**Lenguaje C**

El [lenguaje C](http://www.monografias.com/trabajos/introc/introc.shtml), fue diseñado por Dennies Ritchie en 1970, en los laboratorios Bell de Estados Unidos.

Este lenguaje presenta varias características, entre las cuales están:

1. Lenguaje de programación de propósitos generales
2. Permite la Programación Estructurada
3. Abundancia de Operadores y Tipos de Datos
4. No está asociado a ningún [sistema](http://www.monografias.com/trabajos11/teosis/teosis.shtml) operativo ni a ninguna máquina
5. Popular y Eficaz
6. Permite el [desarrollo](http://www.monografias.com/trabajos12/desorgan/desorgan.shtml) de Sistemas Operativos y programas de aplicación
7. Portabilidad
8. Existen las librerías en las bibliotecas
9. tiene sólo 32 palabras reservadas



\*[bibliotecas](http://www.monografias.com/trabajos10/ponency/ponency.shtml): es el [archivo](http://www.monografias.com/trabajos7/arch/arch.shtml) que contiene [código](http://www.monografias.com/trabajos12/eticaplic/eticaplic.shtml) objeto de una colección de rutinas o funciones que realizan tareas determinadas y se pueden utilizar en los programas.

\*Enlazador: Programa que convierte las funciones de la [biblioteca](http://www.monografias.com/trabajos10/ponency/ponency.shtml) estándar de C, con el código que ha traducido el compilador .

**Estructura de Un programa en C**

Ya estamos apunto de entrar a lo más interesante, a la programación en sí; pero es necesario, primero; mencionar algunos de los errores típicos al programar, para que el lector sepa como identificarlos y así los pueda corregir.

1. **ERROR DE SINTAXIS:** Estos errores son producidos, cuando se hace mal uso de las reglas del lenguaje de programación, y se violan las [normas](http://www.monografias.com/trabajos4/leyes/leyes.shtml) de sintaxis, de ese lenguaje (en nuestro caso C); estos errores son fáciles de detectar por que generalmente es el compilador, que los identifica (Y hasta [muestra](http://www.monografias.com/trabajos11/tebas/tebas.shtml) la línea donde se encuentra dicho error, pero eso depende de la versión del compilador que estemos usando). En este curso he usado Turbo C, en su versión 2 y 3.
2. **ERRORES DE EJECUCIÓN:** Estos errores se producen , cuando le indicamos a la computadora, realizar una determinada acción, y esta la comprende, pero no puede ejecutarla. Por ejemplo, indicarle a la computadora una división entre cero, sumar dos [variables](http://www.monografias.com/trabajos12/guiainf/guiainf.shtml#HIPOTES) a las cuales no se les ha signado [valor](http://www.monografias.com/trabajos14/nuevmicro/nuevmicro.shtml) alguno, etc.
3. **ERRORES DE LÓGICA:** Muchas veces, cuando estamos programando, el compilador no nos indica errores de sintaxis, ni de lógica; pero el resultado de nuestro programa, esta fuera del rango esperado, esto es [producto](http://www.monografias.com/trabajos12/elproduc/elproduc.shtml) de un error de lógica en el código de nuestro programa. Este tipo de errores son muy difíciles de identificar y por supuesto de corregir, ya que generalmente hay que revisar línea por línea de nuestro programa. Ejemplo: El sueldo negativo de un empleado, etc.

La [estructura](http://www.monografias.com/trabajos15/todorov/todorov.shtml#INTRO) de un programa en C, consta de algunas partes esenciales: las cuales son uno o más módulos llamadas funciones, siendo main() la primera función que es llamada cuando empieza la ejecución del programa .

Cada función debe contener :

>Directivas de pre-[procesador](http://www.monografias.com/trabajos5/sisope/sisope.shtml) (instrucciones que se le dan al compilador

#include antes de compilar)

#define

ejemplo:

#include <stdio.h>

Lo que se le esta indicando, es que de las librerías, "Incluya" en nuestro programa la directiva stdio.h, la cual contiene las funciones de entrada y salida de datos (standar input output, en [inglés](http://www.monografias.com/trabajos16/manual-ingles/manual-ingles.shtml)). Si necesitamos las funciones [matemáticas](http://www.monografias.com/Matematicas/index.shtml), debemos especificarlo con la declaratoria:

#include **<**math.h**>**

Si necesitamos las funciones de cadenas:

#inlcude **<**stlib.h**>**

Es necesario aclarar que esto se hace al inicio del programa, y las declaratorias deben llevar el símbolo de numeral (#) seguido de la sentencia "include", y entre [signos](http://www.monografias.com/trabajos36/signos-simbolos/signos-simbolos.shtml) de mayor y menor que (<>) el nombre de la directiva.

