Guía de Ejercicios #6 - Strings

```
Advertencia

La resolución conjunta o grupal de los ejercicios aquí presentes no está
permitida, excepto en la medida en que puedas pedir ayuda a tus compañeros
de clase y a otras personas, y siempre que esa ayuda no se reduzca a que
otro haga el trabajo por vos.

El código fuente entregado por un estudiante debe ser escrito en su
totalidad por dicha persona.
```

Ejercicio 6.1

Realizar una función que reciba una cadena y devuelva el largo de la misma según el siguiente prototipo: **int obtener_largo(char *)**.

Nota: No se admite el uso de ninguna función cuyo prototipo se encuentre en string.h

Ejercicio 6.2

Escribir una función que compare dos strings para probar si los mismos son iguales o distintos. La función debera delvolver 0 si los strings son iguales o 1 si no lo son.

Nota: Utilizar el siguiente prototipo: **int comparar_cadena(char *, char *)**
No se admite el uso de ninguna función cuyo prototipo se encuentre en string.h

Ejercicio 6.3

Realizar una función que reciba una cadena y convierta las letras de un string que esten en minúscula a mayúscula.

Nota: Utilizar el siguiente prototipo: **int a_mayuscula(char *)**
No se admite el uso de ninguna función cuyo prototipo se encuentre en string.h

Ejercicio 6.4

Escribir una función que reciba un puntero a char y convierta las letras de un string que esten en mayúscula a minúscula.

Nota: Utilizar el siguiente prototipo: **int a_minuscula(char *)**
No se admite el uso de ninguna función cuyo prototipo se encuentre en string.h

Ejercicio 6.5

Escribir una función que reciba dos cadenas y que concatene dos strings. En caso de haber podido unir los strings la función devolverá 1, en caso de error debe devolver 0. El inicio del nuevo string resultante deberá apuntar al inicio de posición de memoria del primero, con lo cual el primer string quedará pisado por el nuevo.

Nota: Utilizar el siguiente prototipo: **int unir_cadenas(char *, char *)**
No se admite el uso de ninguna función cuyo prototipo se encuentre en string.h

Ayuda: Dado que aún no hemos visto el pedido de memoria dinámica, una buena opción para trabajar este ejercicio es tener en cuenta el tamaño asignado a cada string y, posteriormente, sus largos para poder agregar el segundo al primero o, informar que no es posible hacerlo.

Ejercicio 6.6

Escribir una función que reciba una frase y cuente el número de vocales y consonantes que contiene. Además, la función deberá informar la cantidad de caracteres que contiene la frase (sin contar los espacios en blanco). Utilizar el siguiente prototipo: **int contador(char *)**. De funcionar devuelve 0 si no, devuelve 1.

Nota: Asumir que cada palabra está separada por un sólo espacio en blanco. Se debe contemplar mayúscula y minúscula, lo cual significa que por ejemplo: 'Hola' es distinto a 'hola'.

Ejercicio 6.7

Realizar una función que cambie la extensión de "un archivo". Su primer argumento es el nombre del archivo y su segundo argumento es la nueva extensión. De realizar alguna modificación devuelve 0, si no devuelve 1.

Nota: Utilizar el siguiente prototipo: **int reemplazar(char *, char *)**

Ayuda: Tener en cuenta que los nombres de los archivos y sus extensiones estan delimitadas por un punto. Por ejemplo, archivo.txt ---reemplazar()---> archivo.ini

Ejercicio 6.8

Escribir una función que reciba una frase y dos caracteres. Cada vez que aparezca el primer caracter lo reemplazara por el segundo en el texto. Si la función realiza algún reemplazo devolverá 0, sino 1.

Nota: Utilizar el siguiente prototipo: **int reemplazar_str(char *, char , char)** La frase a ingresar deberá tener un máximo de 100 caracteres.

Ayuda: Para el ingreso de la frase use la funcion fgets().

Ejercicio 6.9

Escribir una función que reciba una frase y un caracter, la misma deberá retornar cuántas veces aparece dicho caracter en la frase. De no contar ningún caracter o de haber un error deberá retornar -1.
Por ejemplo, en la frase "Tengo que ir al oculista, pero nunca veo el momento" el caracter 'o' aparece seis veces.

