Ejercicios resueltos de cadenas (Tema 3)

A continuación se presenta el código fuente de los problemas de cadenas del tema 3. Comparadlos con vuestras soluciones y plantead en clase cualquier duda o modificación.

Ejercicio 3.21

Dada una cadena, sustituir sus vocales por asteriscos. Por ejemplo, si la cadena de entrada es “Celia Viñas”, la salida debe ser “C*l** V*ñ*s”

```
int main(void)
{
   char cad[100];
   int i;

   printf(  Introduzca una cadena:   );
   gets(cad);

   // Recorremos la cadena buscando las vocales
   for (i = 0; i < strlen(cad); i++)
   {
      if ((cad[i] == 'a') || (cad[i] == 'e') || (cad[i] == 'i') || (cad[i] == 'o') ||
          (cad[i] == 'u') || (cad[i] == 'A') || (cad[i] == 'E') || (cad[i] == 'I') ||
          (cad[i] == 'O') || (cad[i] == 'U'))
      {
               // Hemos encontrado una vocal en la posición i, así que la sustituimos por *
              cad[i] = '*';
      }
   } // Fin del for

   // Por último, mostramos el resultado
   puts(cad);
   return 0;
}
```

Ejercicio 3.22

Pedir una cadena por teclado y mostrarla en la pantalla al revés. Por ejemplo, si la cadena de entrada es “En un lugar de la Mancha”, la salida debe ser “ahcnaM al ed ragul nu nE”

```
int main(void)
{
   char cad[100];
   int i;

   printf("Introduzca una cadena:");
   gets(cad);

   // Recorremos la cadena desde el final hacia el principio
   for (i = strlen(cad); i >= 0; i--)
   {
       // Imprimimos cada carácter
       printf("%c", cad[i]);
   }
```

```
   return 0;
}
```

Ejercicio 3.23

Comprobar si, en una cadena leída por teclado, los caracteres están dispuestos en orden alfabético creciente.

```
int main(void)
{
    int orden, i;
    char cad[100];

    printf("Introduzca una cadena: ");
    gets(cad);

    orden = 1;               // Centinela. Valdrá 1 si las letras están en orden
    i = 1;                   // Contador. Empezaremos en el carácter 1, no en el 0

    while ((orden == 1) && (i < strlen(cad)))
    {
        if (cad[i] < cad[i-1])   // Comparamos cada letra con la anterior
        {
                orden = 0;       // Si estas letras no están en orden, ponemos orden = 0
        }
        i++;
    }

    // Si el bucle termina y orden sigue siendo 1, es que no se encontraron letras desordenadas
    if (orden == 1)
        printf("Las letras de la cadena están en orden alfabético\n");
    else
        printf("Las letras de la cadena NO están en orden alfabético\n");

    return 0;
}
```

Ejercicio 3.24

Leer una cadena por teclado y contar el número de vocales de cada tipo que existen. Por ejemplo, si la cadena de entrada es “Hola, mundo”, la salida debe ser: A = 1, E = 0, I = 0, O = 2, U = 1.

```
int main()
{
    char cad[50];
    int contA, contE, contI, contO, contU;  // Necesitamos un contador para cada tipo de vocal
    int i;

    printf(  Introduzca una cadena:   );
    gets(cad);

    // Inicializamos todos los contadores a cero
    contA = 0;
    contE = 0;
    contI = 0;
    contO = 0;
    contU = 0;

    // Recorremos la cadena buscando las distintas vocales
    for (i = 0; i < strlen(cad); i++)
```

```
    {
       // Miramos el carácter actual para ver si es una vocal
       if ((cad[i] == 'a') || (cad[i] == 'A'))     // Es una A
              contA = contA + 1;

       if ((cad[i] == 'e') || (cad[i] == 'E'))     // Es una E
              contE = contE + 1;

       if ((cad[i] == 'i') || (cad[i] == 'I'))     // Es una I
              contI = contI + 1;

       if ((cad[i] == 'o') || (cad[i] == 'O'))     // Es una O
              contO = contO + 1;

       if ((cad[i] == 'u') || (cad[i] == 'U'))     // Es una U
              contU = contU + 1;
    }

    // Mostramos el contenido final de los contadores
    printf(  A = %i, E = %i, I = %i, O = %i, U = %i\n  , contA, contE, contI, contO, contU);

    return 0;
}
```

