PROGRAMACIÓN

(GRADOS EN INGENIERO MECÁNICO, ELÉCTRICO, ELECTRÓNICO INDUSTRIAL y QUÍMICO INDUSTRIAL)

Sesión	12 (Estructuras de Datos: Cadenas de Caracteres)	
Temporización	1 hora (no presencial)	
Objetivos formativos	• Resolver ejercicios sencillos de cadenas de caracteres con representación semi-estática. Conocer las operaciones básicas que se realizan sobre las cadenas de caracteres. • Conocer la sintaxis de C para la definición de nuevas tipologías de datos y para la implementación de estructuras de datos estáticas de tipo *cadenas de caracteres*. Conocer y saber utilizar las principales funciones de las bibliotecas **string.h** y **stdio.h** para la manipulación de cadenas de caracteres representadas en términos de vectores de caracteres que finalizan su longitud dinámica con el carácter centinela **'\0'**. • Implementar programas modulares en lenguaje de programación C. Identificar y corregir errores sintácticos que surgen durante la codificación. • Probar con datos operacionales la correctitud de los módulos y programas desarrollados e identificar y corregir los errores lógicos que surjan.	
Competencias a desarrollar	• RD1: Poseer y comprender conocimientos • RD2: Aplicación de conocimientos • UAL1: Conocimientos básicos de la profesión • UAL3: Capacidad para resolver problemas • UAL6: Trabajo en equipo • FB3: Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en la ingeniería.	X X X X X
Materiales	Sesiones de teoría (grupo docente) 11.1 a 11.3 + Bibliografía Tema 4 + Internet IDEs : Dev-C++/Code::Blocks (freeware)	
Tarea	Desarrollar los tres sub-programas propuestos en esta ficha de trabajo y presentar un informe según modelo que se adjunta.	
Fecha de entrega	Siguiente sesión del Grupo de Trabajo.	
Criterios de éxito	• Terminar en el tiempo previsto la tarea. • Demostrar, en una prueba escrita u oral, mediante las respuestas a las preguntas del profesor que ha alcanzado los objetivos formativos.	
Plan de trabajo	**Actividad**	**Temporización**
	Estudio de la sintaxis de C para representar el tipo abstracto de datos cadena de caracteres, así como de las principales funciones de la biblioteca **string.h** y **stdio.h** para manipular cadenas de caracteres.	10 mn
	Diseño de los sub-algoritmos correspondientes a cada uno de los ejercicios propuestos. Nota: puede simultanear esta actividad con las dos siguientes (para cada ejercicio).	20 mn

	Implementación en lenguaje C de los sub-programas correspondientes a los algoritmos diseñados. Nota: en el anexo dispone de un programa de prueba para las funciones desarrolladas, donde tan solo tiene que insertar el código fuente de las mismas.	10 mn
	Pruebas: los programas desarrollados serán validados utilizando como mínimo los datos de prueba suministrados. Nota: en caso de detectar errores en esta fase de pruebas, estos deberán ser corregidos modificando el código fuente y/o el algoritmo correspondiente.	10 mn
	Elaboración de la documentación a presentar según modelo adjunto, así como de la respuesta a las cuestiones planteadas en el mismo.	10 mn

Sintaxis de C: cadenas de caracteres

- Constantes de cadena de caracteres: una constante de cadena en C es una secuencia de 0 o más caracteres (normalmente del código ASCII extendido), escrita en una línea de programa y encerrada entre comillas dobles. Ejemplos:

```
"Ejemplo de cadena"
""                      /* Cadena nula                                  */
"\"¡Hola!\", soy yo"    /* Las comillas se representan dentro de la     */
                        /* cadena como \"                               */
"¡Cuidado!\007"         /* Se pueden introducir caracteres especiales   */
"Linea 1\nLinea 2\n"    /* (de control de salida ó de representación    */
                        /* confundible) mediante su código octal,       */
                        /* o su secuencia de escape que es              */
                        /* independiente del sistema de codificación    */
```

- Variables de cadena de caracteres: en C no existe un tipo especial cadena, y éstas se representan normalmente mediante vectores de caracteres, usándose una marca especial o centinela (carácter nulo: '\0') para indicar el final de la cadena.
  - Declaración de una variable cadena: `char nombre_variable[N+1];`
    donde N es el número máximo de caracteres que puede almacenar la cadena (nótese el elemento extra adicional añadido para el centinela).