>Declaraciones Globales

pueden ser:

\*Prototipos de Funciones: También llamadas declaraciones de funciones, lo cual se tratará más adelante

\*Declaraciones de Variables

cabe destacar, que esto se hace seguido de los #include y los #define.

>Función Principal main()

Esta es la función principal de nuestro programa, su cuerpo, por ello NUNCA debe faltar, ya que en ella van contenidas todas las instrucciones de nuestro programa.

main()

{

**declaraciones locales /**\*Comentarios \*/

**sentencias**

}

la función main() va al inicio, luego abrimos llaves y dentro de ellas van las declaraciones de variables, las sentencias de [lectura](http://www.monografias.com/trabajos14/textos-escrit/textos-escrit.shtml), cálculos, asignaciones e impresiones, y con la última llave ( } ), le indicamos el final del programa.

**Ejemplo 1.1**

Programa que a partir del [radio](http://www.monografias.com/trabajos13/radio/radio.shtml), calcula el área de un circulo

#include <stdio.h>

#include <conio.h>

main()

{

float radio, area;

printf("Radio=\n");

scanf("%f", &radio);

area=3.14159\*radio\*radio;

printf("El Area es %f\n\n", area);

getch();

return 0;

}

**Explicación:**

Le indicamos al compilador, que usaremos las bibliotecas <stdio.h> y <conio.h>, ¿por qué <conio.h>?, por que esta biblioteca, contiene las funciones getche(), getch(), etc, y de una de ellas hacemos uso en este pequeño ejemplo.

Luego, le indicamos a nuestro programa el incio de nuestro programa (función main() ).

Declaramos, como [valores](http://www.monografias.com/trabajos14/nuevmicro/nuevmicro.shtml) reales, las variables radio y area (de esto se hablará más adelante). Luego, con la instrucción printf(), mostramos en pantalla el mensaje (Radio=) y scanf se encarga de leer el valor digitado por el usuario. Posteriormente area, es igual al la multiplicación de pi (3.14159), el radio al cuadrado. Se muestra en pantalla ese resultado, luego el programa espera que se presiones cualquier tecla (getch() ) y no retorna ningún valor (return 0).

**Sintaxis de Algunos Elementos de Un Programa en C**

1. como su nombre lo indica, estos son los nombres, con los que identificamos las variables, constantes, funciones, [vectores](http://www.monografias.com/trabajos35/vectores/vectores.shtml), etc, de nuestro programa. Para ello debemos tener presente algunas reglas:

>pueden tener de 1 hasta un máximo de 31 caracteres

>Debe de iniciar con una letra o subrayado

Ejemplo:

(Correctos)

c2

\_c2

( Incorrectos)

2c

2 c

>No es lo mismo una minúscula que una mayúscula, ya que c distingue de entre ellas. Ejemplo: BETA ¹ Beta ¹ beta ¹ BeTa

>No son válidos los identificadores de palabras reservadas. En un inicio hablamos que c posee 32 palabras reservadas, entre ellas están:

float char while

int else return

Estas palabras no pueden ser utilizadas para identificar variables, constantes, funciones etc

1. **identificadores:**

En todo programa que estemos diseñando en C (o en cualquier otro lenguaje de programación); es necesario insertar ciertos comentarios en el código, para que en posteriores modificaciones y cuando se realice el mantenimiento, podamos recordar cosas importantes ya que, en los comentarios, podemos incluir aspectos importantes del programas, explicaciones del funcionamiento de las sentencias, etc.

El formato de los comentarios en C, es el siguiente:

/\*este es un comentario en C \*/

/\*Podemos colocar mucha información importante

de nuestro Programa \*/

1. **Comentarios**

Permite que, el pre-procesador, incluya funciones proporcionadas por el fabricante, a nuestro programa. Ejemplo:

#include <stdio.h> /\* le decimos al compilador que incluya la librería

stdio.h \*/

1. **La Directiva #include**

permite definir constantes simbólicas. Pero hasta ahora ha sido poco lo que hemos hablado acerca de las constantes, es por ello que en aprovechando, este especio; dedicaré unas cuantas líneas para aclarar ello.

