar en mayúscula y otro en minúsculas es de 32: si la A mayúcula tiene un código 65, la a minúscula tiene un código 97. Esto sería válido para todos los caracteres cuyo código sea que 127. Para el resto de los caracteres (vocales acentuadas, la eñe) sería necesario convertirlo uno a uno
   - Convertir una cadena en minúsculas.

Ejercicios (II)

2. Diseñe una función que permita cambiar un número n en base 10 a otra base de numeración b, siendo b un número entre 2 y 16.
3. Diseñe una función que reciba una cadena y devuelva un valor lógico indicando si esa cadena es un palíndromo.
   - Un palíndromo es una frase que se lee igual de izquierda a derecha que de derecha a izquierda como "a cavar a caravaca", "amo la pacífica paloma", "la ruta natural", "Isaac no ronca asi", "yo de lo mínimo le doy".
4. Diseñe un algoritmo que convierta un número decimal a números romanos.