    Ejemplos:
    ```
    char nombre[16];    /* cadena de hasta 15 caracteres    */
    char direccion[31];/* cadena de hasta 30 caracteres     */
    char texto[2001];   /* cadena de hasta 2000 caracteres  */
    ```

  - Declaración de un tipo cadena: `typedef char tipo_cadena[N+1];`
    Declaración de variables: `tipo_cadena nombre_variable;`

    Ejemplos:
    ```
    /* tipos definidos por el usuario */
    typedef char cadena15[16];
    typedef char cadena30[31];
    typedef char cadena2000[2001];
    /* declaración de variables  */
    cadena15 nombre;
    cadena30 direccion;
    cadena2000 texto;
    ```

- Operaciones con cadenas.

- Acceso a caracteres individuales de la cadena: como las cadenas se definen en C como vectores de caracteres, los componentes de estas se acceden como los elementos de un vector (a través de su índice):

  `nombre_variable_cadena[i]`

  donde i denota la posición del carácter, empezando a contar desde 0.
- Tratamiento de la cadena como una unidad: mediante funciones de biblioteca o funciones definidas por el programador. Nótese que en C no se permite la asignación directa a variables tipo cadena (operador =) ni la comparación directa de cadenas (con los operadores relacionales) → en su lugar hay que utilizar una función de una biblioteca externa (***strcpy***, ***strcmp***,…) o definir una función para tal propósito.

Funciones para E/S de cadenas	**#include <stdio.h>**	
printf(...)	int	Para escribir en pantalla una cadena se usa la secuencia de salida %-n1.n2s n1 anchura mínima del campo - ajustar a la izquierda n2 número de caracteres a imprimir Ejemplo: `printf("%s",nombre);`
puts(s)	int	Escribe en pantalla una cadena (hasta \0). Después de escribir salta de línea. Ejemplo: `puts(cadena);`

scanf(...)	int	Para leer del teclado una cadena se usa la secuencia de entrada %s. Nótese que por defecto ***scanf*** lee palabras.
		Ejemplos de utilización de ***scanf*** para leer cadenas de caracteres con espacios en blanco (observe que la variable de cadena no va precedida por "&" por tratarse de un vector):
		`scanf(" %[ ABCDEFGHIJKLMNOPQRSTUVWXYZ]s",nombre);`
		Asigna a la variable de cadena `nombre` los caracteres introducidos por el dispositivo estándar de entrada, y finalizará la lectura cuando aparezca un carácter diferente de los encerrados entre corchetes.
		`scanf(" %[^\n]s", apellidos);`
		Los caracteres de dentro de los corchetes se interpretan como los caracteres de finalización de la asignación por teclado a la variable `apellidos`.

gets(s) Eliminada a partir de C11	char*	Lee una cadena de caracteres del teclado, admite espacios en blanco y finaliza cuando se pulsa ***Intro***. Coloca la marca \0 al final de la cadena. Devuelve un valor (0 ó NULL si hay problemas). Ejemplo: `gets(nombre);`
fgets(s,n,stdin)	char*	Lee una cadena de hasta n-1 caracteres del teclado (entrada estándar), admite espacios en blanco y finaliza cuando se pulsa Intro. Coloca la marca \0 al final de la cadena. Devuelve un valor 0 ó NULL si hay problemas. Ejemplo: `fgets(nombre,20,stdin);`
fflush(stdin)	int	Vacía el buffer de teclado (entrada estándar) eliminando todos los caracteres contenidos en el mismo. Devuelve EOF si ocurre un error, y 0 en caso contrario. Ejemplo: `fflush(stdin);`