Nota: Utilizar el siguiente prototipo: **int contador_de_char(char *, char)**. Las frases admitidas no deben superar los 100 caracteres y se debe distinguir entre mayúscula y minúscula.

Ejercicio 6.10

Escriba una función que reciba una palabra y devuelva un 1 si la misma es palíndroma, en caso contrario devolverá un 0.

Nota: Utilizar el siguiente prototipo: **int es_palindroma(char *)**.

Ayuda: Una palabra palíndroma es aquella que es igual si se lee de izquierda a derecha que de derecha a izquierda. Por ejemplo: Neuquén, oso, ojo, etc.

Ejercicio 6.11

Realice una función que cuente la cantidad de veces se encuentra una palabra en una frase. Tener en cuenta que no se trata de una comparación literal, es decir que no importa si una letra esta en minúscula o mayúscula para este caso.

Nota: Utilizar el siguiente prototipo: **int contador_de_palabras(char *, char *)**.

Ayuda: Para el ingreso tanto de la frase como el de la palabra use la funcion fgets().

Ejercicio 6.12

Realice una función que reciba una palabra por referencia e invierta el orden de sus letras. La palabra recibida no deberá superar los diez caracteres. Por ejemplo, batman --> namtab. En caso de invertirla correctamente devuelve un 1, en caso contrario devuelve 0.

Nota: Utilizar el siguiente prototipo: **int invertidor_de_palabras(char *)**.

Ayuda: Para el ingreso de la palabra use la funcion fgets().

Ejercicio 6.13

Realizar un programa que solicite el ingreso, como cadenas, del nombre y edad de dos personas. Luego del ingreso se deberá mostrar en pantalla el nombre y edad de la persona que sea mayor. Ejemplo,

ingreso 1	ingreso 2	invoco a mi función	salida
julian 20	amadea 32	---->	amadea es mayor, con 32 años

Nota: Usar el siguiente prototipo: **int comparador_personas(char *, char *)** . Corroborar que las edades ingresadas sean válidas, además, los nombres deben tener 20 caracteres o menos.

Ejercicio 6.14

Un sistema de codificación muy famoso es el código Morse, desarrollado por Samuel Morse en 1832, para su uso en el sistema telegráfico. El mismo asigna una serie de puntos y rayas a cada letra del alfabeto, a cada dígito y a unos cuantos caracteres especiales.
Utilizaremos un espacio en blanco para separar cada letra codificada en Morse y tres espacios en blanco entre cada palabra codificada en Morse.

- Realice una función que lea una frase en español y cifre dicha frase en código Morse. La misma retornará 1 de haber sido posible la codificación o 0 en caso de no haber sido posible la misma.

Nota: Utilizar el siguiente prototipo: **int a_morse(char *)**.

Tabla de Código Morse

A •–	B –•••	C –•–•	D –••	E •	F ••–•	G ––•
H ••••	I ••	J •–––	K –•–	L •–••	M ––	N –•
Ñ ––•––	O –––	P •––•	Q ––•–	R •–•	S •••	T –
U ••–	V •••–	W •––	X –••–	Y –•––	Z ––••	
0 –––––	1 •––––	2 ••–––	3 •••––	4 ••••–	5 •••••	6 –••••
7 ––•••	8 –––••	9 ––––•				
+ •–•–•	- –••••–	: –––•••	. •–•–•–	, ––••––	! –•–•––	? ••––••

Ejercicio 6.15

Lea el ejercicio 6.14 y realice una función que lea una frase en código Morse y obtenga su decodificación. La misma retornará 1 de haber sido posible la decodificación o 0 en caso de no haber sido posible la misma.

Nota: Utilizar el siguiente prototipo: **int a_espaniol(char *)**.

Ejercicio 6.16

Las fechas se imprimen comúnmente en varios formatos distintos, dos de los más comunes son:
"01/09/98" y "01 de Septiembre del 1998".
Escriba una función que reciba el primer formato ("01/09/98") y devuelva la fecha con el segundo formato ("01 de Septiembre del 1998"). De poder realizar la conversión devolverá 0, en caso contrario devolvera 1.