Ejercicio 3.25

Comprobar si una cadena introducida por teclado es un *palíndromo*.

```
int main()
{
    char c[50], cad[50];
    int i, j, iz, de, diferencias;

    puts("Teclee una cadena: ");
    gets(c);

    /* Eliminamos los espacios de la cadena c y la copiamos en cad */
    j = 0;
    for (i = 0; i < strlen(c); i++)
    {
        if (c[i] != ' ') {
            cad[j] = c[i];
            j++;
        }
    }
    cad[j] = '\0';    /* Añadimos el nulo al final de la nueva cadena */

    /* Hacemos un bucle con dos contadores: uno empezará por un extremo de la cadena
       y el otro por el otro extremo, hasta que se crucen. Si todos los caracteres de ambos
       extremos coinciden, entonces es un palíndromo */
    iz = 0;  de = strlen(cad)-1;
    diferencias = 0;
    while ((iz < de) && (diferencias == 0))
    {
        if (cad[iz] != cad[de])      /* Hemos encontrado una diferencia: NO ES UN PALINDROMO */
           diferencias++;

        iz++;
        de--;
    }
```

```
    if (diferencias == 0)
       puts("La palabra es un palíndromo");
    else
       puts("La palabra NO es un palíndromo");

    return 0;
}
```

Ejercicio 3.26 – Solución 1: contando los caracteres coincidentes

Leer dos cadenas, cad1 y cad2, y buscar una ocurrencia de cad2 dentro de cad1, sustituyéndola por asteriscos (" * ")

```
int main()
{
    char cad1[50], cad2[50];
    int i,j,coincidencias;

    puts("Teclee el primer texto: ");
    gets(cad1);
    puts("Teclee el segundo texto: ");
    gets(cad2);
    // Comprobamos que la segunda cadena sea más corta que la primera
    if (strlen(cad2) > strlen(cad1)) {
       puts("El segundo texto debe ser más corto que el primero");
       exit(-1);
    }

    for (i=0; i<strlen(cad1); i++)          // Recorremos todas las letras de cad1
    {
        if (cad1[i] == cad2[0])          // Hemos encontrado una posible coincidencia
        {
            coincidencias = 0;
            // contamos cuantas coincidencias hay entre cad1 y cad2 a partir de la posición
            // i de cad1 y de la posición 0 de cad2
            for (j=0; j<strlen(cad2); j++)
            {
                if (cad1[i+j] == cad2[j])
                    coincidencias++;
            }
            // Si el número de coincidencias es igual a la longitud de cad2, era una ocurrencia
            if (coincidencias == strlen(cad2))
            {                                // Sustituimos en cad1 por asteriscos
                for (j=0; j<strlen(cad2); j++)
                    cad1[i+j] = '*';
            }
        }
    }

    puts("El resultado es:");
    puts(cad1);

    return 0;
}
```

Ejercicio 3.26 – Solución 2: usando una cadena auxiliar

Leer dos cadenas, cad1 y cad2, y buscar una ocurrencia de cad2 dentro de cad1, sustituyéndola por asteriscos (" * ")

```
int main()
{
    char cad1[50], cad2[50], cad3[50];
    int i,j;

    puts("Teclee el primer texto: ");
    gets(cad1);
    puts("Teclee el segundo texto: ");
    gets(cad2);
    // Comprobamos que la segunda cadena sea más corta que la primera
    if (strlen(cad2) > strlen(cad1)) {
       puts("El segundo texto debe ser más corto que el primero");
       exit(-1);
    }

    for (i=0; i<strlen(cad1); i++)          // Recorremos todas las letras de cad1
    {
        if (cad1[i] == cad2[0])            // Hemos encontrado una posible coincidencia
        {
            // construimos cad3 copiando desde cad1 tantas letras como tenga cad2
            for (j=0; j<strlen(cad2); j++)
            {
                cad3[j] = cad1[i+j];
            }
            cad3[j] = '\0';                 // Añadimos el nulo al final de cad3
            // Si cad3 y cad2 son iguales, se ha encontrado una ocurrencia
            if (strcmp(cad2, cad3) == 0)
            {                               // Sustituimos en cad1 por asteriscos
                for (j=0; j<strlen(cad2); j++)
                      cad1[i+j] = '*';
            }
        }
    }

    puts("El resultado es:");
    puts(cad1);

    return 0;
}
```

Ejercicio 3.26 – Solución 3: usando la función de librería strstr()

Leer dos cadenas, cad1 y cad2, y buscar una ocurrencia de cad2 dentro de cad1, sustituyéndola por asteriscos (" * ").

Existe una función estándar de C que sirve para localizar una cadena dentro de otra. Esa función es strstr(). Una vez localizada la subcadena, es fácil sustituirla por asteriscos. Observa cómo se usa strstr() en el siguiente código.

