

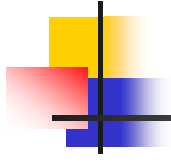
Ejemplos de programas en C

Sistemas Operativos (prácticas)
E. U. Informática en Segovia
Universidad de Valladolid



El lenguaje de Programación C

- Lenguaje de alto nivel de propósito general.
 - Sintaxis sumamente compacta
 - Alta portabilidad (independencia del hardware subyacente)
 - Muy buenas facilidades para acceso a bajo nivel
 - Gran parte de la funcionalidad se implementa en bibliotecas externas
 - E/S
 - Gestión de tiras de caracteres
 - Asignación de memoria
 - ...

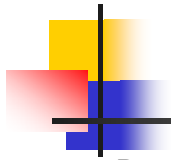


Ejemplo 1: Estructura de un programa

- Inclusión de bibliotecas, declaraciones de variables y tipos, y secuencia de funciones
- Una y sólo una función denominada **main** (programa principal)
- Función:
tipoRetorno Nombre (parametros) {sentencias}

```
/* ejemplo 1.- Escribe un mensaje en pantalla */

# include <stdio.h> /* incluye biblioteca donde se define E/S */
int main( )
{
    /*Este comentario es ignorado por el compilador y*/
    /*no genera código */
    printf("Introducción a la programación en lenguaje c\n");
    return 0;
}
```



Ejemplo 2: Definición de variables

- Declaración variable:
tipo Nombre [=valor];
- Asignación:
variable = expresión;

```
/* ejemplo 2.- multiplica dos números enteros y muestra el
resultado por pantalla */
```

```
#include <stdio.h>
int main( )
{
    int multiplicador; /*se define multiplicador como un entero */
    int multiplicando; /*se define multiplicando como un entero */
    int res; /*se define resultado como un entero*/
    multiplicador = 1000; /*se asignan valores*/
    multiplicando=2;
    res=multiplicador*multiplicando;
    printf("Resultado =%d\n",res); /*se muestra el resultado */
    return 0;
}
```



Ejemplo 3: Definición de variables

```
/* ejemplo 3.- .- multiplica dos números enteros y muestra el
   resultado (utiliza definición múltiple de variables) */

#include <stdio.h>

int main( ) {
    int multiplicador, multiplicando; /*se definen 2 variables*/

    multiplicador =1000; /*se les asigna valor*/
    multiplicando=2;
    printf("Resultado = %d\n", multiplicador*multiplicando);
    /*se muestra el resultado por pantalla*/
    return 0;
}
```

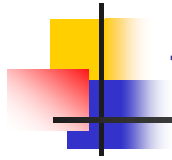


Tipos de variables

■ Variables de tipo entero

TIPO	BYTES	VALOR MINIMO	VALOR MAXIMO
signed char	1	-128	127
unsigned char	1	0	255
unsigned short	2	-32.768	+32.767
unsigned short	2	0	+65.535
signed int	2	-32.768	+32.767
unsigned int	2	0	+65.535
signed long	4	-2.147.483.648	+2.147.483.647
unsigned long	4	0	+4.294.967.295

- Nota: Si se omite el clasificador delante de la variable de tipo entero por defecto se considera “**signed**”



Tipos de variables

- Variables de tipo real

TIPO	BYTES	VALOR MINIMO	VALOR MAXIMO
float	4	3.4E-38	3.4E+38
double	8	1.7E-308	1.7E+308
long double	10	3.4E-4932	3.4E+4932

- Nota: Las variables de punto flotante son siempre con signo



Ejemplo 4: Conversiones de tipo

```
/* ejemplo 4.- se realizan conversiones de tipos implícitas y
explícitas */

#include <stdio.h>
int main() {
    double d , e , f = 2.33 ;
    int i = 6 ;

    e = f * i ; /* e es un double de valor 13.98*/
    printf( "Resultado = %f\n", e);
    d = (int) ( f * i ) ; /* d es un double de valor 13.00*/
    printf( "Resultado = %f\n", d);
    d = (int) f * i ; /* f se ha convertido a un entero truncando*/
    /*sus decimales, d es un double de valor 13.00*/
    printf( "Resultado = %f\n", d);
    return 0;
}
```



Ejemplo 5: Variable tipo carácter

```
/* ejemplo 5.- Distintas formas de asignar un carácter a una
variable de tipo char */
```

```
int main () {
    char c;

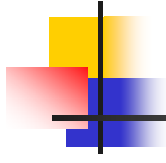
    c=97; /* el valor en decimal del código ASCII*/
    c='a'; /* el carácter entre comillas*/
    c=0x61; /* el valor en hexadecimal del código ASCII*/
    c=0141; /* el valor en octal del código ASCII*/
    return 0;
}
```