Las variables pueden cambiar de valor, durante la ejecución del programa, por eso es que se llaman *variables.* Y las constantes como su nombre lo indica, son valores que permanecen constantes durante toda la ejecución del programa, un ejemplo de ello, es el valor de p (pi) que equivale a 3.14159....

En C existen diferentes tipos de variables, entre ellas tenemos:

**1. Constates Numéricas:**

Son valores numéricos, enteros o de reales (de punto flotante). Se permiten también constantes octales y hexadecimales.

**2. Constantes Simbólicas:**

las constantes simbólicas tiene un nombre (identificador), y en esto se parecen las variables. Sin embargo, no pueden cambiar de valor a lo largo de la ejecución del programa. En C, se pueden definir mediante el preprocesador.

(Tomado del Manual *"Aprenda Lenguaje ANSI C como si estuviera en Primero"* [Escuela](http://www.monografias.com/trabajos13/artcomu/artcomu.shtml) superior de Ingenieros Industriales. Universidad de Navarra. Febrero de 1998).

**Ejemplo:**

#define N 100

#define PI 3.1416

#define B 45

Esta directiva (#define) va, inmediatamente después de los #include. Se escribe la directiva, se deja un espacio y se escribe el identificador de la constante, otro espacio y su valor.

1. **la directiva #define**

/ ! % ^ & \* () - + {} [] \ ; : <> ¿ .

1. **Signos de Puntuación y de Separación**

Al momento de programar en C, esta es una regla de [oro](http://www.monografias.com/trabajos35/oferta-demanda-oro/oferta-demanda-oro.shtml), y la causa por la cual nuestro programa puede darnos muchos errores de sintaxis, cuando se omite, al final de cada sentencia un punto y coma (;). Ya que con ello le indicamos al compilador que ha finalizado una sentencia.

NOTA: el lector no debe confundirse, las directivas: #include, #define. Main(), no llevan punto y coma, por que no son sentencias.

Recordemos el ejemplo 1.1, y vea que al final de cada sentencia lleva su correspondiente punto y coma:

#include <stdio.h>

#include <conio.h>

main()

{

float radio, area;

printf("Radio=\n");

scanf("%f", &radio);

area=3.14159\*radio\*radio;

printf("El Area es %f\n\n", area);

getch();

return 0;

}

1. **Todas las Instrucciones o sentencias del programa terminan con un punto y coma (;)**

Esta consideración toma mayor auge, cuando veamos las instrucciones anidadas en condiciones, ciclos, etc.

**Ejemplo:**

{

...

printf("Hola\n\b");

...

}

1. **Todo Bloque de Instrucciones debe ir entre llaves**
2. **En una línea se pueden escribir más de una instrucción separada por un punto y coma**

Esto es posibles, por que con el punto y coma, le estamos indicando al compilador el fin de una sentencia o instrucción.

Ejemplo:

b = c + d; d = 2\*k;

**Tipos de Datos en C**

Un tipo de dato, se define como *un conjunto de valores que puede tener una variables, junto con ciertas*[*operaciones*](http://www.monografias.com/trabajos6/diop/diop.shtml)*que se pueden realizar con ellas.*

\*[TIPOS DE DATOS](http://www.monografias.com/trabajos11/basdat/basdat.shtml) PREDEFINIDOS