Funciones de manejo de cadenas	**#include <string.h>**	
strlen(s)	int	Devuelve la longitud dinámica de una cadena. Ejemplo: `i=strlen(cadena);`
strcat(s1,s2)	char*	Concatena dos cadenas (añade a la primera la segunda). Ejemplo: `strcat(cadena1,cadena2);`
strcmp(s1,s2)	int	Compara dos cadenas, y devuelve 0 si son iguales y distinto de cero si son diferentes (un valor negativo si s1<s2, y un valor positivo si s1>s2). Ejemplo: `i=strcmp(cadena1,cadena2);`
strcmpi(s1,s2)	int	Compara dos cadenas considerando las mayúsculas igual a las minúsculas, y devuelve 0 si son iguales y distinto de cero si son diferentes (un valor negativo si s1<s2, y un valor positivo si s1>s2). Ejemplo: `i=strcmpi(cadena1,cadena2);`
strcpy(s1,s2)	char*	Copia una cadena: pasa el contenido del segundo argumento al primero. Ejemplo: `strcpy(cadena_d,cadena_f);`
strset(s,c)	char*	Asigna a todos los caracteres de la cadena el carácter c (excluyendo el carácter de terminación \0). Ejemplo: `strset(cadena,caracter);`
strlwr(s)	char *	Convierte las letras mayúsculas de una cadena a minúsculas. Ejemplo: `strlwr(cadena);`
strupr(s)	char *	Convierte las letras minúsculas de una cadena a mayúsculas. Ejemplo: `strupr(cadena);`
Funciones de conversión cadenas/números		
		#include <stdlib.h>
atoi(s)	int	Convierte una cadena en un entero, reconociendo los caracteres en el siguiente orden: • Una cadena opcional de tabuladores y espacios en blanco • Un signo opcional • Una cadena de dígitos El primer carácter no reconocido finaliza la conversión. Devuelve el valor convertido a entero de la cadena de entrada y 0 en el caso de que esta no se pueda convertir. Ejemplo: `i=atoi(cadena);`
atol(s)	long int	Convierte una cadena en un entero largo. Ejemplo: `n=atol(cadena);`
atof(s)	double	Convierte una cadena en un número real. Reconoce además los caracteres punto decimal y la letra 'E' con y sin signo. Ejemplo: `num_real=atof(cadena);`
strtod(s,&p) char *p	double	Convierte una cadena a real en doble precisión. También reconoce +INF y –INF, y +NAN y –NAN. Ejemplo: `num_real_doble = strtod(cadena, &p);`
		#include <stdio.h>
sprintf(s,...)	int	Imprime salida formateada a una cadena (en lugar del monitor). Sintaxis: `sprintf(cadena,"cadena_control",argumentos);`
sscanf(s,...)	int	Lectura de entrada formateada de una cadena (en lugar del teclado). Sintaxis: `sscanf(cadena,"cadena_control",argumentos);`

Conversión carácter / entero

En C los enteros y los caracteres se mezclan libremente. Ejemplo:

```
int n;
char c;
n=c;  /* convierte el carácter en entero (código de representación del carácter) */
c=n; /* convierte el entero en carácter (carácter asociado al código numérico)  */
```

Funciones que manejan caracteres #include <ctype.h>		
toascii(c)	int	Convierte el valor del argumento a ASCII.
toupper(c)	int	Convierte un carácter a mayúsculas. Ejemplo: sal=toupper(ent);
tolower(c)	int	Convierte un carácter a minúsculas.
isalpha(c)	int	Devuelve un valor distinto de 0 (cumple la condición) si el argumento es un carácter alfabético. Ejemplo: i=isalpha(car);
isdigit(c)	int	Devuelve un valor distinto de 0 si el argumento es un dígito.
islower(c)	int	Devuelve un valor distinto de 0 si el argumento es una letra minúscula.
isspace(c)	int	Devuelve un valor distinto de 0 si el argumento es un espacio en blanco.
isupper(c)	int	Devuelve un valor distinto de 0 si el argumento es una letra mayúscula.
isalnum(c)	int	Devuelve un valor distinto de 0 si el argumento es un carácter alfanumérico.
isascii(c)	int	Devuelve un valor distinto de 0 si el argumento es un carácter ASCII (0-127).
iscntrl(c)	int	Devuelve un valor distinto de 0 si el argumento es un carácter de control.
ispunct(c)	int	Devuelve un valor distinto de 0 si el argumento es un signo de puntuación.
isgraph(c)	int	Devuelve un valor distinto de 0 si el argumento es un carácter ASCII gráfico (hex 0x21-0x7e; octal 041-176).
isodigit(c)	int	Devuelve un valor distinto de 0 si el argumento es un dígito octal.
isxdigit(c)	int	Devuelve un valor distinto de 0 si el argumento es un dígito hexadecimal.

