En los archivos de cabecera .h de los diferentes compiladores, existe un gran número de funciones para el tratamiento de las cadenas. Las que aquí veamos suelen estar en string.h Todas las cadenas de caracteres terminan en '/0'.

Partimos de dos cadenas de caracteres

```
char cadena1[100], cadena2[25];
```

a) strlen();

devuelve la longitud de la cadena, sin contar el carácter '\0' de finalización. También podemos:

```
sizeof(cadena)/sizeof(char);
```

nos devuelve el número de caracteres

```
//
#include <stdio.h>
#include <conio2.h>            //Dev-C++
#include <stdlib.h>
#include <string.h>

int main()
{
      char cadena[100] = "Hola";
      int elementos;
      elementos = sizeof(cadena)/sizeof(char);
      printf("El tamaño de la cadena es de                    %d elementos...", elementos);
      printf("\n\nPero contiene \"%s\", es decir %d caracteres...",cadena,\
            strlen(cadena));

      getch();
      return 0;}
```

b) strcat()

Añade la cadena2 a la cadena1

```
strcat(cadena1, cadena2);
```

Precaución: la cadena1 ha de ser de suficiente tamaño para poder almacenar a ambas al concatenarlas.

```
/* strcat()
    char *strcat(char*s1, const char *s2);
    Añade una copia de la cadena apuntada por s2 (incluyendo el carácter nulo) al final de la cadena
    apuntada por s1. El carácter inicial de s2 sobrescribe el carácter nulo al final de s1 VALOR DE
    RETORNO La función retorna el valor de s1. Si la copia hace que los objetos se superpongan,
    entonces el comportamiento no está definido.

*/
//Versión arrays

#include <stdio.h>
#include <string.h>
#include <conio2.h>            //Dev-C++
```

```
int main()
{
	char  s1[11] = "Hola";
	char  s2[11] = "amigos";

	printf(  "s1=%s\t",   s1   );  printf(  "s2=%s\n",
	s2  ); strcat( s1,  s2  );
	printf( "s1=%s\n", s1  );

	getch();
	return  0;}
/*
//Version  punteros
#include <string.h>
#include <stdio.h>
#include <conio2.h> //Dev-C++

int main(void)
{
	char  destination[25];
	char  *blank = " ", *c = "amigos", *cadena = "Hola";

	strcpy(destination, cadena); strcat(destination,
	blank); strcat(destination, c);

	printf("%s\n", destination);
	getch();
	return  0;
}  */
```

c)strcpy()

Copia cadena2 en cadena1:

strcpy(cadena1, cadena2);

```
#include <stdio.h>
#include <string.h>
#include <conio2.h> //Dev-C++

int main(void)
{
    char  string[10];
    char  *str1 = "abcdefghi";

    strcpy(string, str1);  printf("%s\n", string);
    getch();
    return  0;
}
```

d) strncpy()

Copia n caracteres de cadena2 en cadena1

strncpy(cadena1, cadena2, n);

el valor de n es entero.

```
#include <stdio.h>
#include <string.h>
#include <conio2.h>

int main(void)
{
    char  string[10];
    char  str1[] = "abcdefghi";
    strncpy(string, str1, 3); printf("%s\n", string);
    printf("\n%s", str1);  getch();
    return  0;}
```

e) strcmp()

Compara dos cadenas de caracteres:

strcmp(cadena1, cadena2);

La función devuelve 0 si son iguales, -1 si cadena 1 es menor que cadena 2 (es anterior alfabéticamente, cadena1 a cadena2) y +1 en caso contrario. Diferencia mayúsculas y minúsculas.

