Práctica de cadenas

1. En C las funciones básicas para el manejo de cadena son:

- ✓ char *strcpy(const char *dest, const char *orig) -- Copia la cadena de caracteres apuntada por orig (incluyendo el carácter terminador '\0') al vector apuntado por dest. Las cadenas no deben solaparse, y la de destino, debe ser suficientemente grande como para alojar la copia.
- ✓ int strcmp(const char *s1, const char *s2) -- Compara las dos cadenas de caracteres s1 y s2. Devuelve un entero menor, igual o mayor que cero si se encuentra que s1 es, respectivamente, menor que, igual a, o mayor que s2.
- ✓ char *strerror(int errnum) -- Devuelve un mensaje de error que corresponde a un número de error.
- ✓ int strlen(const char *s) -- Calcula la longitud de la cadena de caracteres.
- ✓ char *strncat(char *s1, const char *s2, size_t n) -- Agrega n caracteres de s2 a s1.
- ✓ int strncmp(const char *s1, char *s2, size_t n) -- Compara los primeros n caracteres de dos cadenas.
- ✓ char *strncpy(const char *s1, const char *s2, size_t n) -- Copia los primeros n caracteres de s2 a s1.
- ✓ strcasecmp(const char *s1, const char *s2) -- versión que ignora si son mayúsculas o minúsculas de strcmp().
- ✓ strncasecmp(const char *s1, const char *s2, size_t n) -- versión insensible a mayúsculas o minúsculas de strncmp() que compara los primeros n caracteres de s1.

2. Dada la función strstr de la librería stdio.h, cuyo prototipo es el siguiente:

```
char *strstr (const char *cad1, const char *cad2);
```

Esta función devuelve un puntero a la posición donde comienza la cadena apuntada por cad1 dentro de la cadena apuntada por cad2. Sea el siguiente programa:

```
void main(void)
{      Cadena cad1="En un lugar de la Mancha";
       Cadena cad2="la";
       char *p;
       p = strstr(cad1,cad2);
       strcpy (p, "Andalucía Occidental");
       printf("%s", cad1);}
```

- ○ En un lugar de la Mancha
- ○ No se puede saber
- ○ Produce un error en tiempo de ejecución
- ○ Andalucia Occidental
- ○ En un lugar de Andalucia Occidental

3. Dadas las siguientes variables:

```
int i, j, x, y, n, suma;
float *px;
double dx;
char c,*pc;
Cadena palabra;
struct Treg reg;
```

Y dada la invocación: `hacer(*px,c,palabra,reg);`

Entonces la cabecera de esta función puede ser:

- `void hacer (float x, char d, Cadena vocablo, struct Treg reg2)`
- `void hacer (float *x, char d, char vocablo, struct Treg reg2)`
- `void hacer (float x, char c, char palabra, struct Treg reg)`
- `void hacer (float &x, char c, Cadena palabra, struct Treg reg)`
- `Ninguna de las soluciones propuestas es correcta`

4. Teniendo en cuenta una asignación que hemos hecho para la cadena Saludo, hemos escrito varias versiones de una función que calcule la longitud de una cadena, ¿cuáles de ellas funcionan y cuáles no?:

a)
```
int
LongitudCadena(char
cad[])
{
   int l = 0;
   while(cad[l]) l++;
   return l;
}
```
- Sí
- No

b)
```
int
LongitudCadena(char
cad[])
{   int l;
   for(l = 0; cad[l]
!= 0; l++);
   return l;
}
```
- Sí
- No

c)
```
int
LongitudCadena(char
cad[])
{  int l = 0;
   do {
      l++;
   } while(cad[l] !=
0);
   return l;}
```
- Sí
- No

5. Escribe un programa que calcule la longitud de una cadena de caracteres. En este programa, puedes declarar y usar la cadena como lo haces habitualmente, pero se te pide que use notación de punteros en lugar de los [] al momento de obtener el largo de la misma.
6. Si definimos char palabra[]="Hola" entonces:

- `palabra[4] es el carácter a`
- `No da error de compilación, pero no reserva memoria para la variable palabra`
- `palabra[4] es el carácter \0`
- `Tendremos un error de compilación`
- `palabra[4] es el carácter "`

7. Si un programa contiene la sentencia:

```
strcpy(palabra,'a');
```

- El programa no tiene ningún problema
- El programa dará un error de compilación
- El programa dará un error de ejecución
- El programa dará un avISO (warning) de compilación
- El programa no dará error alguno pero no funcionará correctamente

8. Se declara: char s[4]; y se ejecuta la siguiente instrucción: scanf("%s", s); se introduce por teclado: PEPITO GRILLO, e inmediatamente se ejecuta la instrucción: puts(s);

¿Cuál de los siguientes literales se visualizará en la pantalla?

- PEPI
- PEPITO GRILLO
- PEPITO
- PEPIT
- Ninguno, porque hay un error de sintaxis en alguna instrucción.

9. Dado el siguiente código:

```
Cadena cad1="Hola", cad2;
strcpy(cad2,cad1);
if(strcmp(cad1,cad2))
 puts("Mensaje1");
else
 puts("Mensaje2");
```

- Hay un error de compilación
- Hay un error de ejecución
- Mensaje1
- Mensaje2
- Ninguna de las anteriores

10. Indique la salida del siguiente fragmento de código al ejecutarse:

```
char *s1, *s2, s3[]="3", s4[]= "4", s5[3];
s1=s3;
s2=s4;
strcpy (s5,s1);
strcat(s5, s2);
printf ("La salida es %s, %s, %s", s1, s2, s5);
```

- No existe error de compilación pero puede haber error de ejecución
- Dicho fragmento nunca puede ejecutarse porque existen errores de

compilación

- La salida es "3", "4", "34"
- Ninguna de las restantes respuestas es correcta
- La salida es 3, 4, 34

11. El valor que devuelve la expresión strcmp("2345","23345") es:

- Un valor entero menor que cero
- Un valor numérico resultado de 2345-23345
- Un valor entero mayor que cero
- Un valor numérico resultado de 23345-2345
- Ninguna es correcta

12. Al ejecutarse el siguiente código:

```
char a[]="24", b[]="230";
int r1, r2;

r1= strcmp(a,b);
strcat(a,b);
r2= strcmp(a,b);
r1= r1 && r2;
```

Los valores de r1 y r2 serán:

- No se puede saber
- Ninguna de las restantes respuestas es correcta
- r1 > 0 y r2 > 0
- r1 > 0 y r2 < 0
- r1 < 0 y r2 < 0

13. Al ejecutarse el siguiente código:

```
char *f()
{
	char tmp1[]= "P1", tmp2[]= "P2";
	int i;
	if (i>10) return (tmp1); else return (tmp2);
}
main()
{
	printf("%s %s", f(),f());
}
```

La salida en pantalla es:

- Existe un error de compilación
- Ninguna de las restantes soluciones es correcta
- Una cadena de longitud y contenido indefinidos
- Se mostrará alguno de los siguientes mensajes "P1 P1", "P1 P2", "P2 P2", "P2 P1"
- El programa dará un error al ejecutarse

14. Indica la salida del siguiente fragmento de código al ejecutarse:

```
char *s1,*s2,s3[]="3",s4[]="4";

strcpy(s1,s3);
strcpy(s2,s4);
printf("La salida es %s,%s",s1,s2);
```

- El programa daría un error de compilación
- La salida es 3,4
- Ninguna es cierta
- El programa no da error de compilación pero puede dar un error de ejecución
- La salida es "3","4"