Nota: Utilizar el siguiente prototipo: **int fecha_a_numeros(char *)**

Ejercicio 6.17

Lea el enunciado del ejercicio 6.16 y escriba una función que reciba el segundo formato ("01 de Septiembre del 1998") y devuelva la fecha con el primer formato ("01/09/98"). De poder realizar la conversión devolverá 0, en caso contrario devolvera 1.

Nota: Utilizar el siguiente prototipo: **int fecha_a_letras(char *)**

Ejercicio 6.18

Escriba un programa que ayude al usuario a realizar conversiones sencillas y especificar que tipo de conversiones desea realizar.
Para ello el usuario deberá especificar los nombres de las unidades(metros, millas, pulgadas, etc) que desea usar y, el programa deberá responder a preguntas sencillas como:
¿Cuántos metros hay en una milla?
¿Cuántas millas hay en dos yardas?

Nota: Tenga encuenta que en este caso sólo son válidas las medidas de longitud presentadas a continuación. En cualquier otro caso, el programa deberá informar que no es posible realizar dicha conversión.

medida	**equivalente en metros**	**fórmula para pasar de metro a la nueva medida:**
milla	1609 metros	para obtener un resultado aproximado, multiplica el valor de longitud por 1609
pie	0.3048 metros	para obtener un resultado aproximado, divide el valor de longitud entre 3,281
pulgada	0.0254 metros	divide el valor de longitud entre 39,37

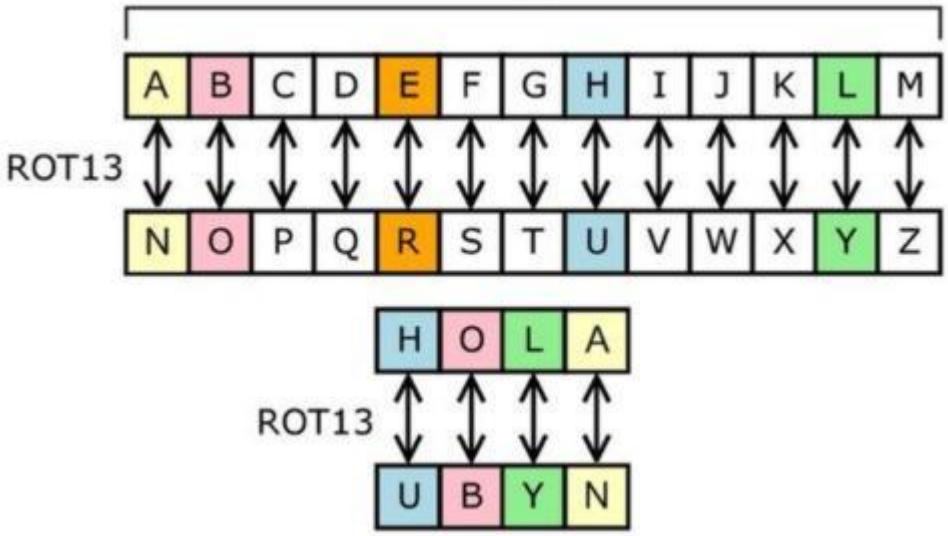
- Realice una función con el prototipo **int es_valida(char*)** que recibirá la pregunta del usuario y validará si la misma tiene medidas válidas de conversión. De ser válida devuelve 1, si no, devuelve 0.
- Luego deberá realizar funciones que conviertan de una unidad a otra, tenga en cuenta trabajar con variables flotantes.

Ejercicio 6.19

En criptografía, el cifrado César, también conocido como cifrado por desplazamiento o ROT13, es una de las técnicas de cifrado más simples y más usadas. Es un tipo de cifrado por sustitución en el que una letra en el texto original es reemplazada por otra letra que se encuentra un número fijo de posiciones más adelante en el alfabeto. Por ejemplo, con un desplazamiento de 3, la A sería sustituida por la D (situada 3 lugares a la derecha de la A), la B sería reemplazada por la E, etc. Este método debe su nombre a Julio César, que lo usaba para comunicarse con sus generales.
Entonces, para codificar un mensaje, simplemente se debe buscar cada letra de la línea del texto original y escribir la letra correspondiente en la línea codificada. Para decodificarlo se debe hacer lo contrario.

- Escribir una función que codifique una frase del español al cifrado César.