/*strcmp() Es una funciónque compara las cadenas de caracteres, se utiliza para ordenarlas de mayor a menor, según el numero del carácter en el código ASCII. Se utiliza de
este modo strcmp(cad1,cad2) si cad1=cad2 envía un cero, si cad1<cad2 envía un numero negativo, si cad1>cad2 envía un numero positivo. Esta función diferencia mayúsculas de minúsculas considera
mayores las letras minúscula debido a que están situadas después de las mayúsculas en el código ascii,
esta función puede utilizarse paraordenar cadenas de caracteres alfabéticamente. si bien hay que tener en cuenta que las palabras estén todas en mayúscula o minúscula. si no queremos que

considere esta diferencia podemos utilizar las funciones strcmpi o stricmp*/

```
#include
<string.h>
#include
<stdio.h>
#include <conio2.h>            //Dev-
C++

int
main(voi
d)
{
        char cadena1[100] = "BBB", cadena2[100] = "bbb", cadena3[100] = "ccc";
        int ptr;

        ptr = strcmp(cadena1, cadena2);
        if (ptr > 0) printf("CADENA %s ES MAYOR QUE CADENA %s\n",cadena2,cadena1);
    else
         printf("CADENA %s ES MENOR QUE CADENA %s\n",cadena1,cadena2);

    ptr = strcmp(cadena2, cadena3);

    if (ptr > 0) printf("CADENA %s ES MAYOR QUE CADENA %s\n",cadena2,cadena3);
    else
         printf("CADENA %s ES MENOR QUE CADENA %s\n",cadena2,cadena3);

    getch();
    return 0; }
```

f) strcmpi()

Como el anterior, pero sin diferenciar mayúsculas y minúsculas.

/* Compara los caracteres del código ASCII, sin considerar mayúsculas y minúsculas. Si el carácter es el mismo le considera igual este en mayúscula o minúscula. Lo coloca c onsiderando mayores a los primeros s í m b o l o s del c ó d i g o ASCII exactamente de mayor a menor numero ASCII. Van de mayor a menor. Funciona así strcmpi (cad1,cad2) si cad1<cad2 envía un numero negativo , si cad1>cad2 envía un numero positivo, si cad1=cad2 e n v í a un 0. ESTA FUNCION HACE LO MISMO QUE LA FUNCION stricmp(cad1,cad2)*/

```
#include <string.h>
#include <stdio.h>
#include <conio2.h> //Dev-C++
int main(void)
{
    char buf1[] = "BBB", buf2[] = "bbb";
    int ptr;

    ptr = strcmpi(buf2, buf1);

    if (ptr > 0)
         printf("cadena %s es mayor que cadena %s\n",buf2,buf1);

    if (ptr < 0)
         printf("cadena %s es menor que cadena %s\n",buf2,buf1);
```

```
    if (ptr == 0)
        printf("cadena %s es igual que  cadena %s\n",buf2,buf1);
    getch();
    return  0;}
```

g) stricmp()

Lo mismo que el anterior.

/* Compara los caracteres del código ASCII, sin considerar mayúsculas y minúsculas. Si el carácter es el mismo le considera igual este en mayúscula o minúscula. Lo coloca considerando mayores a los primeros símbolos del código ASCII exactamente de mayor a menor numero ASCII. Van de mayor a menor. Funciona así strcmpi (cad1,cad2) si cad1<cad2 envía un numero negativo , si cad1>cad2 envía un numero positivo, si cad1=cad2 envía un 0. ESTA FUNCION HACE LO MISMO QUE LA FUNCION strcmpi(cad1,cad2)*/

```
#include <string.h>
#include <stdio.h>
#include <conio2.h>
int main(void)
{
    char  buf1[] = "d", buf2[] = "D";
    int ptr;

    ptr = stricmp(buf2, buf1);

    if (ptr > 0)
        printf("CADENA %s ES  MAYOR QUE  %s \n",buf2,buf1);

    if (ptr < 0)
        printf("CADENA %s ES  MENOR  QUE  %s \n",buf2,buf1);

    if (ptr == 0)
        printf("LAS DOS  CADENAS  SON  IGUALES %s=%s\n",buf1,buf2);
    getch();
    return  0; }
```

h) strncmp()

Compara los n caracteres primeros de una cadena.

strncmp(cadena1, cadena2, n)

devuelve un entero con las mismas características que strcmp()

