# Cadenas, Arreglos, Punteros, Postincremento & Preincremento

Esp. Ing. José María Sola, profesor.

Revisión 0.1.0
2017-06-04

# Tabla de contenidos

# 1

# Introducción

El Lenguaje C no posee el tipo de dato *string* pero usa una *convención* para representar strings: *arreglo* de *caracteres* donde el primer *caracter nulo* (`'\0'`) indica el fin de la cadena. El manejo de strings requiere entonces el conocimiento de arreglos, y en C, los arreglos están ítimamente relacionados con los *punteros*. Para recorrer una cadena es necesario incrementar un índice o un puntero, y el uso de *pre* o *pos incremento* resulta en una semántica diferente. El objetivo de este artículo es presentar una aplicación de estos conceptos.

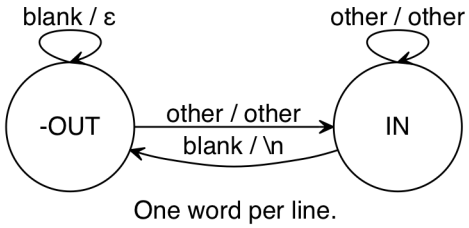
# 2

# Caso de estudio

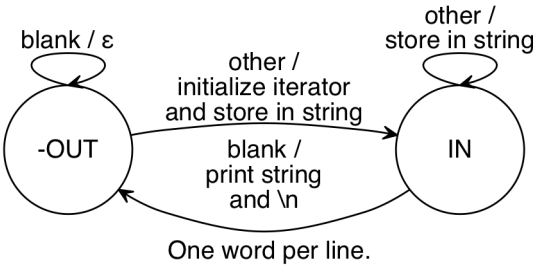
Nuestro caso de estudio está basado en el ejercicio 1-12 de [KR1988]:

> "1-12. Escriba un programa que imprima su entrada una palabra por línea."

La forma más directa y simple de resolver este problema es construir una *máquina de estados accionanadora* tipo *Mealy* que, mientras esté en el estado *IN* ("dentro de Palabra") imprima cada carácter y cuando salga de *IN* a *OUT* ("fuera de palabra") imprima el caracter nueva línea (`'\n'`).




Pero en este caso de estudio almacenamos todos los caracteres de la palabra en un string, y al salir de *IN*, imprimimos el string completo que forma la palabra, seguido del `'\n'`, ya que el objetivo es ejemplificar el recorrido de un arreglo con índice con puntero, y el uso de pre y posincremento.


blank / ε
other /
store in string
other /
initialize iterator
and store in string
-OUT
IN
blank /
print string
and \n
One word per line.

# 3

# Cuatro formas de iterar el string

El string lo almacenamos en el arreglo `a`. Al ser un arreglo debemos especificar su tamaño el cual va a ser la longitud máxima de una palabra más un carácter para el cárcger nulo (`'\0'`):

```
char a[MAX+1];
```

`MAX` es una constante.

Esta declaración impone una restricción al tamaño de las palabras que el programa puede procesar. La solución más directa, que es imprimir a medida se lee, no tienen tal restricción.

Las acciones de nuestra máquinq de mealy son:

- Inicilizar el iterador.
- Almacenar el carácter en el string y avanzar.
- Imprimir el string.

La forma de iterar el string son con iteradores tipo:

- índice, que es el offset desde la base del arreglo.
- puntero, que apunta a un caracter del arreglo.

La formas de avanzar el iterador son:

- usarlo en una sentencia y luego preincrementarlo en otras.

- usarlo y posincrementarlo en la misma sentencia.

Las alternativas se resumen en la siguiente tabla:

| Estado | Índice | Índice con incremento | Puntero | Puntero con incremento |
|---|---|---|---|---|
| **Declaración del string** | `char a[MAX+1];` | | | |
| **Declaración del iterador** | `int i;` | | `char *p;` | |
| **Inicializar iterador** | `i=0;` | | `p=a;` | |
| **Almacenar caracter en string** | `a[i]=c;`<br>`++i;` | `a[i++]=c;` | `*p=c;`<br>`++p;` | `*p++=c;` |
| **Imprimir srting** | `printf("%s\n",a);` | | | |

Analice cada una de las cuatro variantes y responda con justificación:

1. ¿Por qué `p=a;` es semánticamente correcta?
2. ¿Cuál es la semántica de `a[i++]=c`? Dibuje su árbol de expresión.
3. ¿Cuál es la semántica de `*p++=c`? Dibuje su árbol de expresión.
4. ¿Cuál de las cuatro versiones es más legible?
5. ¿Cuál es más eficiente en espacio?
6. ¿Cuál es más eficiente en tiempo?
7. ¿Cuántas operaciones realiza cada una?
8. ¿Cuál aprovecha más las características del lenguaje C?

# Bibliografía

[KR1988] Brian W. Kernighan and Dennis Ritchie. The C Programming Language, 2nd Edition. 1988.