Secuencias de escape

- Existe una serie de caracteres no imprimibles que el editor los toma como órdenes, por lo que la manera de acceder a ellos es mediante una secuencia de escape

CODIGO	SIGNIFICADO	VALOR ASCII (decimal)	VALOR ASCII (hexadecimal)
'\n'	nueva línea	10	0x0A
'\r'	retorno de carro	13	0x0D
'\f'	nueva página	12	0x0C
'\t'	tabulador horizontal	9	0x09
'\b'	retroceso (<i>backspace</i>)	8	0x08
'\"'	comilla simple	39	0x27
'\''	comillas	34	0x22
'\\'	barra hacia atrás (<i>backslash</i>)	92	0x5C
'\? '	Interrogación	63	0x3F
'\nnn'	cualquier carácter (donde nnn es el código ASCII expresado en octal)		
'\xnn'	cualquier carácter (nn es el código ASCII expresado en hexadecimal)		



Ejemplo 6: Tamaño de las variables

```
/* ejemplo 6.- Uso del operador sizeof para determinar el tamaño de
una variable */
#include <stdio.h>

int main () {
    char c;
    int n, d;

    n= sizeof(c);
    printf(" el número de bytes de la variable c es %d\n", n);
    n= sizeof(d);
    printf(" el número de bytes de la variable d es %d\n", n);
    n= sizeof(int);
    printf(" el número de bytes que ocupa el tipo entero es %d\n",
n);
    printf(" el número de bytes que ocupa el tipo double es %d\n",
sizeof(double));
    return 0;
}
```



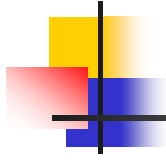
Ejemplo 7: Constantes y E/S simple

```
/* ejemplo 7.- Calcula el perímetro de una circunferencia cuyo
radio se introduce por teclado */

#include <stdio.h>
#define PI 3.1416      /* definición de constante */

int main () {
    float perim, radio; int dos=2;

    printf(" Calcula el perímetro de una circunferencia\n");
    printf(" Indique el tamaño de radio de la circunferencia");
    scanf("%f", &radio);
    perim= dos*PI*radio;
    printf(" El perímetro de la circunferencia es %f\n", perim);
    printf("Valores utilizados para calcular el perímetro:\n");
    printf(" Constante PI=%f, valor de dos = %d, radio=%f ",
PI, dos,radio);
    return 0;
}
```



Ejemplo 8: La función `printf()`

```
/* ejemplo 8.- Ilustra formatos con reales y enteros */

#include <stdio.h>
#define va_int 805
#define va_float 332.41e-1

int main () {
    printf(" %f \n", va_float); /* imprime 33.241*/
    printf(" %.1f \n", va_float); /* imprime 33.2*/
    printf(" %.4f \n", va_float); /* imprime 33.2410*/
    printf(" %.14e \n", va_float); /* imprime 3.3241e+01*/
    printf(" %d \n", va_int); /* imprime 805*/
    printf(" %10f\n", va_float); /* imprime 33.241*/
    return 0;
}
```



Ejemplo 9: La función `scanf()`

```
/* Ejemplo 9.- Introducción de datos por teclado */

#include <stdio.h>

int main () {
    int i;
    float x;
    printf(" teclee el número entero i ");
    scanf("%d", &i);
    printf(" teclee el número real x ");
    scanf("%f", &x);
    return 0;
}
```



Ejemplo 10: La instrucción **if-else**

- Cuando hay muchas opciones el programa se hace difícil de entender
- **getchar()**, lee un carácter de la entrada estándar (teclado)

```
#include <stdio.h>
int main() {
    int c;

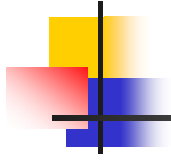
    printf(" Menu:\n");
    printf(" A=Añadir a la lista\n");
    printf(" B=Borrar de la lista\n");
    printf(" O=Ordenar la lista\n");
    printf(" I=Imprimir la lista\n");
    printf(" Escriba su selección y luego <enter>\n");
    if ((c=getchar()) != '\n') {
        if (c=='A') printf(" Has seleccionado añadir\n");
        else if (c=='B') printf(" Has seleccionado borrar\n");
        else if (c=='O') printf(" Has seleccionado ordenar\n");
        else if (c=='I') printf(" Has seleccionado imprimir\n");
    } else printf(" No has seleccionado una opcion valida");
}
```