15. Construye un programa que utilice todas las funciones del punto 1 mediante un menú
16. Escribe un programa que busque un carácter determinado en una cadena. Debe mostrar un número con la posición de la cadena en que fue encontrado el carácter (la primera posición es la cero), si no se encontró mostrará -1. Los datos de entrada serán una cadena y un carácter. Puede usar la cadena en la forma habitual, pero se le pide que use notación de punteros en lugar de los [] al momento de buscar el carácter.
17. Implementa el siguiente algoritmo para ordenar un array de caracteres. La idea es recorrer simultáneamente el array desde el principio y desde el final, comparando los elementos. Si los valores comparados no están en el orden adecuado, se intercambian y se vuelve a empezar el bucle. Si están bien ordenados, se compara el siguiente par. El proceso termina cuando los punteros se cruzan, ya que eso indica que hemos comparado la primera mitad con la segunda y todos los elementos estaban en el orden correcto. Habrá tres datos: char* vector, int nElementos, int ascendente o sea un puntero al inicio de la cadena, su largo y una variable que será 1 cuando se desee ordenar en forma ascendente, y cero en caso contrario. De nuevo, no se deben usar [] (índices), sólo punteros y aritmética de punteros.

Fuente: http://c.conclase.net

18. Analiza el siguiente segmento de código y explica que produce:

```
#include <stdio.h>
int main( int argc, char **argv )
{ char *p, *q;
  if ( argc != 2 || !*argv[1] )    exit(1);
  p = q = argv[1];
  while (*q)   q++;
    for ( q-- ; *p == *q && p < q; p++, q-- ) ;
     printf( "%s : %s\n", argv[1], (p >= q) ? "verdadero" : "falso"
); }
```

Fuente: http://www.epublish.cl/leus/articles/epublish_20020810181612.html

19. Analiza los siguientes programas, si hay errores, indícalos:

```
a)
#include "conio.h"
#include "stdio.h"
#include "ctype.h"

main(){
clrscr();
char cad1[100], *ptr_cad1;
char cad2[100], *ptr_cad2;
printf("Dame la cadena: ");
gets(cad1); ptr_cad1=cad1;
ptr_cad2=cad2;
```

```
do
{
*ptr_cad2=*ptr_cad1;
*ptr_cad2++;
} while(*ptr_cad1++);
ptr_cad2=cad2;
do
{ printf("%c",*ptr_cad2);
}while(*ptr_cad2++);
getch();
}
```

```
b)
#include "stdio.h"
main()
{ char *puntero="Hola";
for (int i=0;i<sizeof(puntero);
```

```
i++)
printf("%s\n",puntero+i);

getchar();
}
```

Fuentes: http://ejemplos.mis-algoritmos.com/ejercicios-con-punteros
http://www.foro.lospillaos.es/viewtopic.php?p=9610

20. Dadas las siguientes declaraciones:
    int x, int_array[MAX];
    x = int_array[4];
    explica qué traduce el compilador para acceder al cuarto elemento del arreglo y asignárselo a x.
21. ¿Qué diferencia hay entre el nombre de un arreglo y un puntero?.
22. ¿Cómo sabe el compilador el tamaño de un objeto al que apunta un puntero?.
23. Explica qué pasa si se olvida el carácter nulo como último carácter de una cadena.
24. Explica qué es una variable tal como la que se define en la siguiente declaración y dá un ejemplo de valor posible para la misma:
    int ***miVariable y trate de representarla gráficamente.
25. Explica el significado de las siguientes declaraciones:
    a) int (*uno)[12];
    b) int *dos[12];

Adaptado de:
http://materias.fi.uba.ar/7502E/download/guia06.pdf

26. Aritmética de punteros: resuelve

- ✓ Declara un puntero a char llamado B y otro llamado C
- ✓ Asígnale memoria a B para 30 caracteres
- ✓ Copia en B una palabra cualquiera
- ✓ Usando aritmética de punteros, modifica todos los caracteres de la cadena para que pase de minúscula a mayúscula.
- ✓ Usando aritmética de punteros C debe apuntar al carácter de la mitad de la cadena B
- ✓ Imprime C, que sale en pantalla?