/*strncmp() Compara "n" caracteres (posteriores al caracter nulo),no se tiene en cuenta de la cadena apuntada S1 con la cadena apuntada S2. Valor de retorno:La funciónretornaun numero entero mayor,igual o menor que 0,apropiadamente según la cadena S1 es mayor,igual o menor que la cadena apuntada por S2.*/

```
#include <stdio.h>
#include <stdlib.h>
#include <conio2.h>
int main()
{
   char  s1[8]="Vanessa";
   char  s2[8]="Aurora";
   int i;

   printf("\nLa primera cadena es %s",s1);  printf("\nLa segunda cadena es %s",s2);
   i = strncmp(s1,s2,4);
        printf("\nLas 4 primeras letras de  %s son  ",s1);
        if(i<0)
             printf("\nmenores que");
else if(i>0)
                          printf("\nmayores que");
                           else
                                printf("\niguales a  ");
                                      printf(" %s",s2);
   getch();
   return  0; }
```

i) strcoll()

Compara la cadena1 con la cadena2, ambas interpretadas acordes a la categoría LC_COLLATEde la localidad actual. La función retorna un número entero mayor, igual, o menor que cero, apropiadamente según la cadena cadena1 es mayor, igual, o menor que la cadena2 (afabéticamente hablando),

```
/*strcoll() La función retorna un número entero mayor, igual, o menor que cero, apropiadamente
según la cadena uno es mayor, igual, o menor que la cadena dos, cuando ambas son interpretadas
apropiadamente según        la localidad actual.*/

#include <stdio.h>
#include <string.h>
#include <conio2.h>

int main(void)
{
      char  dos[] = "Leon",                  uno[] = "Castilla";
      int control;

      control = strcoll(uno, dos);

      if (control == 0)
            printf("Las cadenas son  iguales...\n\n");
      if (control <                0)
            printf("%s es anterior a %s\n",  uno,  dos);
      if (control >                0)
            printf("%s es anterior a %s\n",  dos,  uno);
            getch();
      return  0;}
```

i) strchr() y strrchr.cpp

Localiza la primera aparicion de c en la cadena1:

strchr(cadena1,c)

si el carácter c no existe, devuelve NULL

/*strstr() Localiza la primera aparición de un caracter (convertido a unsigned char) en la cadena apuntada por s (incluyendo el carácter nulo). VALOR DE RETORNO: La función retorna un puntero a partir del carácter encontrado. Si no se ha encontrado el carácter, c, entonces retorna un puntero null. */

```
#include <stdio.h>
#include <string.h>
#include <conio2.h>            //Dev-C++
int main()
{
      char  s[20] = "Hola amigos";
      char  c = 'm';            //si m no exixte, devuuelve null

      printf( "s=%s\t",  s );
      printf( "c=%c\n",  c );
      printf( "strchr=%s\n", strchr(  s, c ) );

      getch();
      return  0;}
```

j) strrchr()

Localiza la última aparición del carácter de la variable c en cadena1:

strrchr(cadena, c)

/*la funcion strrchr busca la letra q le indicas e imprime a partir de la ultima como ella que haya (incluida) si se pone una letra q no esta para q la busque, pone NULL*/

```
  #include <stdio.h>
#include <string.h>
#include <conio2.h>

int main()
{
      char  cad[] = "strrchr  imprime a partir de  la letra q introduces";
      char  letra;

    printf("Intro una  letra: ");
    fflush(stdin);
      scanf("%c", &letra);


      printf( "\n\ncad=  %s\t", cad  );
      printf( "\n\nLetra  para  buscar= %c\n", letra );
      printf( "\nstrrchr=  %s\n", strrchr( cad,  letra ) );
      getch();
       return  0;}
```

k) strspn()

nos dice cuantos caracteres de la cadena hay en la cadena2

strspn(cadena1, cadena2);

/*Busca el primer elemento "s1[i]" en la cadena "s1" que no sea igual a ninguno de los elementos de "s2" y devuelve "i".*/

```
#include <stdio.h>
#include <string.h>
#include <conio2.h>

int main(void)
{
    char cadena1[] = "567890"; char
    cadena2[] = "5111167"; int numero;

    numero = strspn(cadena1, cadena2);
    printf("nos dice cuantos caracteres de la cadena2 hay en la cadena1,que son %d\n", numero);
    getch();
    return 0;}
```

l) strrev()

Invierte la cadena1:

strrev(cadena);