Ejemplo 11: La instrucción **switch**

```
#include <stdio.h>
int main() {
    int nota;

    printf(" Inserte una nota: "); scanf("%d",&nota);
    switch(nota) {
        case 0: printf("\nSuspenso"); break;
        case 1: printf("\nSuspenso"); break;
        case 2: printf("\nSuspenso"); break;
        case 3: printf("\nSuspenso"); break;
        case 4: printf("\nSuspenso"); break;
        case 5: printf("\nAprobado"); break;
        case 6: printf("\nBien"); break;
        case 7: printf("\nNotable"); break;
        case 8: printf("\nNotable"); break;
        case 9: printf("\nSobresaliente"); break;
        case 10: printf("\nSobresaliente"); break;
        default: printf("esa nota es incorrecta");
    }
    return 0;
}
```



Ejemplo 12: La instrucción switch

```
#include <stdio.h>

int main() {
    int nota;

    printf(" Inserte una nota: "); scanf("%d",&nota);
    switch(nota) {
        case 0: case 1: case 2: case 3: case 4: printf("\nSuspenso"); break;
        case 5: printf("\nAprobado"); break;
        case 6: printf("\nBien"); break;
        case 7:
        case 8: printf("\nNotable"); break;
        case 9:
        case 10: printf("\nSobresaliente"); break;
        default: printf("esa nota es incorrecta");
    }
    return 0;
}
```



Ejemplo 13: La instrucción switch

```
/* ejemplo 13.- Conversión entre euros y pesetas */

#include <stdio.h>
#define euro 166.386

int main() {
    float n,x; int opcion;

    printf("la cantidad: "); scanf("%f",&n);
    printf("1-Ptas a Euros 2-Euros a ptas"); scanf("%d",&opcion);
    switch(opcion) {
        case 1: x=n/euro;
                printf("%f Pesetas son %f Euros\n",n,x); break;
        case 2: x=n*euro;
                printf("%f Euros son %f Pesetas\n",n,x); break;
        default: printf("opción incorrecta\n");
    }
    return 0;
}
```

Punteros y arrays en C

■ Arrays

- Un array es una variable indexable que contiene muchos objetos (denominados elementos) de un mismo tipo que se almacenan consecutivamente en memoria.
- Los elementos están indexados desde 0 hasta n-1

```
int B[10];
int N[5] = { 5, 25, 12, 2, 8 };
int a, b;
a = N[2]; // a es 12
N[2] = 4; // Se modifica 3er elemento
b = N[3]*N[3]; // b vale 4
b = N[10]; // Error de programación!!!
```

500	502	504	506	508		
5	24	4	2	8	...	...
N[0]	N[1]	N[2]	N[3]	N[4]		

– Arrays multidimensionales

```
float v[2][3] = {1, 2, 3, 4, 5, 6 };
char c2[2][3];
float v12 = v[1][2]; // v12 vale 6.0
```

300	304	308	312	316	320	
1.0	2.0	3.0	4.0	5.0	6.0	...
v[0][0]	v[0][1]	v[0][2]	v[1][0]	v[1][1]	v[1][2]	

Punteros y arrays en C

■ Punteros

- Un puntero es una variable que contiene la dirección de otro objeto.
- Se define como: **tipo *nombre**
- El operador & obtiene la dirección de una variable.
- Se denomina *Indirección* devolver el dato apuntado por el puntero(**operador ***)
- La declaración **int N[]** es equivalente a **int *N**;
- Ejemplo: (Compilador asigna memoria a partir posición memoria 500)

```
int b;
int x = 12;
int *p;
int N[3] = {1, 2, 3};
char *pc; //puntero a carácter
p = &x; //p toma la dirección de x
b = *p; //b toma el valor del entero
        //apuntado por p
*p = 10; //El entero apuntado por p,
        //pasa a tomar el valor 10
p = N; //p toma la dirección del
        //primer elto. del array N
```

500	502	504	506	508	510	512
?	12	?	1	2	3	?
b	x	p	N[0]	N[1]	N[2]	pc

500	502	504	506	508	510	512
?	12	502	1	2	3	?
b	x	p	N[0]	N[1]	N[2]	pc

500	502	504	506	508	510	512
12	12	502	1	2	3	?
b	x	*p	p	N[0]	N[1]	N[2]

