ICOM-2016 Práctica 7

1. Implemente la función **`Replace`** que reemplace en el string **`s`** todas las apariciones del carácter **`viejo`** por el carácter **`nuevo`**. Prototipo:

```
void Replace(char s[], char nuevo, char viejo);
```

2. Implementar una función **`StrLen`** que reciba un arreglo de caracteres y devuelva la longitud del string. Prototipo:

```
int StrLen(char Array[]);
```

3. Implementar una función **`StrCmp`** que reciba dos arreglos de caracteres y devuelva un entero <0, =0 ó >0 dependiendo que el primer string sea menor, igual o mayor que el segundo string (alfabéticamente). Prototipo:

```
int StrCmp(char Array1[], char Array2[]);
```

4. Implementar una función **`StrCat`** que reciba dos arreglos de caracteres y concatene el segundo string al primero. (El primero debería tener dimensión suficiente para ello. Prototipo:

```
void StrCat(char Array1[], char Array2[]);
```

5. Implementar la función **`StrStr`** que reciba dos arreglos de caracteres y busque sobre el primero de ellos por la ocurrencia del segundo. Prototipo:

```
int StrStr(char s1[], char s2[]);
```

Como retorno, **`StrStr`** deberá retornar el índice del elemento sobre `s1` donde comienza la ocurrencia de **`s2`**. Si **`s2`** no aparece en **`s1`**, la función deberá retornar -1. Ejemplos:

`StrStr("JUAN ESTA CASADO CON MARIA", "ASADO")` retornará 11.

`StrStr("ABCDE", "BCE")` retornará -1.

6. Un palíndromo es aquella palabra o frase que puede ser leída tanto de izquierda a derecha como de derecha a izquierda**.** Ejemplo: *LA RUTA NOS APORTO OTRO PASO NATURAL.* Implementar una función que retorne verdadero (1) o falso (0) indicando si la cadena de caracteres recibido define un palíndromo. Prototipo:

```
int EsPalindromo(char frase[]);
```

ejemplos de prueba: “Yo hago yoga hoy”, “Yo dono rosas, oro no doy”, “Atale, demoniaco Cain, o me delata”, “A la gorda drogala”.

7. Escribir una función que reciba como parámetro una cadena de caracteres que puede comenzar con espacios en blanco, y los elimine desplazando los caracteres útiles hacia la izquierda. (operación "*left trim*"). Prototipo:

```
void LeftTrim (char phrase[]);
```

8. Hacer un programa que lea el archivo *mapa.dat* e imprima los contenidos de la tercera y cuarta columna como matrices en pantalla.

9. Hacer un programa que lea un archivo con 2 columnas de valores en punto flotante representando puntos (x,y) y diga para que x el valor de y es máximo. (*peaks.dat*).

10. Hacer un programa que lea un archivo que contenga palabras (las palabras tienen una longitud máxima de 32 caracteres (**`MAX_WORD_SZ`**)) y arme estadísticas acerca de la cantidad de veces que se repite cada una. Como resultado, el programa deberá imprimir cada palabra con las veces que ésta se repite. Se deberá soportar diccionarios de hasta **`MAX_WORDS`** palabras distintas. Utilizar estructuras como las siguientes:

```
typedef struct {
    char word[MAX_WORD_SZ];         // palabra en si
    unsigned int repCnt;            // cantidad de repeticiones
} StatRecord_t;

typedef struct {
    StatRecord_t records[MAX_WORDS];
    unsigned usedCnt;               // cantidad de entradas en uso
} StatWords_t;
```

Hints:

- Antes de comenzar a juntar estadística se deberá inicializar el atributo **`usedCnt`** a 0.
- Para cada palabra leída se deberá primero chequear su existencia en el diccionario, si existe se deberá incrementar su cantidad de repeticiones (**`repCnt`**), si no existe se deberá agregar la nueva palabra (si es que todavía hay entradas disponibles en el diccionario), inicializar su **`repCnt`** en 1 e incrementar **`usedCnt`** en 1.
- Probar el programa ejecutándolo redirigiendo el dispositivo standard de entrada hacia un archivo.