//strrev escribe la cadena al revés

```
#include <string.h>
#include <stdio.h>
#include <conio.c>

int main()
{
    char cad[50];

    printf("Intro frase: ");
    gets(cad);
    printf("\n\nLa cadena es: %s\n", cad);
    strrev(cad);
    printf("\ncon la funcion string:                    %s\n", cad);

    getch();
    return 0;}
```

m) strstr()

Devuelve la primera coincidencia de cadena2 en cadena1 o NULL si no existen coincidencias.

strstr(cadena1, cadena2)

/*Si se encuentra la totalidad de la segunda cadena en la primera cadena escribe a partir de donde coincide la primera y segunda cadena.*/

```
#include <stdio.h>
#include <string.h>
#include <conio2.h>            //Dev-C++
int main()
{
      char  s1[13] = "Hola a todos";
      char  s2[3] = "ol";

      printf( "s1=%s\n", s1  );
      printf( "s2=%s\n", s2  );
      printf( "strstr(s1,s2) = %s\n", strstr( s1,  s2  ) );
      getch();
      return  0;}
```

n) strtok()

Permite separar una cadena cadena1 en partes usando como delimitadores los caracteres de otra cadena2.

/* Permite separar una cadena "s1" en partes usando como delimitadores los caracteres de otra "s2". Cada vez que se invoca, devuelve un puntero a la siguiente palabra de la cadena "s1". Devuelve un puntero nulo cuando no existe ninguna palabra que devolver. La primera vez que se llama a "strtok", se utiliza realmente "s1" en la llamada. Las llamadas posteriores utilizan un puntero nulo como primer argumento. Se puede usar un conjunto diferente de delimitadores en cada llamada. Esta función modifica la cadena "s1". Cada vez que se encuentra una palabra, se pone un carácter nulo donde estaba el delimitador.*/

```
 #include <string.h>
 #include <stdio.h>
 #include <conio2.h>

 int main(void)
 {
       char  input[16] = "abc,d,juan";
       char  *p;
       p = strtok(input, ",");
       if (p)          printf("%s\n", p);
       p = strtok(NULL, ",");
       if (p)          printf("%s\n", p);
       p = strtok(NULL, ",");
       if (p)          printf("%s\n", p);

       getch();
       return  0;}
```

o) strsert()

Cambia los caracteres de cadena1 por el carácter contenido en la variable char c:

```
strset(cadena1, c);
```

//cambia toda la cadena por la letra que se le indique

```
#include <stdio.h>
```

```
#include <string.h>
#include <conio2.h> //Dev-C++

int main()
{
      char  cad[50];
      char  letra;

      printf("Intro frase: ");
        gets(cad);
      printf("Intro una  letra: ");
        scanf("%c", &letra);
      printf("\n\nLa cadena es:  %s\n", cad);
        strset(cad, letra);
      printf("Con la funcion strset:                    %s\n",  cad);

      getch();
      return  0;}
```

p) strpbrk( )

Localiza la primer aparición del primera carácter de cadena2 en cadnea 1:

```
strpbrk(cadena1, cadena2);
```

/*Si se encuentra alguna letra de la primera cadena en la segunda cadena imprime desde esa letra de la primera cadena hasta el final de la misma*/

```
#include <stdio.h>
#include <string.h>
#include <conio2.h>
int main()
{
      char  s1[13] = "Hola a todos";
      char  s2[5] = "hola";
      printf( "s1=%s\n", s1  );
      printf( "s2=%s\n", s2  );
      printf( "strpbrk(s1,s2) = %s\n",  strpbrk(  s1,  s2  ) );
      getch();
      return  0;}
```

q) strerror()

Maneja los errores del sistema. Convierte el número de error en errnum a un mensaje de error (una cadena de caracteres).

```
strerror(  errnum  )
```

2) OTRAS FUNCIONES ESPECIALES PARA EL TRATAMIENTO DE CADENAS DE CARACTERES

En programación en Lenguaje C es frecuente el hecho de tener que comprobar si un determinado carácter es del tipo solicitado. Por ejemplo: si en una variable entera debo introducir valores enteros, ¿cómo evito la introducción de otro tipo de valores como caracteres…? A este concepto

frecuentemente se le denomina **Validación de datos de entrada**. Para conocer si un determinado es adecuado contamos con una serie de funciones especiales, a veces llamadas **macros**, que se encuentran en **ctype.h**

a)isdigit()