|  |
| --- |
| TABLA CON LOS TIPOS DE DATOS PREDEFINIDOS EN C |
| >ENTEROS: numeros completos y sus negativos |
| **Palabra reservada:** | **Ejemplo** | **Tamaño (byte)** | **Rango de valores** |
| int | -850 | 2 | -32767 a 32767 |
| VARIANTES DE ENTEROS |
| short int | -10 | 1 | -128 a 127 |
| unsigned int | 45689 | 2 | 0 a 65535 |
| long int | 588458 | 4 | -2147483648 a 2147483647 |
| unsigned long | 20000 | 4 | 0 a 4294967295 |
| >REALES: números con decimales o punto flotante |
| **Palabra reservada:** | **Ejemplo** | **Tamaño (byte)** | **Rango de valores** |
| float | 85 | 4 | 3.4x10-38 a 3.4x1038 |
| VARIANTES DE LOS REALES |
| double | 0.0058 | 8 | 1.7x10-308 a 1.7x10308 |
| long double | 1.00E-07 | 10 | 3.4x10-4932 a 1.1x104932 |
| >[CARÁCTER](http://www.monografias.com/trabajos34/el-caracter/el-caracter.shtml): letras, digitos, [símbolos](http://www.monografias.com/trabajos36/signos-simbolos/signos-simbolos.shtml), signos de puntuación. |
| **Palabra reservada:** | **Ejemplo** | **Tamaño (byte)** | **Rango de valores** |
| char | 'O' | 1 | 0 ......255 |

**Declaración de Variables**

Una Variable, como su nombre lo indica, es capaz de almacenar diferentes valores durante la ejecución del programa, su valor *varía.* Es un lugar en [la memoria](http://www.monografias.com/trabajos16/memorias/memorias.shtml) el cual, posee un nombre (identificador), y un valor asociado.

La declaración de variables en C, se hace en minúsculas.

Formato:

***Tipo\_de\_dato nombre\_de\_la\_variable;***

**Ejemplos:**

\*Declare una variable de tipo entero y otra de tipo real, una con el nombre de "x" y otra con el identificador "y":

int x;

float y;

\*Declare una variable de tipo entero llamada moon, e inicialícela con un valor de 20

int x = 20;

\*Declare una variable de tipo real, llamada Pi, e inicialícela con una valor de 3.1415

float pi=3.1415;

\*Declare una variable de tipo caracrter y asígnele el valor de "M"

char car = ’M’;

\*Declare una variable llamada nombre, que contenga su nombre:

char nombre[7]="Manuel";

**Explicación:**

En el apartado anterior, se explicó, que C, no tiene el tipo de dato llamado string, o mejor conocido como cadenas de texto, pero nosotros podemos hacer uso de ellas, por medio de un *arreglo,* (de lo cual hablaremos con más detalle, posteriormente); pero para declarar este tipo de datos colocamos el tipo de datos, es decir la palabra reservada *char* luego el nombre, e inmediatamente abrimos, entre corchetes, va el número de letras, que contendrá dicha variable. Es muy importante que al momento de declarar el tamaño, sea un número mayor, al verdadero número de letras; por ejemplo, la palabra "Manuel", solo tiene 6 letras, pero debemos declararlo para 7 letras ¿Por qué?.

Veámoslo gráficamente, en la memoria, C crea un variable llammada nombre y esta posee la palabra Manuel, así:



en realidad, hay 7 espacios, pero la cuanta llega hasta 6, por que c, toma la primera posición como la posición cero, y para indicar el final de la cadena lo hace con un espacio en blanco.

**Declaración de Constantes**

Las constantes, como su nombre lo indica, *son valores que se mantiene invariables durante la ejecución del programa.*

Su formato es el siguiente:

**const** tipo\_de\_dato nombre= valor;

donde const, es una palabra reservada, para indicarle al compilador que se esta declarando una constante**.**

**Ejemplo:**

const int dia=7;

const float pi=3.14159;

const char caracter= ‘m’;

const char fecha[]="25 de diciembre";

**Caso Especial Constantes Simbólicas**

Las constantes simbólicas, se declaran mediante la directiva #define, como se explicó anteriormente. Funcionan de la siguiente manera, cuando C, encuentra el símbolo que representa a la constante, lo sustituye por su respectivo valor.

Ejemplo:

#define N 150

#define PI 3.1416

#define P 50

**Entrada y Salida Por Consola**

Entrada y Salida por consola: se refiere a las operaciones que se producen en el [teclado](http://www.monografias.com/trabajos5/sisope/sisope2.shtml#tecla) y en la pantalla de la computadora. En C no hay palabras claves para realizar las [acciones](http://www.monografias.com/trabajos4/acciones/acciones.shtml) de Entrada/Salida, estas se hacen mediante el uso de las funciones de la biblioteca estándar (stadio.h).

Para utilizar las funciones de E / S debemos incluir en el programa el archivo de cabecera stdio.h, mediante la declaratoria:

#include <stdio.h>

Las Funciones de E / S más simples son getchar() que lee un carácter del teclado, espera un retorno de carro (¿ ), es decir un *enter* y el eco aparece. Es decir la tecla presionada.