Codificación en C del ejemplo de las sesiones de Grupo Docente

Ejemplo

Construir un programa que lea por teclado dos cadenas de hasta 50 caracteres y que escriba en pantalla una nueva cadena obtenida mediante la intercalación de los caracteres de las dos cadenas iniciales, esto es, mediante la inserción alternativa de 1 carácter de cada cadena hasta donde sea posible. Ejemplo:

cadena1 → "12345"
cadena2 → "abcdefgh"
intercalación de las cadenas → "1a2b3c4d5efgh"

Codificación

```
/* Programa que combina dos cadenas en           */
/* una nueva cadena, insertando alternativamente */
/* los caracteres de ambas cadenas               */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <ctype.h>
#include <math.h>
#include <string.h>
#define N 50   /* longitud maxima de las cadenas de entrada */

/* Nuevos tipos de datos */
typedef char cadena50[N+1];
typedef char cadena100[2*N+1];

/* Prototipos de funciones */
void intercalar_cadenas(cadena50 cad1,cadena50 cad2,cadena100 cad);
void leer_cadena(char *c, int n);

int main(){
    cadena50 cad1,cad2; /* cadenas a intercalar */
    cadena100 cad;      /* cadena resultante    */
    char c;

    do{ system("cls");
        printf("INTERCALACION DE CADENAS DE CARACTERES\n");
        printf("======================================\n\n");
        printf("Introduzca primera cadena: ");
        leer_cadena(cad1,N);
        printf("\nIntroduzca segunda cadena: ");
        leer_cadena(cad2,N);
        intercalar_cadenas(cad1,cad2,cad);
        printf("\nCadena resultante: %s",cad);
        printf("\n\nDesea efectuar una nueva operacion (s/n)? ");
        c=toupper(getch());
    }while (c!='N');
    return 0;
}

void intercalar_cadenas(cadena50 cad1,cadena50 cad2,cadena100 cad){
    int i,j;        /* indices de vectores                  */
    int l1,l2;      /* longitudes de las cadenas cad1 y cad2 */

    i=0;
    j=0;
    l1=strlen(cad1);
    l2=strlen(cad2);
    while((i<l1)&&(i<l2)){
        cad[j]=cad1[i];
        j=j+1;
        cad[j]=cad2[i];
        j=j+1;
        i=i+1;
    }
    while(i<l1){
        cad[j]=cad1[i];
        i++;
```

```
        j++;
    }
    while(i<l2){
        cad[j]=cad2[i];
        i++;
        j++;
    }
    cad[j]='\0';
    return;
}

void leer_cadena(char *cad, int n){
/* lee por teclado y devuelve una cadena de hasta n caracteres */
/* Nota: la cadena debe preveer una posicion adicional para la */
/* marca fin de cadena (\0) --> tamaño minimo n+1 caracteres   */
    int i,fin;
    char c;

    i=0;
    fin=0;
    while((i<n)&&(!fin)){
        c=getch();
        switch(c){
        case '\r': fin=1;
                   printf("\n");
                   break;
        case '\b': if (i>0){
                     --i;
                     printf("\b \b");
                   }
                   break;
        default:   cad[i]=c;
                   ++i;
                   printf("%c",c);
                   break;
        }
    }
    cad[i]='\0';
    if (i==n) printf("\n");
}
```

Ejercicios: desarrollo de programas

Ejercicio 1

Suponiendo que las cadenas de caracteres vienen representadas como vectores de caracteres terminados en un carácter especial (***'\0'***), construir un módulo que invierta una cadena de hasta 50 caracteres, esto es, que acepte como argumento una cadena de caracteres y la devuelva modificada de manera que el primer carácter sea el último, el segundo el penúltimo,..., y el último el primero. Ej: para ***cad="123456"***, devolvería ***cad="654321"***. Suponer que se dispone de la función que calcula la longitud de la cadena (***longitud_cadena***).

Datos de prueba

cadena original	cadena invertida
"1234567890"	"0987654321"
"12"	"21"
"1"	"1"
""	""

Ejercicio 2

Suponiendo que las cadenas de caracteres vienen representadas como vectores de caracteres terminados en un carácter especial (***'\0'***), construir un módulo que acepte como argumento una cadena de hasta 50 caracteres y la devuelva modificada de manera que se eliminen todos los caracteres no alfabéticos de la misma. Ej: para ***cad="ab1c d3;e"***, devolvería ***cad="abcde"***. Suponer que se dispone de la función que calcula la longitud de la cadena (***longitud_cadena***).

Datos de prueba

cadena original	cadena alfabética
"ab1c d3;e"	"abcde"
"1234567"	""
"abcde"	"abcde"
""	""

Ejercicio 3

Suponiendo que las cadenas de caracteres vienen representadas como vectores de caracteres terminados en un carácter especial (***'\0'***), construir un módulo que imprima en pantalla todas las palabras de una cadena de hasta 50 caracteres dada como argumento. Ej: para ***cad="ab 34 ef"***, imprimiría las palabras ***"ab"*** y ***"ef"***. Suponer que se dispone de la función que calcula la longitud de la cadena (***longitud_cadena***). Nota: una palabra solo puede contener caracteres alfabéticos (letras).

