FUNCIONES DE LIBRERIAS DEL LENGUAJE C

FUNCIONES DE CARACTERES Y CADENAS

La biblioteca estándar de C tiene un rico y variado conjunto de funciones de manejo de caracteres y caracteres. En una implementación estándar, las funciones de cadena requieren el archivo de cabecera STRING.H, que proporciona sus prototipos. Las funciones de caracteres utilizan CTYPE.H, como archivo de cabecera.

ISALPHA: Devuelve un entero. DISTINTO DE CERO si la variable es una letra del alfabeto, en caso contrario devuelve cero. Cabecera: <ctype.h>.
int isalpha(variable_char);

ISDIGIT: Devuelve un entero. DISTINTO DE CERO si la variable es un número (0 a 9), en caso contrario devuelve cero. Cabecera <ctype.h>.
int isdigit(variable_char);

ISGRAPH: Devuelve un entero. DISTINTO DE CERO si la variable es cualquier carácter imprimible distinto de un espacio, si es un espacio CERO. Cabecera <ctype.h>.
int isgraph(variable_char);

ISLOWER: Devuelve un entero. DISTINTO DE CERO si la variable esta en minúscula, en caso contrario devuelve cero. Cabecera <ctype.h>.
int islower(variable_char);

ISPUNCT: Devuelve un entero. DISTINTO DE CERO si la variable es un carácter de puntuación, en caso contrario, devuelve cero. Cabecera <ctype.h>
int ispunct(variable_char);

ISUPPER: Devuelve un entero. DISTINTO DE CERO si la variable esta en mayúsculas, en caso contrario, devuelve cero. Cabecera <ctype.h>
int isupper(varible_char);

EJEMPLO: Cuenta el numero de letras y números que hay en una cadena. La longitud debe ser siempre de cinco por no conocer aún la función que me devuelve la longitud de una cadena.

```
#include<stdio.h>
#include<ctype.h>
void main(void)
{
        int ind,cont_num=0,cont_text=0;
        char temp;
        char cadena[6];
        clrscr();
        printf("Introducir 5 caracteres: ");
        gets(cadena);
        for(ind=0;ind<5;ind++)
```

```
	{
		temp=isalpha(cadena[ind]);
		if(temp)
			cont_text++;
		else
			cont_num++;
	}
	printf("El total de letras es %d\n",cont_text);
	printf("El total de números es %d",cont_num);
	getch();
}
```

EJEMPLO: Utilizando el resto de funciones nos va a dar información completa del valor que contiene la variable.

```
#include <stdio.h>
#include <ctype.h>
void main(void)
{
	char letra;
	clrscr();
	printf("Introducir valor: ");
	letra=getchar();
	if(isdigit(letra))
		printf("Es un número");
	else
	{
		if(islower(letra))
			printf("Letra en minúscula");
		if(isupper(letra))
			printf("Letra en mayúscula");
		if(ispunct(letra))
			printf("Caracter de puntuación");
		if(!isgraph(letra))
			printf("Es un espacio");
	}
	getch();
}
```

MEMSET: Inicializa una región de memoria (buffer) con un valor determinado. Se utiliza principalmente para inicializar cadenas con un valor determinado. Cabecera <string.h>.
memset (var_cadena,'carácter',tamaño);

STRCAT: Concatena cadenas, es decir, añade la segunda cadena a la primera, la primera cadena no pierde su valor origina. Lo único que hay que tener en cuenta es que la longitud de la primera cadena debe tener la longitud suficiente para guardar la suma de las dos cadenas. Cabecera <string.h>.
strcat(cadena1,cadena2);

STRCHR: Devuelve un puntero a la primera ocurrencia del carácter especificado en la cadena donde se busca. Si no lo encuentra, devuelve un puntero nulo. Cabecera <string.h>.
strchr(cadena,'carácter');
strchr("texto",'carácter');

STRCMP: Compara alfabéticamente dos cadenas y devuelve un entero basado en el resultado de la comparación. La comparación no se basa en la longitud de las cadenas. Muy utilizado para
comprobar contraseñas. Cabecera <string.h>.
strcmp(cadena1,cadena2);
strcmp(cadena2,"texto");

RESULTADO	VALOR DESCRIPCIÓN
Menor a Cero	Cadena1 menor a Cadena2.
Cero	Las cadenas son iguales.
Mayor a Cero	Cadena1 mayor a Cadena2.