Comprueba que lo que introducimos es un número decimal.

```
/*************ISDIGIT***********/

//Comprueba sin un carácter es un dígito decimal.
//Devuelve un valor no nulo es un dígito y un 0 si es otro caracter

#include <stdio.h>
#include <ctype.h>
#include <conio2.h>                 //Dev-C++


int main()
{
      char  cadena[] = ";0ñfáR(4h&~?RÛ1/";
      int i;

      for(i = 0; cadena[i]; i++)
            printf("%c, %d\n", cadena[i], isdigit(cadena[i]));

      getch();
      return  0;}
```

b) isalnum()

Comprueba sin un carácter es alfanumérico. Es una macro que verifica el entero c pertenece al rango de letras (*A* a *Z* ó *a* a *z*) o al de dígitos (*0* a *9*), por defecto

```
/*Macro isalnum ANSI C int isalnum(int c); Comprueba sin un carácter es alfanumérico. isalnum es una macro
que verifica el entero c pertenece al rango de letras (A a Z o a a z) o al de dígitos (0 a 9), por defecto. La
verificación se hace mediante una tabla, y su comportamiento depende le la categoría LC_CTYPE
actual. Valor de retorno: El valor de retorno será no nulo si c es una letra o un número, y cero en
caso contrario.          */

#include <stdio.h>
#include <ctype.h>
#include<conio2.h> //Dev-C++

int  main()
{
      char  cadena[] = "hola7 =";
      int i;

      for(i = 0; cadena[i]; i++)
            printf("%c, %d\n", cadena[i], isalnum(cadena[i]));
      getch();
      return  0;}
```

c) isalpha()

Comprueba sin un carácter es alfabético. Es una macro verifica si el dato c pertenece al rango de letras (*A* a *Z* o *a* a *z*), por defecto.

/*Macro isalpha ANSI C int isalpha(int c); Comprueba sin un carácter es alfabético. isalpha es una macro que verifica si un valor entero pertenece al rango de letras (A a Z o a a z), por defecto.Valor de retorno: El valor de retorno será no nulo si c es una letra y cero en caso contrario. */

```
#include <stdio.h>
#include <ctype.h>
#include <conio2.h>                //Dev-C++

int main()
{
      char  cadena[] = "2qq7qqñ 12345";
      int i;

      for(i = 0; cadena[i]; i++)
            printf("%c, %d\n", cadena[i], isalpha(cadena[i]));
      getch();

      return  0;}
```

d) iscntrl()

Comprueba sin un carácter es de control.Verifica que un caracter pertenece al rango caracteres de control (*0x00* a *0x1F* y *0x7F*, es decir de ASCII 0 a ASCII 31 y el 127)

/*iscntrl es una macro que verifica el entero c pertenece al rango caracteres de control. El valor de retorno será no nulo si es un carácter "delete" o un carácter de control.*/

```
#include <stdio.h>
#include <ctype.h>
#include <conio2.h> //Dev-C++

int main()
{
      char  cadena[] = ";0ñs\003áR(h\177&~?\037RÛ1/";
      int i;

      for(i = 0; cadena[i]; i++)
            if(isprint(cadena[i]))
                  printf("%c, %d\n", cadena[i], iscntrl(cadena[i]));
            else
                  printf("%d, %d\n", cadena[i], iscntrl(cadena[i]));

      getch();
      return  0;
}
```

e) ispunct()

Comprueba sin un carácter es un signo de puntuación. Verifica que el dato pertenece al rango

de caracteres de los signos de puntuación, que por defecto son todos menos los alfanuméricos y el espacio'. El valor de retorno será no nulo si el carácter es un signo de puntuación. En los demás casos, devuelve cero.