Datos de prueba

cadena original	Palabras de la cadena
"ab 34 ef"	"ab","ef"
"ab cd ef de"	"ab","cd","ef","de"
"12 45;o+98 9h"	"o","h"
"12 34 56"	

Anexo

Programa de prueba para los tres ejercicios desarrollados: el programa lee una cadena de hasta 50 caracteres y a través de un sistema de menús permite la selección de una operación correspondiente a cada uno de los tres ejercicios resueltos, presentando a continuación el resultado de la misma en pantalla.

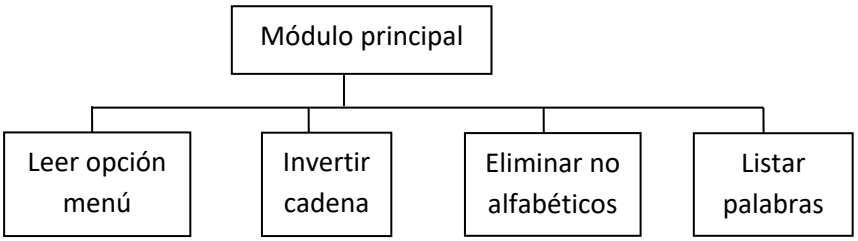
Diseño

Diseño preliminar

Diseño de datos

Tipos cadena50: cadena[51]

Estructura del programa



Interfaces entre módulos

Nombre módulo	Tipo parám.	Nombre parám.	Tipo de datos
Módulo principal			
Leer opción menú	S	c	carácter
Invertir cadena	E/S	cad	cadena50
Eliminar no alfabéticos	E/S	cad	cadena50
Listar palabras	E	cad	cadena50

Diseño detallado

```
Algoritmo EjerciciosCadenasCaracteresVarios
Var     c: carácter
        cad,cad2: cadena50
Inicio  Repetir Escribir("EJERCICIOS DE CADENAS DE CARACTERES")
                Escribir("Introduzca una cadena de caracteres: ")
                Leer(cad)
                Repetir leer_opcion_menu(c)
                        Escribir("Cadena original: ",cad)
                        cad2 ←cad
                        Según_sea(c) Hacer
                         '1':    invertir_cadena(cad2)
                                 Escribir("Cadena invertida: ",cad2)
                        '2':     eliminar_no_alfabeticos(cad2)
                                 Escribir("Cadena modificada: ",cad2)
                         '3':    listar_palabras(cad)
                         '0':     Escribir("FIN DE EJECUCION")
                        Fin_según_sea
                Hasta que(c='0')
                Escribir("Desea efectuar una nueva operacion (s/n)? ")
                Leer(c)
        Hasta que (c='N')ó(c='n')
Fin_algoritmo_principal

Procedimiento leer_opcion_menu(c:carácter(S))
Inicio  Escribir("OPERACIONES CON CADENAS DE CARACTERES")
        Escribir("1.- Invertir cadena")
        Escribir("2.- Eliminar caracteres no alfabeticos de cadena")
        Escribir("3.- Listar palabras de una cadena")
        Escribir("0.- Finalizar")
        Escribir("Introduzca opcion: ")
```