STRCPY: Copia el contenido de la segunda cadena en la primera. El contenido de la primera se pierde. Lo único que debemos contemplar es que el tamaño de la segunda cadena sea menor o igual a la cadena donde la copiamos. Cabecera <string.h>.
strcpy(cadena1,cadena2);
strcpy(cadena1,"texto");

STRLEN: Devuelve la longitud de la cadena terminada en nulo. El carácter nulo no se contabiliza. Devuelve un valor entero que indica la longitud de la cadena. Cabecera <string.h>.
variable=strlen(cadena);

STRNCAT: Concatena el número de caracteres de la segunda cadena en la primera cadena. La primera no pierde la información. Hay que controlar que la longitud de la primera cadena debe tener longitud suficiente para guardar las dos cadenas. Cabecera <string.h>.
strncat(cadena1,cadena2,nº de caracteres);

STRNCMP: Compara alfabéticamente un número de caracteres entre dos cadenas. Devuelve un entero según el resultado de la comparación. Los valores devueltos son los mismos que en la función strcmp. Cabecera <string.h>.
strncmp(cadena1,cadena2,nº de caracteres);
strncmp(cadena1,"texto",nº de caracteres);

STRNCPY: Copia un número de caracteres de la segunda cadena a la primera. En la primera cadena se pierden aquellos caracteres que se copian de la segunda. Cabecera <string.h>.
strncpy(cadena1,cadena2,nº de caracteres);
strncpy(cadena1,"texto",nº de caracteres);

STRRCHR: Devuelve un puntero a la última ocurrencia del carácter buscado en la cadena. Si no lo encuentra devuelve un puntero nulo. Cabecera <string.h>.

strrchr(cadena,'carácter');
strrchr("texto",'carácter');

STRPBRK: Devuelve un puntero al primer carácter de la cadena que coincida con algún carácter de la cadena a buscar. Si no hay correspondencia devuelve un puntero nulo. Cabecera <string.h>.
strpbrk("texto","cadena_de_busqueda");
strpbrk(cadena,cadena_de_busqueda);

TOLOWER: Devuelve el carácter equivalente al de la variable en minúsculas, si la variable es una letra y la deja como esta si la letra esta en minúsculas. Cabecera <ctype.h>.
variable_char=tolower(variable_char);

TOUPPER: Devuelve el carácter equivalente al de la variable en mayúsculas, si la variable es una letra y la deja como esta si la letra esta en minúsculas. Cabecera <ctype.h>.
variable_char=toupper(variable_char);

EJEMPLO: Copia, concatena, mide e inicializa cadenas.

```
#include <stdio.h>
#include <string.h>
void main(void)
{
	char origen[10], destino[10];
	clrscr();
	printf("Introducir Origen: ");
	gets(origen);
	strcpy(destino,origen);
	printf("origen: %s\ndestino: %s\n\n",destino,origen);
	getch();
	memset(destino,'\0',10);
	memset(destino,'x',6);
	if(strlen(origen)<4)
	{
		printf("Se pueden concatenar\n");
		strcat(destino,origen);
	}
	else
		printf("No se pueden concatenar\n");
	printf("%s",destino);
	getch();
}
```

EJEMPLO: Pide una contraseña de entrada y la compara con "abcde"

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main(void)
```

```
{
        char contra[6]="abcde";
        char tu_contra[6];
        clrscr();
        printf("CONTRASEÑA ");
        gets(tu_contra);
        if(strcmp(tu_contra,contra)!=0)
                printf("ERROR");
        else
                printf("OK");
}
```

EJEMPLO: busca la primera coincidencia y muestra la cadena a partir de esa coincidencia.

```
#include<stdio.h>
#include<string.h>
void main(void)
{
        char letra;
        char *resp;
        char cad[30];
        clrscr();
        printf("Cadena: ");
        gets(cad);
        printf("Buscar Letra: ");
        letra=getchar();
        resp=strchr(cad,letra);
        if(resp)
                printf("%s",resp);
        else
                printf("No Esta");
        getch();
}
```

EJEMPLO: busca la última coincidencia y muestra la cadena a partir de ese punto.