/*Comprueba sin un carácter es un signo de puntuación. Verifica que el dato pertenece al rango de caracteres de los signos de puntuación, que por defecto son todos menos los alfanuméricos y el espacio'. El valor de retorno será no nulo si el carácter es un signo de puntuación. En los demás casos, devuelve cero.*/

```
#include <stdio.h>
#include <ctype.h>
#include <conio2.h> //Dev-C++

int main()
{
      char  cadena[] = "aAb.Bc/Cd(D3:1&,75%$/";
      int i;

      for(i = 0; cadena[i]; i++)
            printf("%c, %d\n", cadena[i], ispunct(cadena[i]));
     getch();
      return  0;}
```

f) isgraph()

Comprueba sin un carácter es imprimible. Verifica si un carácter pertenece al rango de caracteres con representación gráfica, que por defecto son todos menos el espacio ' '. El valor de retorno será no nulo si c es un carácter gráfico.

/*Comprueba sin un carácter es imprimible. Verifica si un carácter pertenece al rango de caracteres con representación gráfica, que por defecto son todos menos el espacio ' '. La función devuelve un valor no nulo si el carácter es gráfico.*/

```
#include <stdio.h>
#include <ctype.h>
#include <conio2.h> //Dev-C++

int  main()
{
      char  cadena[] = ";0 ñsáR(h &~?RÛ  1/";
      int i;

      for(i = 0; cadena[i]; i++)
            printf("%c, %d\n", cadena[i], isgraph(cadena[i]));

      getch();
      return  0;}
```

g) isprint()

Comprueba sin un carácter es imprimible. isgraph es una macro que verifica si el carácter pertenece al rango de caracteres imprimibles, que por defecto son todos los caracteres imprimibles, incluido el espacio ' '. El valor de retorno será no nulo si c es un carácter imprimible.

/*isprintf() Comprueba sin un carácter es imprimible. isgraph es una macro que verifica si el caracter pertenece al rango de caracteres imprimibles, que por defecto son todos los caracteres imprimibles, incluido el espacio ' '. El valor de retorno será no nulo si c es un carácter imprimible.*/

```
#include <stdio.h>
#include <ctype.h>
#include <conio2.h>

int main()
{
      char  cadena[] = ";0 ñ\003sáR(h &~?\177RÛ  1/";
      int i;

      for(i = 0; cadena[i]; i++)
            printf("%c, %d\n", cadena[i], isprint(cadena[i]));

      getch();
      return  0;}
```

h) isspace()

Comprueba si un carácter es de tipo espacio. isspace es una macro que verifica que el character pertenece al grupo de caracteres de espacio, ' ', tab, retorno de carro, nueva línea, tabulador vertical o salto de página.

```
#include <conio.h>
#include <stdio.h>
#include <ctype.h>

int main(void)
{
    char  c = '\a';                  //secuencia de escape sonido del sistema char  d = '\n';        //salto de
    línea
    if (isspace(c))
        printf("%c es un caracter espacio\n",c);
    else
        printf("%c no es un caracter espacio\n",c);
    if (isspace(d))
        printf("%c ES un caracter espacio\n",d);
    else
printf("%c no es un caracter espacio\n",d);

    getch();
    return  0;}
```

i) isupper()

Comprueba sin un carácter es de tipo mayúscula. islower es una macro que verifica si el caracter pertenece al rango de caracteres de letras mayúsculas, que por defecto son los que están en el rango *A* a *Z*. El valor de retorno será no nulo si c es un carácter en mayúscula.

```
//issuper()
#include <stdio.h>
```

```
#include <ctype.h>
#include <conio2.h> //Dev-C++

int main()
{
	char  cadena[] = "aAbBcCdD31&75$/";
	int i;

	for(i = 0; cadena[i]; i++)
		printf("%c, %d\n", cadena[i], isupper(cadena[i]));

	getch();
	return  0;}
```

j) islower()

Comprueba sin un carácter es de tipo minúscula. islower es una macro que verifica si el caracter pertenece al rango de caracteres de letras minúsculas, que por defecto son los que están en el rango *a* a *z*. El valor de retorno será no nulo si es un carácter en minúscula.