**\*putchar():** Imprime un carácter en la pantalla, en la posición actual del cursor.

Algunas variaciones:

**\*getche():** Aparece el Eco

**\*getch():** No aparece el eco

estas instrucciones se encuentran en la biblioteca *conio.h*

Veamos un ejemplo:

Programa que espera que se presiona una tecla, la muestra en pantalla, y además muestra el carácter siguiente:

Ejemplo 2.1:

#include <stdio.h>

#include <conio.h>

main()

{

char car;

clrscr(); /\*Se encarga de borrar la pantalla por eso se llama claer screen\*/

car=getchar();

putchar(car+1);

getch();

return 0;

}

Ejemplo 2.2:

#include <stdio.h>

#include <conio.h>

main()

{

char x; /\*Declaramos x como caracter\*/

printf("Para Finalizar Presione cualquier Tecla:");

x= getchar();/\*Captura y muestra el caracter presionado\*/

getch();/\*Espera a que se presione cualquier otra tecla para finalizar\*/

return 0;

}

**Entrada / Salida de Cadenas**

Una Cadena, es una frase, compuesta por varias palabras. En C, podemos hacer uso de las cadenas, mediante, la sentencia:

**\*gets():** Lee una cadena de carácter introducido por el teclado. Se puede introducir caracteres hasta que se de un retorno de carro, (enter); el cual no es parte de la cadena; en su lugar se coloca un terminador nulo \0.

**\*puts():** Imprime en pantalla, el argumento guardado en la variable que se manda a impresión.

Ejemplo 2.3

Diseñe un programa en C, que lea su nombre; lo salude y mande a impresión su nombre, usando gets e y puts

#include <stdio.h>

#include <conio.h>

main()

{

char nombre[40];

puts("digite su nombre:");

gets(nombre);

puts("BIENVENIDO:");

puts(nombre);

getch();

return 0;

}

NOTA: No haré mucho énfasis en estas instrucciones, ya que más adelante, veremos las instrucciones scanf() y printf(), que son mucho más completas.

**Entrada / Salida Por Consola con Formato**

Las funciones gets, puts, getch, etc; son utilizadas, en una forma un poco rudimentaria, sin embargo; C posee otra serie de funciones, que son más completas, las cuales nos permiten leer e imprimir (en pantalla), datos con un formato determinado, el cual ha sido definido por el programador.

**Salida Hacia Pantalla [printf()]**

Se utiliza para imprimir en pantalla cadenas de texto solas, o mandar a pantalla el valor de alguna variable, o constante, o una combinación de las anteriores. Su formato es el siguiente:

**Printf(**"cadena de control", nombre\_de\_variables**);**

En donde:

Cadena de control: contiene códigos de formato que se asocian con los tipos de datos contenidos en las variables.

|  |  |
| --- | --- |
| **Código** | **Formato** |
| %d | Un entero |
| %i | Un entero |
| %c | Una caracter |
| %s | Una cadena |
| %f | Un real |
| %ld | Entero largo |
| %u | Decimal sin signo |
| %lf | Doble posición |
| %h | Entero corto |
| %o | Octal |
| %x | Hexadecimal |
| %e | Notación Científica |
| %p | Puntero |
| %% | Imprime Porcentaje |

TABLA 2.2

Ejemplo:

Int suma=10;

Printf("La suma es %d", suma);

**Explicación:**

Declaramos primero la variable como entero, con un valor de 10, luego la función printf, el mensaje va entre comillas dobles, luego en el lugar que queremos que aparezca el valor, colocamos el formato de la variable, cerramos comillas, luego una coma y el nombre de la variable. Es importante recalcar, que en la posición que coloquemos el formato es donde aparecerá el valor de la variable en este caso, 10.

Ejemplo:

Char nombre[7]="Manuel";

printf("%s es en creador de este manual", nombre);

NOTA: el número de argumentos que tendrá la función printf() es indefinido, por lo que se puede transmitir cuantos datos sean necesarios.