```
#include<stdio.h>
#include<string.h>
void main(void)
{
        char letra;
        char *resp;
        char cad[30];
        clrscr();
        printf("Cadena: ");
        gets(cad);
        printf("Buscar Letra: ");
        letra=getchar();
        resp=strrchr(cad,letra);
        if(resp)
```

```
                printf("%s",resp);
        else
                printf("No Esta");
        getch();
}
```

EJEMPLO: En este ejemplo se busca en una cadena a partir de un grupo de caracteres. Si no encuentra coincidencia con ninguna letra en la cadena muestra un mensaje de error.

```
#include<stdio.h>
#include<string.h>
void main(void)
{
        char letras[5];
        char *resp;
        char cad[30];
        clrscr();
        printf("Introducir cadena: ");gets(cad);
        printf("Posibles letras(4): ");gets(letras);
        resp=strpbrk(cad,letras);
        if(resp)
                printf("%s",resp);
        else
                printf("Error");
        getch();
}
```

FUNCIONES MATEMÁTICAS

El estándar C define 22 funciones matemáticas que entran en las siguientes categorías, trigonométricas, hiperbólicas, logarítmicas, exponenciales y otras. Todas la funciones requieren el archivo de cabecera MATH.H. Si un argumento de una función matemática no se encuentra en el rango para el que esta definido, devolverá un valor definido EDOM.

ACOS: Devuelve un tipo double. Muestra el arcococoseno de la variable. La variable debe ser de tipo double y debe estar en el rango –1 y 1, en otro caso se produce un error de dominio. Cabecera <math.h>.
double acos(variable_double);

ASIN: Devuelve un tipo double. Muestra el arcoseno de la variable. La variable debe ser de tipo double y debe estar en el rango –1 y 1, en otro caso se produce un error de dominio. Cabecera <math.h>.
double asin(variable_double);

ATAN: Devuelve un tipo double. Muestra el arcotangente de la variable. La variable debe ser de tipo double y debe estar en el rango –1 y 1, en otro caso se produce un error de dominio. Cabecera <math.h>.
double atan(variable_double);

COS: Devuelve un tipo double. Muestra el coseno de la variable. La variable debe ser de tipo double y debe estar expresada en radianes. Cabecera <math.h>.
double cos(variable_double_radianes);

SIN: Devuelve un tipo double. Muestra el seno de la variable. La variable debe ser de tipo double y debe estar expresada en radianes. Cabecera <math.h>.
double sin(variable_double_radianes);

TAN: Devuelve un tipo double. Muestra la tangente de la variable. La variable debe ser de tipo double y debe estar expresada en radianes. Cabecera <math.h>.
double tan(variable_double_radianes);

COSH: Devuelve un tipo double. Muestra el coseno hiperbólico de la variable. La variable debe ser de tipo double y debe estar en el rango –1 y 1, en otro caso se produce un error de dominio. Cabecera debe ser <math.h>.
double cosh(variable_double);

SINH: Devuelve un tipo double. Muestra el seno hiperbólico de la variable. La variable debe ser de tipo double y debe estar en el rango –1 y 1, en otro caso se produce un error de dominio. Cabecera debe ser <math.h>.
double sinh(variable_double);

TANH: Devuelve un tipo double. Muestra la tangente hiperbólico de la variable. La variable debe ser de tipo double y debe estar en el rango –1 y 1, en otro caso se produce un error de dominio. Cabecera debe ser <math.h>.
double tanh(variable_double);
EJEMPLO:

```
#include<stdio.h>
#include<math.h>
void main(void)
{
        double radianes;
        clrscr();
        printf("Introducir radianes: ");
        scanf("%f",&radianes);
        printf("Coseno= %f\n",cos(radianes));
        printf("Seno= %f\n",sin(radianes));
        printf("Tangente= %f",tan(radianes));
        getch();
}
```

CEIL: Devuelve un double que representa el menor entero sin ser menor de la variable redondeada. Por ejemplo, dado 1.02 devuelve 2.0. Si asignamos –1.02 devuelve –1. En resumen redondea la variable a la alta. Cabecera <math.h>.
double ceil(variable_double);

FLOOR: Devuelve un double que representa el entero mayor que no sea mayor a la variable redondeada.

Por ejemplo dado 1.02 devuelve 1.0. Si asignamos –1.2 devuelve –2. En resumen redondea la variable a la baja. Cabecera <math.h>.
double floor(variable_double);

FABS: Devuelve un valor float o double. Devuelve el valor absoluto de una variable float. Se considera una función matemática, pero su cabecera es <stdlib.h>.
var_float fabs(variable_float);

LABS: Devuelve un valor long. Devuelve el valor absoluto de una variable long. Se considera una función matemática, pero su cabecera es <stdlib.h>.
var_long labs(variable_long);

ABS: Devuelve un valor entero. Devuelve el valor absoluto de una variable int. Se considera una función matemática, pero su cabecera es <stdlib.h>.
var_float abs(variable_float);

MODF: Devuelve un double. Descompone la variable en su parte entera y fraccionaria. La parte decimal es el valor que devuelve la función, su parte entera la guarda en el segundo termino de la función. La variables tienen que ser obligatoriamente de tipo double. Cabecera <math.h>.
var_double_deimal= modf(variable,var_parte_entera);

POW: Devuelve un double. Realiza la potencia de un número base elevado a un exponente que nosotros le indicamos. Se produce un error si la base es cero o el exponente es menor o igual a cero. La cabecera es <math.h>.
var_double=pow(base_double,exponente_double);

EJEMPLO: se ingresa un binario y se lo convierte a decimal.

```
#include<stdio.h>
#include<string.h>
#include<math.h>
void main(void)
{
        char bin[30];
        int i, j=0, deci=0, bit;
        printf("ingrese un binario entero: ");
        scanf("%s",bin);
        printf("\nbinario= %s", bin);
        for (i=strlen(bin)-1; i>=0; i--)       // i toma como valor inicial la longitud del string
        {
                bit = bin[i] - '0';          // toma un bit de la posicion i, y lo convierte a entero
                deci=deci + bit * pow(2, j);   // conversion de binario a decimal
                j++;                                //  j es el exponente de la potenciacion

        };
      printf("\ndecimal= %d", deci);
}
```

SQRT: Devuelve un double. Realiza la raíz cuadrada de la variable. La variable no puede ser negativa, si lo es se produce un error de dominio. La cabecera <math.h>.
var_double=sqrt(variable_double);

EJEMPLO:

```
#include<stdio.h>
#include<math.h>
void main(void)
{
        float num;
        double num_dec,num_ent;
        clrscr();
        printf("Introducir Numero: ");
        scanf("%f",&num);
        gotoxy(9,3);printf("ALTO: %.1f",ceil(num));
        gotoxy(1,4);printf("REDONDEO");
        gotoxy(9,5);printf("BAJO: %.1f",floor(num));
        num_dec=modf(num,&num_ent);
        gotoxy(12,8);printf("ENTERA: %.2f",num_ent);
        gotoxy(1,9);printf("DESCONPONGO");
        gotoxy(12,10);printf("DECIMAL: %.2f",num_dec);
        gotoxy(1,13);
        printf("VALOR ABSOLUTO: %.2f",fabs(num));
        gotoxy(1,16);
        printf("R.CUADRADA: %.2f",sqrt(fabs(num)));
        getch();
}
```

LOG: Devuelve un double. Realiza el logaritmo natural (neperiano) de la variable. Se produce un error de dominio si la variable es negativa y un error de rango si es cero. Cabecera <math.h>.
double log(variable_double);

LOG10: Devuelve un valor double. Realiza el logaritmo decimal de la variable de tipo double. Se produce un error de dominio si la variable es negativo y un error de rango si el valor es cero. La cabecera que utiliza es <math.h>.
double log10(var_double);

RAMDOMIZE(): Inicializa la semilla para generar números aleatorios. Utiliza las funciones de tiempo para crear esa semilla. Esta función esta relacionada con random. Cabecera <stdlib.h>.
void randomize();

RANDOM: Devuelve un entero. Genera un número entre 0 y la variable menos uno. Utiliza el reloj del ordenador para ir generando esos valores. El programa debe llevar la función randomize para cambiar la semilla en cada ejecución. Cabecera <stdlib.h>.
int random(variable_int);

EJEMPLO: Genera seis números aleatorios entre 1 y 49. No se repite ninguno de ellos.