	Leer(c) Fin_procedimiento Procedimiento invertir_cadena(cad:cadena50(E/S)) Inicio { ejercicio invertir cadenas } Fin_procedimiento Procedimiento eliminar_no_alfabeticos(cad:cadena50(E/S)) Inicio { ejercicio eliminar caracteres no alfabéticos } Fin_procedimiento Procedimiento listar_palabras(cad:cadena50(E)) Inicio { ejercicio listar palabras } Fin_procedimiento
Codificación	

```
/* Programa de prueba para los tres    */
/* ejercicios de cadenas de caracteres */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <ctype.h>
#include <math.h>
#include <string.h>
#define N 50    /* longitud maxima de la cadena de entrada */

/* Nuevos tipos de datos */
typedef char cadena50[N+1];

/* Prototipos de funciones */
void leer_cadena(char *c, int n);
void leer_opcion_menu(char *c);
void invertir_cadena(char *cad);
void eliminar_no_alfabeticos(char *cad);
void listar_palabras(char *cad);

/* Definiciones de funciones */
int main(){
    char c;
    cadena50 cad,cad2;

    do{ system("cls");
        printf("EJERCICIOS DE CADENAS DE CARACTERES\n");
        printf("==================================\n\n");
        printf("Introduzca una cadena de caracteres: ");
        leer_cadena(cad,N);
        do{ leer_opcion_menu(&c);
            printf("\n\nCadena original:  %s\n",cad);
            strcpy(cad2,cad);
            switch(c){
            case '1': invertir_cadena(cad2);
                      printf("Cadena invertida: %s\n",cad2);
                      getch();
                      break;
            case '2': eliminar_no_alfabeticos(cad2);
                      printf("Cadena modificada: %s\n",cad2);
                      getch();
                      break;
            case '3': listar_palabras(cad);
                      break;
            case '0': printf("FIN DE EJECUCION");
                      getch();
                      break;
            default: printf("\a");
            }
        }while(c!='0');
        printf("\n\nDesea efectuar una nueva operacion (s/n)? ");
        c=toupper(getch());
    }while (c!='N');
    return 0;
}
```

```
void leer_cadena(char *cad, int n){
/* lee por teclado y devuelve una cadena de hasta n caracteres */
/* Nota: la cadena debe preveer una posicion adicional para la */
/* marca fin de cadena (\0) --> tamaño minimo n+1 caracteres   */
    int i,fin;
    char c;

    i=0;
    fin=0;
    while((i<n)&&(!fin)){
        c=getch();
        switch(c){
        case '\r': fin=1;
                   printf("\n");
                   break;
        case '\b': if (i>0){
                     --i;
                     printf("\b \b");
                   }
                   break;
        default:   cad[i]=c;
                   ++i;
                   printf("%c",c);
                   break;
        }
    }
    cad[i]='\0';
    if (i==n) printf("\n");
}

void leer_opcion_menu(char *c){
       system("cls");
       printf("OPERACIONES CON CADENAS DE CARACTERES\n");
       printf("====================================\n\n");
       printf("\t1.- Invertir cadena\n");
       printf("\t2.- Eliminar caracteres no alfabeticos de cadena\n");
       printf("\t3.- Listar palabras de una cadena\n");
       printf("\t0.- Finalizar\n\n");
       printf("\t\tIntroduzca opcion: ");
       *c=getch();
}

void invertir_cadena(char *cad){
   /* ejercicio invertir cadena */
}

void eliminar_no_alfabeticos(char *cad){
   /* ejercicio eliminar caracteres no alfabeticos */
}

void listar_palabras(char *cad){
   /* ejercicio listar palabras */
}
```

Asignatura	Programación		
Plan de Estudios	Grados en Ingeniero Mecánico, Eléctrico, Electrónico Industrial y Químico Industrial		
Actividad	Trabajo individual	**Sesión**	12
Tiempo empleado			

Apellidos, nombre	**DNI**	**Firma**

1.- Documentación del diseño y de la implementación de los programas desarrollados.

Documentar de forma adecuada los productos de ingeniería obtenidos en las fases principales de desarrollo de los sub-programas de esta práctica: para el diseño utilizar la notación formal de pseudo-código propuesta en las clases de teoría y para la codificación utilizar directamente el listado fuente en lenguaje C.	
Ejercicio	(enunciado del ejercicio)
Diseño	(pseudo-código del módulo)
Codificación	(listado fuente en lenguaje de programación C del sub-programa)
. . . .	

2.- Resultados de aprendizaje (reflexión): marque con una cruz los objetivos que cree haber alcanzado tras realizar esta actividad, y rellene en el campo de observaciones aquellos aspectos que cree que necesita mejorar (si los hubiera):

Objetivos formativos	**Cumplimiento**
• Resolver ejercicios sencillos de cadenas de caracteres con representación semi-estática. Conocer las operaciones básicas que se realizan sobre las cadenas de caracteres. • Conocer la sintaxis de C para la definición de nuevas tipologías de datos y para la implementación de estructuras de datos estáticas de tipo *cadenas de caracteres*. Conocer y saber utilizar las principales funciones de las bibliotecas **string.h** y **stdio.h** para la manipulación de cadenas de caracteres representadas en términos de vectores de caracteres que finalizan su longitud dinámica con el carácter centinela **'\0'**. • Implementar programas modulares en lenguaje de programación C. Identificar y corregir errores sintácticos que surgen durante la codificación. • Probar con datos operacionales la correctitud de los módulos y programas desarrollados e identificar y corregir los errores lógicos que surjan.	
Observaciones	