Ejemplo:

Int x=12, y=15;

char z=’D’;

float v=10.2563;

printf("Estos son números %d %d %f; y esta es una letra %c", x,y,v,z);

También podemos hacer algunos arreglos, al formato de salida, por ejemplo, si deseamos imprimir un número real justificado a la izquierda podemos colocar:

printf("%-f", z);

para justificar colocarle signo: %+f

%20f >> Longitud numérica del campo

%.2f >>Imprime el valor con sólo dos decimales

**Secuencias de Escapes**

Indica que debe ejecutar algo extraordinario.

|  |  |
| --- | --- |
| **Carácter de Escape** | **Explicación** |
| \n | Simula un Enter. Se utiliza para dejar una línea de por medio |
| \t | Tabulador horizontal. Mueve el cursor al próximo tabulador |
| \v | Tabulador vertical. |
| \a | Hace sonar la alarma del sistema |
| \\ | Imprime un carácter de diagonal invertida |
| \? | Imprime el carácter del signo de interrogación |
| \" | Imprime una doble comilla |

TABLA 2.3

Ejemplos:

1) printf("Manuel \n Antonio \n Ortez\n\n);

2) int x=15;

printf("El Valor de la variable es %d\n\n", x);

3) float x=8.5689, pi=3.1416;

printf("El valor de x es %.2f\t\n",x);

printf("\t Y el valor de pi es %.2f\n\n", pi);

**Entrada Desde Teclado**

Se realiza mediante la función scanf(), su formato es:

**scanf(**"Cadena de control", [Dirección](http://www.monografias.com/trabajos15/direccion/direccion.shtml) y nombre de la variable**);**

Ejemplo 2.4

Diseñe un programa que guarde y muestre la nota del examen final de 3 alumnos

#include <stdio.h>

#include <conio.h>

main()

{

float n1, n2, n3;

char nom1[10], nom2[10], nom3[10];

printf("Introduzca el Nombre del Primer alumno:\n");

scanf("%s", nom1);

printf("Introduzca la nota de este alumno:\n");

scanf("%f", &n1);

printf("Digite el nombre del segundo alumno:\n");

scanf("%s", nom2);

printf("Su nota es:\n");

scanf("%f", &n2);

printf("Finalmente el ultimo alumno es:\n");

scanf("%s", nom3);

printf("Y su nota es:\n");

scanf("%f", &n3);

getch();

return 0;

}

**Explicación:**

Primero, iniciamos con las directivas del preprocesador:

#include <stdio.h>

#include <conio.h>

Con la cual le indicamos al compilador, que de su librería añada a nuestro programa las funciones estándar de entrada y salida; así como las entradas y salidas por consola (stadio.h y conio.h, respectivamente).

Luego declaramos la variables, que contendrán las notas como reales (o de punto flotante:

float n1, n2, n3;

Ya que, las notas pueden ser deciamales, por ejemplo 9.6, 8.5; etc.

Luego declaramos las variables, que contendrán las notas, caba aclarar que al momento de las declaraciones las podemos hacer en el orden que deseemos, pueden ser primeros los tipo char y luego los float, o viceversa, pero teniendo el cuidado que las variables que contendrán las nombres lleven la longitud máxima entre corchetes, para nuestro caso, 10. ( [10] ).

Posteriormente, mostramos en pantalla, un mensaje con el cual le indicamos al usuario que introduzca los datos respectivos:

printf("Introduzca el Nombre del Primer alumno:\n");

A continuación, va la función scanf, primero y entre comillas el tipo de dato que va a leer:

scanf("%s", nom1);

como puede notarse, va a leer la cadena de texto que contendrá la variable nom1. cabe aclarar, que cuando se van a leer cadenas de texto, no es necesario colocar la dirección (&), lo cual no sucede con los otros tipos de datos:

scanf("%f", &n1);

Después de haber leído los datos, espera a que se presiones cualquier tecla para finalizar la ejecución del programa.

Ejemplo 2.5

Programa que imprime dos veces, la cadena de texto que se ha introducido:

#include <stdio.h>

#include <conio.h>

main()

{

char cadena[15];

printf("Digite la cadena:\n\n");

scanf("%s", cadena);

printf("\n\t LA CADENA ES LA SIGUIENTE:\n\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("%s\n", cadena);

printf("%s\n", cadena);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getch();

return 0;

}

Es importante, que el lector, intente correr, en su máquina estos ejemplos, para que comprenda con mayor facilidad.