Práctica 10. Cadenas de caracteres en C++

El tipo de dato cadena de seudocódigo no existe como tal en C++. Para este lenguaje, al igual que para C, una cadena no es más que un vector de elementos de tipo carácter con una característica especial: el último elemento del vector debe ser el carácter especial '\0'. Esta marca es utilizada por el compilador como marca de final de la cadena: al encontrar el '\0', el compilador sabe que la cadena que está tratando ha terminado. La necesidad de introducción de este carácter especial obliga a que tengamos cuidado al declarar vectores de caracteres que vayan a almacenar cadenas. Si queremos almacenar una cadena que tiene 10 caracteres, debemos declarar un vector de tamaño 11 para que se pueda guardar también el '\0'.

Es necesario observar que las cadenas son vectores de caracteres, pero que no necesariamente todos los vectores de caracteres son cadenas: para que un vector de caracteres sea una cadena debe tener el carácter '\0' como último elemento.

A pesar de no existir un tipo de dato cadena, C++ sí que reconoce constantes de cadena, que son los caracteres que aparecen entre comillas dobles (“). Por ejemplo, “Hola”.

Inicialización de vectores para almacenar cadenas

Si al declarar un vector de caracteres deseamos darle un valor inicial de tipo cadena, tenemos dos posibilidades:

1. Especificar la cadena entre comillas dobles, omitiendo el '\0', aunque manteniendo el tamaño del vector como si estuviera:

   char cadena[5]=”Hola”;

2. Especificar la cadena entre comillas dobles, omitiendo el '\0', y sin especificar el tamaño para el vector. El compilador tomará el tamaño adecuado, reservando también espacio para el '\0':

   char cadena[]=”Hola”;

Aunque la segunda forma es la más recomendable, ya que nos evita tener que contar los caracteres de la cadena con que estamos inicializando, la primera será necesaria cuando queramos que el vector tenga un tamaño mayor que el de la cadena inicial (si por ejemplo pensamos que más adelante en el programa se pueda almacenar una cadena de más caracteres en el mismo vector).

Lectura de cadenas. Funciones de C para el manejo de cadenas.

El operador cin, presenta un problema a la hora de introducir una cadena de caracteres por teclado y es que no admite espacios en blanco (recordar que cin utiliza los espacios en blanco como separadores, los lee pero no los almacena). Observa el siguiente ejemplo:

```
char cadena1[15], cadena2[15];
cin>>cadena1;
cin>>cadena2;
```

Si introducimos por teclado la cadena HOLA A TODOS, tras la primera lectura el valor de cadena será "HOLA", es decir, que se omite todo lo que va detrás del primer espacio en blanco. Esto es debido a que los espacios en blanco actúan como marca de fin de cadena. En la segunda lectura, el valor que se lee para la cadena sería "A".

Para evitar este problema, utilizaremos la función **gets(<cadena>)** para lectura, definida en el fichero de cabecera **stdio.h** del lenguaje C. Por ejemplo,

```
char cadena[20];
gets(cadena);
```

- **Función strlen (string length, longitud de cadena)**

Devuelve el número de caracteres de la cadena que se le pasa como parámetro sin contar el carácter '\0'. Por ejemplo, strlen("Hola") devuelve 4.

- **Función strcpy (string copy, copia de cadena)**

Excepto en el momento de la inicialización, en C no es posible dar valor a una variable cadena en una sentencia de asignación. Así,

```
char cadena[10];
cadena="Hola";        /*¡¡¡ERROR!!!*/
```

es incorrecto. La única forma de hacer esto es a través de la función **strcpy**, que copia en la variable cadena que se le pasa como primer parámetro el valor de la cadena que se le pasa como segundo parámetro. La forma correcta de hacer la asignación anterior es pues:

```
char cadena[10];
strcpy(cadena,"Hola");   /*¡¡¡CORRECTO!!!*/
```

- **Función strcmp (string compare, comparación de cadenas)**

Sirve para comparar dos cadenas lexicográficamente (es decir, por orden alfabético). No compara cadenas por tamaño. Devuelve un entero negativo si la primera es menor que la segunda, un entero positivo si la primera es mayor que la segunda y cero si las dos cadenas son la misma.

```
strcmp("Ala","Hola");    /*Devuelve un valor <0*/
strcmp("Hola","Ala");    /*Devuelve un valor >0*/
strcmp("Hola","Hola");   /*Devuelve 0*/
```

Todas ellas tienen su prototipo en el fichero de cabecera <**string.h**>.

EJERCICIOS:

1) **Traduce el siguiente algoritmo a C++ (dado un dni, calcula la letra del NIF correspondiente):**

```
algoritmo calculoNIF
variables
        dni : entero
        letranif :carácter
principio
        leer(dni)
        letranif←calculoletranif(dni)
        escribir(letranif)
fin

función calculoletranif (dni :entero) devuelve carácter
variables
        tabla : vector[1..23] de carácter
        resto : entero
principio
        {inicializar tabla a "TRWAGMYFPDXBNJZSQVHLCKE"}
        resto←dni MOD 23
        devuelve(tabla[resto+1])
fin
```

2) **Diseñar un programa que lea una cadena de caracteres y la invierta.**

3) **Crear un programa que compruebe si un texto introducido por teclado es o no un palíndromo.**