27. Explica el funcionamiento de los siguientes programas y qué emite cada uno de ellos.

```
a)
#include <stdio.h>
#include <stdlib.h>
main()
```

```
{
char *nombre = "PEPITO PEREZ";
  int i=0;
   do
```

```
      printf("%c", *(nombre+i));
    while(*(nombre+i++));
printf("\n");
    puts(nombre);
b)
#include <stdio.h>
#include <stdlib.h>
main()
  {
    int a, *p;
    a=5;
    printf("%s\n", nombre);
    system("pause");
    return 0;
  }
    p=&a;
    *p+=7;
printf("\nEl valor final de a es:
%d\n", a);
    system("pause");
    return 0;
  }
```

28. El acrónimo de un nombre consiste en una cadena formada por las letras iniciales mayúsculas de cada palabra que aparece en dicho nombre. Se pide escribir un programa que, a partir de un nombre introducido por el usuario, construya una nueva cadena con su acrónimo y la muestre por pantalla. Por ejemplo, para el nombre "Informática Básica" se emitirá la cadena "IB", mientras que para el nombre "Fundamentos de Informática" se emitirá "FI".

29. Analiza el siguiente programa, qué hace?, si hay errores, señalarlos y modifica el programa para que funcione:

a)
```
#include "conio.h"
#include "stdio.h"
#include "ctype.h"

main(){
clrscr();
char cad1[100], *ptr_cad1;
char cad2[100], *ptr_cad2;
printf("Dame la cadena: ");gets(cad1); ptr_cad1=cad1;
ptr_cad2=cad2;
do {
 *ptr_cad2=*ptr_cad1;  *ptr_cad2++;
} while(*ptr_cad1++);
ptr_cad2=cad2;
do
{  printf("%c",*ptr_cad2);
}while(*ptr_cad2++);
getch();
}
```

b)
```
#include "stdio.h"
main()
{char *puntero="Hola";
for (int i=0;i<sizeof(puntero);i++)
    printf("%s\n",puntero+i);
getchar();
}
```

30. Construye un programa que permita ingresar un texto y resuelve:

    a. Contar la cantidad de letras de un texto que termina en punto (los textos se leen letra por letra).
    b. Contar la cantidad de palabras, separadas por uno o más espacios, de un texto que termina en punto.
    c. Dado un texto terminado en punto, determinar cuál es la vocal que aparece con mayor frecuencia.
    d. Dado un texto terminado en "/0" se pide determinar cuántas veces aparece determinada letra, leída de teclado.
    e. Dado un texto terminado en "/0" averiguar qué cantidad de letras tiene la palabra más larga. Supone que nunca sucede que la primera letra del texto es una "/0".
    f. Leer dos letras de teclado y luego un texto terminado en "/0". Se pide determinar la cantidad de veces que la primera letra precede a la segunda en el texto

31. Dado un texto de un telegrama que termina en punto:

a. Contar la cantidad de palabras que posean más de 10 letras
b. Informar la cantidad de veces que aparece cada vocal
c. Informar el porcentaje de espacios en blanco.

Nota: Las palabras están separadas por un espacio en blanco.

32. Dado un texto que finaliza en punto, se pide:
    a. La posición inicial de la palabra más larga,
    b. La longitud del texto,
    c. Cuantas palabras con una longitud entre 8 y 16 caracteres poseen más de tres veces la vocal “a”

    Nota: Las palabras pueden estar separadas por uno o más espacios en blanco. Puede haber varios espacios en blanco antes de la primera palabra y también después de la última. Se considera que una palabra finaliza cuando se encuentra un espacio en blanco o un signo de puntuación.

33. Escribe una función inversa que recibe una cadena como parámetro y devuelve los caracteres de la misma en orden inverso.

34. Escribe un programa que lea una frase y a continuación visualice cada palabra de la frase en columna, seguido del número de letras que componen cada palabra. La frase termina con un punto (.)

Ejemplo: Frase: La casa es linda.
Solución:

La 2
casa 4
es 2
linda 5