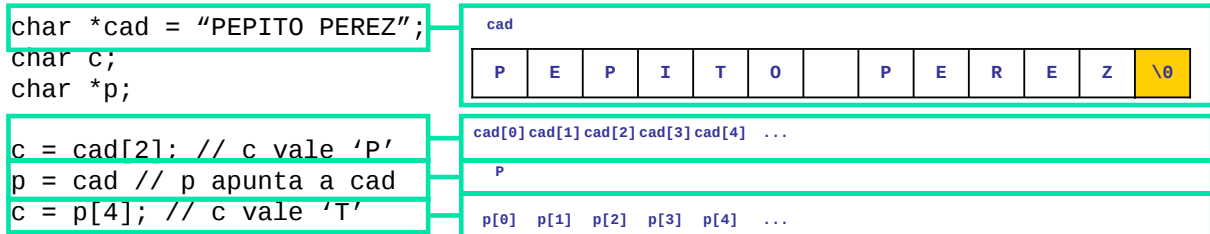
500	502	504	506	508	510	512
12	10	502	1	2	3	?
b	x	*p	p	N[0]	N[1]	N[2]



Punteros y arrays en C

■ Cadenas (*strings*)

- Una cadena es un array de caracteres cuyo último elemento contiene el carácter nulo ('\\0')
- Para trabajar con cadenas utilizar funciones de strings (**strcpy**, **strlen**, etc.)
- Ejemplos:



Punteros y arrays en C

• Aritmética de punteros

- C permite realizar varias operaciones con variable punteros
- Operadores incremento/decremento (++/--).
- Suma y resta (desplazamiento de la posición)
- El desplazamiento es siempre del tipo al que apunta la variable
- Ejemplos:

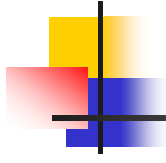
```
int dat[5] = {1, 2, 3, 4, 5};
int *p;
int i, n;

p= dat; //p toma la direccion de
        //dat[0]
p= dat+2; //p toma la dirección de
        //dat[2]
n= sizeof(datos)/sizeof(int);
for (i= 0; i < n; i++) {
    printf("dat[%u]= %u\\t", i, dat[i]);
    printf("p[%u]= %u\\n", i, *p++);
}
```

SALIDA POR PANTALLA>>

dat[0]= 1	p[0]= 3
dat[1]= 2	p[1]= 4
dat[2]= 3	p[2]= 5
dat[3]= 4	p[3]= 8911
dat[4]= 5	p[4]= 8305

500	502	504	506	508	510	512
1	2	3	4	5	?	?
dat[0]	dat[1]	dat[2]	dat[3]	dat[4]	...	...
dat	dat+1	dat+2	dat+3	dat+4	dat+5	...
	p	p+1	p+2	p+3	...	



Punteros y arrays en C

- Otra clase de punteros
 - Puntero genéricos: su tipo es **void** y pueden apuntar a cualquier tipo de dato
 - Puntero nulo: es una variable puntero cuyo valor es 0 (se utiliza **NULL**)
 - El valor **NULL** se utiliza para indicar que ha ocurrido algún error o,
 - Para señalar que un apuntador no apunta a ningún dato (es buena práctica inicializar los punteros a este valor)



Ejemplo 14: Instrucción **while**

```
/* ejemplo 14.- Lee una palabra y lo escribe al revés */
```

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

```
int main () {  
    char l, palabra[21];  
    int i;  
  
    printf("Teclee una palabra de menos de 20 letras:");  
    scanf("%s", palabra);  
    i = 0;  
    while(palabra[i++] != '\0') ;  
    l = i-1; printf("%s tiene %d letras\n", palabra, l);  
    printf("%s escrita al revés es: ", palabra);  
    i = l;  
    while (i > 0)  
        printf("%c", palabra[--i]);  
    return 0;  
}
```

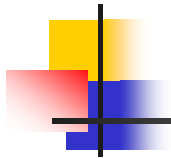


Ejemplo 15: do-while

```
/* ejemplo 15.- Suma de n números introducidos por teclado (valor  
    <0 para terminar) */
```

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

```
int main() {  
    int num=0,suma=0;  
  
    do {  
        suma=suma+num;  
        printf("un número: ");  
        scanf("%d",&num);  
    } while(num>=0);  
    printf("suma es: %d",suma);  
    return 0;  
}
```



Ejemplo 16: for y parámetros por valor

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

```
int f(int m, int n);
```

```
int main() {  
    int i;  
    printf("Prueba: función entero  
    elevado a potencia");  
    for (i=0; i<10; i++) {  
        printf("2^%d es:%d",i,f(2,i));  
        printf("3^%d es:%d",i,f(3,i));  
    }  
    return 0;  
}
```

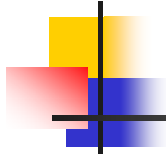
```
int f(int base,int n) {  
    int i,p=1;  
  
    for (i=1;i<=n ;i++) p=p*base;  
    return p;  
}
```

