



### 1. Recorrido de una cadena de caracteres representada como vector

```
/*  
 * Pre: La cadena «cad» finaliza con una componente cuyo valor es el carácter nulo ('\0').  
 * Post: Ha devuelto el número de componentes de la cadena «cad» cuyo valor  
 *       es el de una letra mayúscula o minúscula del alfabeto inglés.  
 */  
int contarLetras(const char cad[]);
```

### 2. Problema 2 del examen de 2.<sup>a</sup> convocatoria del curso 2017-18 (1 punto)

El cuerpo de la función concatenar que se presenta a continuación contiene algunos errores. La cabecera y especificación de la misma son, en todo caso, correctas y no han de ser modificadas. Las funciones presentes en el módulo de biblioteca cstring son visibles en el código que se presenta a través de su inclusión en la directiva `#include` adecuada.

```
/*  
 * Pre: La cadena de caracteres «resultado» ha sido declarada con al menos strlen(prefijo) +  
 *       + strlen(sufijo) + 3 caracteres, «prefijo» y «sufijo» son cadenas de caracteres acabadas  
 *       en el carácter nulo '\0' y «numero» es positivo y tiene como mucho dos dígitos cuando se  
 *       escribe en base 10.  
 * Post: «resultado» es la cadena de caracteres acabada en el carácter nulo '\0' resultante de  
 *       concatenar la cadena «prefijo» con la representación decimal de dos dígitos de «numero»  
 *       y con la cadena «sufijo».  
 */  
void concatenar(char resultado[], const char prefijo[], const int numero, const char sufijo[]) {  
    strcpy(resultado, prefijo);  
    int i = strlen(prefijo);  
    resultado[i] = numero % 10 + '0';  
    resultado[i + 1] = numero / 10 + '0';  
    resultado[i + 2] = '\0';  
    strcat(resultado, sufijo);  
    return resultado;  
}
```

Identifica los errores presentes en el código, indica con claridad y precisión en qué consisten y escribe el código corregido de la función concatenar.

### 3. Funciones predefinidas para trabajar con cadenas de caracteres

Implementa las siguientes funciones, equivalentes a las definidas en la biblioteca cstring para la gestión de cadenas de caracteres representadas como vectores. Hazlo simplemente como ejercicio (es decir, cuando programes, nunca utilices tu implementación de las funciones de este problema en lugar de las que ya están predefinidas en cstring).

```
/*  
 * Pre: La cadena «cadena» finaliza con una componente cuyo valor es el carácter nulo ('\0').  
 * Post: Ha devuelto la longitud de la cadena «cadena», tal y como lo haría la función «strlen»  
 *       de la biblioteca predefinida «cstring».  
 */  
int longitud(const char cadena[]);
```

```
/*  
 * Pre: La cadena «origen» finaliza con una componente cuyo valor es el carácter nulo ('\0').  
 *       Sea n el número de caracteres de la cadena «origen»: la cadena «destino» tiene una  
 *       dimensión igual o superior a n caracteres.  
 * Post: Ha copiado los n caracteres de «origen» en «destino» y ha acabado «destino» con un  
 *       carácter nulo ('\0'), tal y como lo haría la función «strcpy» de la biblioteca  
 *       predefinida «cstring».  
 */  
void copiar(char destino[], const char origen[]);
```



# Programación 1

## Cadenas de caracteres

```
/*
 * Pre: La cadena «origen» finaliza con una componente cuyo valor es el carácter nulo ('\0').
 * Sea n el número de caracteres de la cadena «origen»: la cadena «destino» tiene una
 * dimensión suficiente como para añadirle los n caracteres de la cadena «origen».
 * Post: Ha concatenado los n caracteres de «origen» en «destino», a partir del último carácter
 * que «destino» tenía inicialmente y ha acabado «destino» con un carácter nulo ('\0'), tal
 * y como lo haría la función «strcat» de la biblioteca predefinida «cstring».
 */
void concatenar(char destino[], const char origen[]);
```

```
/*
 * Pre: Las cadenas «cad1» y «cad2» finalizan cada una con una componente cuyo valor es el
 * carácter nulo ('\0'). Sea n el número de caracteres de la cadena «cad1» y m el número de
 * caracteres de la cadena «cad2».
 * Post: Ha comparado las cadenas «cad1» y «cad2», carácter a carácter desde la componente 0.
 * Ha devuelto 0 si el contenido de ambas es idéntico (es decir, si n=m y todos los
 * caracteres de «cad1» coinciden con los de «cad2»). Ha devuelto un valor positivo si en
 * la primera discrepancia entre ambas el carácter de la cadena «cad1» tiene un código
 * mayor que el de «cad2» y ha devuelto un valor negativo si en la primera discrepancia
 * entre ambas el carácter de la cadena «cad1» tiene un código ASCII menor que el de la
 * cadena «cad2», tal y como lo haría la función «strcmp» de la biblioteca predefinida
 * «cstring».
 */
int comparar(const char cad1[], const char cad2[]);
```

#### 4. Problema 3 del examen de 1.ª convocatoria del curso 2018-19 (2,5 puntos)

Escribe el código de la función limpiar cuya especificación se muestra a continuación:

```
/*
 * Pre: «palabra» es una cadena de caracteres acabada en el carácter '\0' y de menos de MAX_LONG
 * caracteres que no contiene ningún carácter blanco (ni espacios en blanco, tabulaciones o
 * caracteres de cambio de línea).
 * Post: Cuando se ha terminado de ejecutar esta función, la cadena de caracteres «palabra»
 * contiene únicamente letras minúsculas del alfabeto inglés, correspondientes a las que
 * contenía inicialmente (cuando se comenzó a ejecutar esta función). Cualquier otro
 * carácter ha sido eliminado.
 *
 * Ejemplos:
 *
 * Valor inicial de «palabra»      Valor final de «palabra»
 * -----
 * ""                               ""
 * "En"                             "en"
 * "un"                             "un"
 * "Mancha, "                       "mancha"
 * "corredor."                      "corredor"
 * "- ¡Oh!"                         "oh"
 * "¿Duermen?"                     "duermen"
 * "1604"                           ""
 * "H2S04"                         "hso"
 */
void limpiar(char palabra[]);
```

De la solución que se presente se valorará, entre otros aspectos, que se trabaje directamente sobre el parámetro palabra sin utilizar cadenas de caracteres adicionales.