Como puede verse, las funciones estándar de cadenas no hacen nada imprescindible, pues todas esas cosas las podemos programar si es necesario (ver solución anterior). Pero es conveniente conocer qué ayudas nos ofrece el lenguaje para poder aprovecharnos de ellas.

```
int main()
{
    char cad1[50], cad2[50], *cad3;
    int i,j;

    puts("EJERCICIO 5-19 (b): Buscar una cadena dentro de otra");
    puts("Teclee el primer texto: ");
    gets(cad1);
    puts("Teclee el segundo texto: ");
    gets(cad2);

    cad3 = strstr(cad1, cad2);   // Busca cad2 dentro de cad1
    if (cad3 != NULL)            // Se ha encontrado una ocurrencia. cad3 apunta a ella
      for (i=0; i<strlen(cad2); i++)
      {
        cad3[i] = '*';           // Sustituimos por * tantas veces como letras tenga cad2
      }
    puts("El resultado es:");
    puts(cad1);

    return 0;
}
```

Ejercicio 3.27

.Escribe un programa que pida al usuario que introduzca una palabra por teclado de un máximo de 8 caracteres, y luego la oculte en una sopa de letras de 8x8 caracteres generada al azar. La palabra debe aparecer en la sopa escrita de izquierda a derecha, a una altura seleccionada aleatoriamente. La sopa de letras generada debe almacenarse en una matriz de caracteres y mostrarse por la pantalla.

```
int main()
{
        char palabra[100];
        char sopa[8][8];
        int f, c, i, fila, columna;
        srand(time(NULL));

        /* Generamos la sopa de letras aleatoria */
        for (f = 0; f < 8; f++)
                for (c = 0; c < 8; c++)
                        /* El código ASCII de la A es 65 y el de la Z es 91, así que
                           sacamos un aleatorio de 65 a 91 */
                        sopa[c][f] = rand()%26 + 65;

        /* Pedimos al usuario que teclee su palabra */
        printf("Introduzca una palabra (máximo 8 letras)\n");
        gets(palabra);

        /* Comprobamos la longitud de la palabra. Si es mayor que 8, la truncamos */
        if (strlen(palabra) > 8)
                palabra[8] = '\0';

        /* Seleccionamos al azar la fila y la columna en la que ocultar la palabra */
        fila = rand()%8;
        columna = rand()%(8 - strlen(palabra));

        /* Insertamos la palabra dentro de la sopa */
```

```
        i = 0;
        for (c = columna; c < columna + strlen(palabra); c++)
        {
                sopa[c][fila] = palabra[i];
                i++;
        }

        /* Por último, mostramos la sopa de letras */
        for (f = 0; f < 8; f++)
        {
                for (c = 0; c < 8; c++)
                {
                        printf("%c ", sopa[c][f]);
                }
                printf("\n");
        }

        return 0;
}
```

Ejercicio 3.28

Escribe un programa que lea un texto por teclado y separe las palabras que haya en él, escribiéndolas en líneas consecutivas. Además, el programa debe contar las palabras.

```
int main()
{
        char cad[100];
        int i, cont;

        /* Pedimos al usuario que teclee su cadena */
        printf("Introduzca una cadena de caracteres: ");
        gets(cad);

        /* Recorremos la cadena buscando los espacios */
        cont = 0;
        for (i = 0; i < strlen(cad); i++)
        {
                if (cad[i] != ' ')    /* No es un espacio: mostramos el carácter en la pantalla */
                        printf("%c", cad[i]);
                else    /* ES UN ESPACIO: hacemos un salto de línea e incrementamos el contador */
                {
                        printf("\n");
                        cont++;
                }
        }

        /* Como la última palabra no está seguida de un espacio, la contamos ahora */
        printf("\n");
        cont++;

        printf("Total: %i palabras\n", cont);

        return 0;
}
```

Ejercicio 3.29

Leer un número decimal en forma de cadena y convertirlo a su equivalente en notación binaria

***Nota importante**: para convertir un número (por ejemplo, el 1) en un carácter ASCII (es decir, en un '1'), basta con sumar 48 al número, ya que 48 es el código ASCII del carácter '0', 49 el del carácter '1', 50 el del '2', y así sucesivamente.*

```
int main(void)
{
    int resto, i, j;
    char n_bin[100], result[100];
    int n_dec;

    printf("Introduzca numero decimal: ");
    scanf("%i", &n_dec);

    i = 0;
    while (n_dec > 0)            // Bucle principal
    {
        resto = n_dec % 2;       // Vamos obteniendo los restos de las divisiones
        n_dec = n_dec / 2;
        result[i] = resto + 48;  //... los convertimos a carácter y los almacenamos en "result"
        i++;
    }
    result[i] = '\0';            // Añadimos el nulo al final de la cadena

    // "result" contendrá el número binario AL REVÉS. Tenemos que darle la vuelta
    j = 0;
    for (i=strlen(result)-1; i>=0; i--)
    {
       n_bin[j] = result[i];    // En "n_bin" obtendremos el binario en el orden correcto
       j++;
    }
    n_bin[j] = '\0';            // Añadimos el nulo al final de la cadena

    printf("El número binario equivalente es: %s\n\n", n_bin);

    return 0;
}
```