/*Comprueba sin un carácter es de tipo mayúscula. islower es una macro que verifica si el caracter pertenece al rango de caracteres de letras mayúsculas, que por defecto son los que están en el rango A a Z. El valor de retorno será no nulo si c es un carácter en mayúscula.*/

```
#include <stdio.h>
#include <ctype.h>
#include <conio2.h>

int main()
{
	char  cadena[] = "aAbBcCdD31&75$/";
	int i;

	for(i = 0; cadena[i]; i++)
		printf("%c, %d\n", cadena[i], isupper(cadena[i]));

	getch();
	return  0; }
```

k) isxdigit()

Comprueba sin un carácter es un dígito hexadecimal. isxdigit es una macro que verifica el entero c pertenece al rango caracteres de dígitos decimales y hexadecimales, por defecto del rango *'0'* a *'9', 'a'* a *'f'* y *'A'* a *'F'*.

```
#include <conio.h>
#include <stdio.h>
#include <ctype.h>

int main(void)
{
    char  c = 'C';
```

```
    if (isxdigit(c))
        printf("%c es un digito hexadecimal\n",c);
    else printf("%c no es un digito hexidecimal digit\n",c);

    getch();
    return 0;}
```

l) isascii()

Comprueba sin un carácter pertenece al ASCII de 7 bits (0 a 127)

/*Macro isascii ANSI C int isascii(int c);Comprueba sin un carácter pertenece al ASCII de 7 bits.isascii es una macro que verifica el entero c pertenece al rango de (0 a 127). Valor de retorno:El valor de retorno será no nulo si c está en el rango entre 0 y 127, en hexadecimal entre 0x00 y 0x7f.*/

```
#include <stdio.h>
#include <ctype.h>
#include <conio2.h>

int main()
{
      char cadena[] = "139 eñ";
      int i;

      for(i = 0; cadena[i]; i++)
            printf("%c, %d\n", cadena[i], isascii(cadena[i]));

      getch();
  return 0;}
```

3)FUNCIONESDEINTERCAMBIONUMEROS/CARACTERES

a) itoa()

Transforma un entero a cadena.

```
//itoa.cpp transforma un entero a cadena (itoa = Integer TO Ascii)
//ver ultoa.cpp

#include <stdlib.h>
#include <stdio.h>
#include <conio.c>

int main(void)
{
      int numero = 127;
      char cadena[25];

      itoa(numero, cadena, 127);
      printf("Numero entero = %d Cadena                = %s\n", numero+3, cadena);
      getch();
      return 0;}
```

b) atoi()

Convierte una cadena a número entero

/*atoi trasforma una cdena de caracteres numéricos a entero, con el que se puede hacer operaciones.*/

```
//Ver atof.cpp y atol.cpp
  #include <stdlib.h>
  #include <stdio.h>
  #include <conio2.h>


  int main(void)
  {
        int n;
        char cadena_numeros[] = "12345.67";
        n = atoi(cadena_numeros);
        printf("Como cadena: = %s\n", cadena_numeros);
        printf("Como número entero (aumentado en tres unidades) = %d\n", n+3);
        getch();
        return 0;}
```

c) atof()

Convierte una cadena a una representación numérica float

```
//atof: convierte a float una cadena de caracteres de numeros.
//ver atoi.cpp y atol.cpp
  #include <stdlib.h>
  #include <stdio.h>
  #include <conio2.h>

  int main(void)
  {
        float f;
        char cadena_numeros[] = "12345.67";

        f = atof(cadena_numeros);
        printf("Como cadena de numeros: = %s", cadena_numeros+3);
        printf("\nComo dato tipo float (aumentado en tres unidades): \
                    = %.2f\n", f+.01);
        getch();
        return 0;}
```

d) atol()

```
//atol.cpp Transforma una cadena de numeros a dato tipo long
//Ver atoi.cpp, atof.cpp


#include <stdlib.h>
#include <stdio.h>
#include <conio.c>

int main(void)
  {
        long l;
```

```
    char  cadena_numeros[] = "98765432";

    l = atol(cadena_numeros);
    printf("Como cadena:= %s\n", cadena_numeros);  printf("Como valor long (aumentado
    en  3):= %ld\n", l+3); getch();
    return(0);}
```