Nota: Utilizar el siguiente prototipo: **int a_cesar(char *)**. Devuelve 1 si codifica con éxito o 0 en caso contrario. La función no deberá codificar frases que tengan más de 50 caracteres. Para el ingreso de la frase usar la funcion fgets().



Ejercicio 6.20

Leer el ejercicio 6.19 y realizar una función que decodifique una frase del cifrado César al español.

Nota: Utilizar el siguiente prototipo: **int a_espaniol(char *)**. Devuelve 1 si decodifica con éxito o 0 en caso contrario. La función no deberá decodificar frases que tengan más de 50 caracteres. Para el ingreso de la frase usar la funcion fgets().

Ejercicio 6.21

Escriba un programa que use la generación de números aleatorios para crear oraciones.
Se deberán usar cuatro arrays apuntadores a char llamados: articulo, sustantivo, verbo y preposicion. El programa deberá crear una oración seleccionando una palabra al azar de cada uno de los arrays en el orden siguiente:
1)articulo, 2) sustantivo, 3)verbo, 4)preposicion, 5)articulo y 6) sustantivo.
Conforme se seleccione cada palabra, deberá ser concatenada con las palabras anteriores en un string lo suficientemente extenso para contener toda la oración.
Las palabras deben estar separadas por un espacio y el programa deberá generar diez oraciones de este tipo.
Los strings que deben utilizar para el armado de oraciones son:

Artículo	Sustantivo	Verbo	Preposición
el	niño	manejo	hacia
un	perro	salto	desde
uno	gato	corrio	sobre
algun	pueblo	camino	debajo
ningun	auto	esquivo	entre

Nota: Utilizar el siguiente prototipo para la función generadora de oraciones: **int gen_oracion(char *)**. Devuelve 1 si logra hacer la oración ó 0 si no lo logra y, recibe una cadena donde se irán concatenando las palabras.

Ejercicio 6.22

Realizar un programa que reciba tres números enteros y luego proceda a realizar un sorteo. Se deberá sortear veinte veces y, cada vez que aparezca algún número elegido por el usuario, deberá irse armando la palabra **BINGO**.
Se deberá informar el número que sale sorteado y, en caso de que salga algún número elegido por el usuario, se debe mostrar en pantalla las letras del BINGO que se completaron hasta el momento.
De completarse la palabra el usuario recibirá un mensaje de felicitación por haber conseguido el BINGO.

- Realizar una función **int sorteo(int , int, int)** que reciba los números del usuario y retorne 1 de poder hacer el sorteo, 0 si hubo algún error ó 2 de haberse conseguido el BINGO.
- Realizar otra función **void felicitacion()** que imprima por pantalla la felicitación de haber conseguido el BINGO.

Nota: Los números a trabajar deberán encontrarse entre el 0 y el 10 además, para la realización del sorteo puede usar la función rand().

Ejercicio 6.23

Escribir un programa que le pida a tres alumnos: el apellido, número de legajo (entero de 8 cifras) y la nota de un examen. A continuación se deberá mostrar la información en 3 cadenas de caracteres. El orden deberá ser: legajo, apellido y nota de examen. Las lineas deberán mostrarse de menor a mayor según el número de legajo. Por ejemplo,

legajo	apellido	nota
1634804	Levi	6
1736500	Herrera	9
1757843	Potter	7

Nota: Usar el siguiente prototipo: **int validador_personas(char *, char *,char *)** para corroborar que las notas ingresadas sean váidas (del 1 al 10), que los legajos tengan 7 cifras enteras y además, que los nombres tengan 20 caracteres o menos. De ser válido la función retorna 1, de no serlo retorna 0.
Si el ingreso es válido use una función **void mostrar_personas(char *,char *, char *)** para comparar los legajos y luego imprimir por pantalla la información en el orden ya mecionado.

Ayuda: Para comparar los legajos puede usar la función **atoi()**. Puede consultar el man para más información sobre dicha función.

Referencias

Algunos ejercicios fueron obtenidos y adaptados de:

- Guía de Trabajos Prácticos 2011 - Informática I - Departamento de Electrónica - UTN FRBA
- Cómo programar en C/C++ -Deitel Prentice-Hall