```
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
	int num=0,num1=0,repe,temp;
	int valores[6];
	clrscr();
	printf("Loteria primitiva: ");
	randomize();
	for(;;)
	{
		repe=1;
		if(num==6)
			break;
		temp=random(49)+1;
		for(num1=0;num1<=num;num1++)
		{
			if(valores[num1]==temp)
			{
				valores[num1]=temp;
				num--;
				repe=0;
				break;
			}
		}
		if (repe==1)
			valores[num]=temp;
			num++;
		}
	for (num=0;num<6;num++)
		printf("%d ",valores[num]);
	getch();
}
```

FUNCIONES DE CONVERSIÓN

En el estándar de C se definen funciones para realizar conversiones entre valores numéricos y cadenas de caracteres. La cabecera de todas estas funciones es STDLIB.H. Se pueden dividir en dos grupos, conversión de valores numéricos a cadena y conversión de cadena a valores numéricos.

ATOI: Devuelve un entero. Esta función convierte la cadena en un valor entero. La cadena debe contener un número entero válido, si no es así el valor devuelto queda indefinido. La cadena puede terminar con espacios en blanco, signos de puntuación y otros que no sean dígitos, la función los ignora. La cabecera es <stdlib.h>.
int atoi(variable_char);

ATOL: Devuelve un long. Esta función convierte la cadena en un valor long. La cadena debe contener un número long válido, si no es así el valor devuelto queda indefinido. La cadena puede terminar con espacios en blanco, signos de puntuación y otros que no sean dígitos, la función los ignora. La cabecera es <stdlib.h>.
long atol(variable_char);

ATOF: Devuelve un double. Esta función convierte la cadena en un valor double. La cadena debe contener un número double válido, si no es así el valor devuelto queda indefinido. La cadena puede terminar con espacios en blanco, signos de puntuación y otros que no sean dígitos, la función los ignora. La cabecera es <stdlib.h>.
double atof(variable_char);

SPRINTF: Devuelve una cadena. Esta función convierte cualquier tipo numérico a cadena. Para convertir de número a cadena hay que indicar el tipo de variable numérica y tener presente que la longitud de la cadena debe poder guardar la totalidad del número. Admite también los formatos de salida, es decir, que se puede coger distintas partes del número. La cabecera es <stdlib.h>.
sprintf(var_cadena,"identificador",var_numerica);

ITOA: Devuelve una cadena. La función convierte un entero en su cadena equivalente y sitúa el resultado en la cadena definida en segundo termino de la función. Hay que asegurarse que la cadena se lo suficientemente grande para guardar el número. Cabecera <stdlib.h>.
itoa(var_entero,var_cadena,base);

BASE	DESCRIPCIÓN
2	Convierte el valor en binario.
8	Convierte el valor a Octal.
10	Convierte el valor a decimal.
16	Convierte el valor a hexadecimal.

LTOA: Devuelve una cadena. La función convierte un long en su cadena equivalente y sitúa el resultado en la cadena definida en segundo termino de la función. Hay que asegurarse que la cadena se lo suficientemente grande para guardar el número. Cabecera <stdlib.h>.
ltoa(var_long,var_cadena,base);

EJEMPLO:

```
#include<stdio.h>
#include<stdlib.h>
void main(void)
{
        char texto[4];
        char ntext[10],ntext1[10];
        int num;
        float total;
        clrscr();
        printf("Numero de 3 digitos: ");
        scanf("%d",&num);
```

```
        fflush(stdin);
        printf("Cadena numerica: ");
        gets(ntext);
        fflush(stdin);
        printf("Cadena numerica: ");
        gets(ntext1);
        fflush(stdin);
        sprintf(texto,"%d",num);
        printf("%c %c %c\n",texto[0],texto[1],texto[2]);
        total=atof(ntext)+atof(ntext1);
        printf("%.3f",total);
        getch();
}
```

FUNCIONES DE FECHA Y HORA

Estas funciones utilizan la información de hora y fecha del sistema operativo. Se definen varias funciones para manejar la fecha y la hora del sistema, así como los tiempos transcurridos. Estas funciones requieren el archivo de cabecera TIME.H. En este archivo de cabecera se definen cuatro tipos de estructuras para manejar las funciones de fecha y hora (size_t , clock_t , time_t , time).

TIME: Devuelve la hora actual del calendario del sistema. Si se produce un error devuelve –1. Utiliza la estructura time_t a la cual debemos asignar una etiqueta que nos servirá para trabajar con la fecha y hora. Por si sola no hace nada, necesita otras funciones para mostrar los datos. Cabecera <time.h>.