d) itoa()

```
//itoa.cpp transforma un entero a cadena (itoa = Integer TO Ascii)
//ver ultoa.cpp

#include <stdlib.h>
#include <stdio.h>
#include <conio.c>

int main(void)
{
    int numero = 127;
    char  cadena[25];

    itoa(numero, cadena, 127);
    printf("Numero entero = %d Cadena                = %s\n",  numero+3, cadena);
    getch();
    return  0;}
```

e)ultoa()

```
//ultoa.cpp          transforma un número a cadena

#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

int main( void )
{
    long numero = 123456789L;
    char  string[25];

    ultoa(numero,string,10);       //En Dev-C++  es _ultoa, en  otros es ultoa
    printf("cadena = %s unsigned long = %lu\n", string, numero+3);
    getch();
    return  0;}
```

Como funciones complementarias pueden considerarse toupper() y tolower() que transforman caracteres individuales a mayúsculas o minúsculas respectivamente. Para transformas una cadena a mayúsculas o minúsculas, se pueden emplear estas funciones para realizar transformaciones individuales mediante un bucle que recorra la cadena.

Tolower()

/**********TOLOWER**********/

Convierte un carácter, en un parámetro entero ch, a minúscula. Valor de retorno: ch debe estar en el rango 0 a 255, y si está entre A y Z lo convierte a su equivalente en el rango a a z, el resto de los valores no son modificados. El valor de retorno es el valor convertido si ch era una mayúscula, o

el valor original en caso contrario. Nota: los caracteres en acentuados, o con diéresis, en mayúscula y la Ñ no sufren modificaciones.*/

```
#include <stdio.h>
#include <ctype.h>
#include <conio2.h>
#include <string.h>
int main()
{
char cadena[] = "ESTO ES UNA CADENA PARA DEMOSTRAR LA FUNCION DE TOLOWER";
    int i;

    for(i = 0; i<strlen(cadena); i++)
        cadena[i] = tolower(cadena[i]);

    printf("%s\n", cadena);
    getch();
    return 0; }
```

toupper()

/*****************************TOUPPER*******************************/

Convierte un carácter, en un parámetro entero ch, a mayúscula. Valor de retorno: ch debe estar en el rango 0 a 255, y si está entre a y z lo convierte a su equivalente en el rango A a Z, el resto de los valores no son modificados. El valor de retorno es el valor convertido si ch era una minúscula, o el valor original en caso contrario.

Nota: los caracteres en acentuados, o con diéresis, en minúscula y la ñ no sufren modificaciones.*/

```
#include <stdio.h>
#include <ctype.h>
#include <conio2.h>
#include <string.h>
int main()
{
    char    cadena[]= "esto es una   cadena para demostrar la funcion de toupper";
    int i;

    printf("Tamaño de cadena              %d", sizeof(cadena));
    printf("\nTamaño del dato char %d", sizeof(char)); printf("\nLongitud de
    cadena %d\n\n", strlen(cadena));

    for(i = 0; i<sizeof(cadena)/sizeof(char); i++) cadena[i] = toupper(cadena[i]);

    printf("%s\n", cadena);
    getch();
    return 0;}
```

Para pasar cadenas de caracteres a mayúsculas, tenemos la función

strupr()

/*convierte las letras minúsculas en una secuencia de letras mayúsculas*/

```
#include <stdio.h>
```

```
#include <string.h>
#include <conio2.h>

int main(void)
{
      char  string[] = "abcdefgh", *ptr;
      printf("las letras son:  abcdefgh\t");
      /* convierte la cadena a mayusculas*/
      ptr = strupr(string); printf("%s\n", ptr);
      getch();
      return  0; }
```

Para pasar de mayúsculas a minúsculas, tenemos

strlwr()

/*convierte las letras mayúsculas a minúsculas*/

```
#include <stdio.h>
#include <string.h>
#include <conio2.h>

int main(void)
{
      char  string[] = "ASAASSR RE QWE", *ptr;
      printf("las letras son:  ASAASSR RE QWE\t");
      /* convierte la cadena a mayusculas*/
      ptr = strlwr(string);
      printf("\nPero convertidas...                    %s\n",  ptr);
      getch();
      return  0;
}
```