```
time_t nombre_etiqueta;
.
.
nombre_etiqueta=time(NULL);
```

CTIME: Devuelve un puntero a una cadena con un formato día semana mes hora:minutos:segundo año \n\0. La hora del sistema se obtiene mediante la función time. Cabecera <time.h>.

```
puntero=ctime(&etiqueta_estructura_time_t);
```

EJEMPLO:

```
#include<stdio.h>
#include<time.h>
void main(void)
{
        time_t fecha_hora;
        clrscr();
        fecha_hora=time(NULL);
        printf(ctime(&fecha_hora));
        getch();
}
```

GETTIME: Devuelve la hora del sistema. Utiliza la estructura dostime_t para guardar la información referente a la hora. Antes de hacer referencia a la función hay que crear una etiqueta de la estructura. Cabecera <dos.h>.

```
_dos_gettime(&etiqueta_estructura_dostime_t);
struct dostime_t
{
        unsigned hour;
        unsigned minute;
        unsigned second;
        unsigned hsecond;
}
```

EJEMPLO:

```
#include<stdio.h>
#include<dos.h>
void main(void)
{
        struct dostime_t ho;
        clrscr();
        _dos_gettime(&ho);
        printf(" %d:%d:%d",ho.hour,ho.minute,ho.second);
        getch();
}
```

GETDATE: Devuelve la fecha del sistema. Utiliza la estructura dosdate_t para guardar la información referente a la fecha. Antes de hacer referencia a la función hay que crear una etiqueta de la estructura. Cabecera <dos.h>.

```
_dos_getdate(&etiqueta_estructura_dosdate_t);
struct dosdate_t
{
        unsigned day;
        unsigned month;
        unsigned year;
        unsigned dayofweek;
}
```

EJEMPLO:

```
#include<stdio.h>
#include<dos.h>
void main(void)
{
        struct dosdate_t fec;
        clrscr();
        _dos_getdate(&fec);
        printf("%d/%d/%d\n",fec.day,fec.month,fec.year);
        getch();
}
```

SETTIME: Permite cambiar la hora del sistema. Utiliza la estructura dostime_t para guardar la información referente a la hora. Antes de hacer referencia a la función hay que crear una etiqueta de la estructura. Cabecera <dos.h>.
_dos_settime(&etiqueta_estructura_dostime_t);

SETDATE: Permite cambiar la fecha del sistema. Utiliza la estructura dosdate_t para guardar la información referente a la fecha. Antes de hacer referencia a la función hay que crear una etiqueta de la estructura. Cabecera <dos.h>.
_dos_setdate(&etiqueta_estructura_dosdate_t);

EJEMPLO:

```
#include<stdio.h>
#include<dos.h>
void main(void)
{
        struct dosdate_t fec;
        struct dostime_t ho;
        clrscr();
        printf("Introducir fecha: ");
        gotoxy(19,1);
        scanf("%u%*c%u%*c%u",&fec.day,&fec.month,&fec.year);
        printf("Introducir hora: ");
        gotoxy(18,2);
        scanf("%u%*c%u%*c%u",&ho.hour,&ho.minute,&ho.second);
        _dos_settime(&ho);
        _dos_setdate(&fec);
}
```

DIFFTIME: Devuelve un double. La función devuelve la diferencia, en segundos, entre una hora inicial y hora final. Se restara la hora final menos la hora inicial. Tenemos que obtener la hora al iniciar un proceso y al terminar el proceso volveremos a tomar la hora. Cabecera <time.h>.
double difftime(hora_final,hora_inicial);

EJEMPLO:

```
#include<stdio.h>
#include<time.h>
void main(void)
{
        time_t inicio, final;
        double tiempo,cont;
        clrscr();
        inicio=time(NULL);
        for(cont=0;cont<50000;cont++)
                clrscr();
        final=time(NULL);
        tiempo=difftime(final,inicio);
        printf("Tiempo: %.1f",tiempo);
        getch();
}
```