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

```
int f(int m, int n);
```

```
int main() {  
    int i;  
    printf("Prueba: función entero  
    elevado a potencia");  
    for (i=0; i<10; i++) {  
        printf("2^%d es:%d",i,f(2,i));  
        printf("3^%d es:%d",i,f(3,i));  
    }  
    return 0;  
}
```

```
int f(int base,int n) {  
    int p=1;  
  
    while (n-- >0) p=p*base;  
    return p;  
}
```



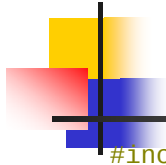
Ejemplo 17: Parámetros por referencia

```
#include <stdio.h>

void intercambio(int *px,int *py);
int main() {
    int a, b;

    printf("a= " );scanf("%d", &a);
    printf("b= " );scanf("%d", &b);
    intercambio(&a,&b);
    printf("a= %d\tb= %d" , a, b);
}
void intercambio(int *px,int *py) {
    int temp;

    temp =*px; /* temp contiene el valor de px*/
    *px=*py; /* px y py contienen el mismo valor */
    *py= temp; /* py contienen el valor inicial de px*/
}
```



Ejemplo 18: Funciones de cadenas

```
#include <stdio.h>
#include <string.h>
#define MSG "Introduce tira de caracteres: "
void compara (char a[], char b[]) {
    int i=strcmp(a,b);
    if (!i) printf("iguales"); /* falso <-> i=0 */
    else if (i<0) printf("%s > %s",a,b); else printf("%s < %s",a,b);
}
void concatena (char a[], char b[]) {
    if (strlen(a)+strlen(b)<80) {strcat (a,b); printf ("%s",a);}
}
void copia (char a[], char b[]) {
    strcpy(a,b); printf ("%s y %s",a,b);
}
void longitud (char a[], char b[]) {
    printf("%s = longitud %d: %s = longitud %d",a,strlen(a),b,strlen(b));
}
int main() {
    char x[80], y[80]; /* vectores */
    printf (MSG); scanf ("%s",x); printf (MSG); scanf ("%s",y);
    longitud(x,y); compara(x,y); concatena(x,y); copia(x,y);
}
```



Ejemplo 19: Argumentos en línea de ordenes

```
#include <stdio.h>

int main(int argc, char* argv[]) {
    int i;

    /* Prueba de parámetros para función main */
    printf("Se han pasado %3d argumentos:", argc);
    for(i=0; i<argc; i++) printf("arg[%5d]= %s", i, argv[i]);
    return 0;
}
```

- Ejecución de **ej19**:

ej19 parametro1 parametro2

- Salida:

```
arg[0]= ej19
arg[1]= parametro1
arg[2]= parametro2
```



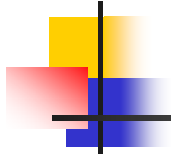
Ejemplo 20: función malloc()

```
/* ejemplo 20 .- Creación de un vector de enteros */

#include <stdio.h>
#include <stdlib.h>
#define N 10

int main(void) {
    int *p, i;

    p = (int *)malloc(N * sizeof(int));
    if (p == NULL) {
        printf("Error: no había memoria"); exit(-1);
    }
    for (i = 0; i < N; i++) p[i] = i;
    for (i = 0; i < N; i++) printf("i: %d v[%d]: %d", i, i, p[i]);
    free(p);
    return 0;
}
```



Ejemplo 21: Función `atoi()`

```
int atoi(const char *numPtr);
```

- Convierte la porción inicial de la cadena apuntada por `numPtr` a una representación de `int`.
- Valor de retorno:
 - La función `atoi()` retorna un entero que es el valor convertido.

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

```
int main() {
    char numPtr[5] = "1234";

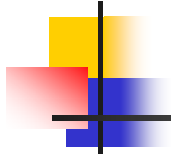
    printf( "Convirtiendo la cadena %s en un numero: %d\n",
            numPtr, atoi(numPtr));
    return 0;
}
```



Ejemplo 22: Función `itoa()`

```
#include <stdio.h>
#include <string.h>
#define MAX_LEN 12
void natural (int n, char s[]) {
    int i=0, j; char t[MAX_LEN];
    do {
        t[i++]=n%10+(int)'0';
    } while ((n/=10)>0);
    for (j=0; j<i; j++) s[j]=t[i-j-1];
    s[i]='\0';
}
void entero (int n, char s[]) {
    if (n<0) {s[0]='-'; natural(-n,s+1);} else natural(n,s);
}
int main() {
    int n; char s[MAX_LEN];

    printf("escribe el entero a convertir");
    scanf("%d",&n); entero(n,s); printf("%s",s);
}
```



Ejemplo 23.- Función `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

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

```
int main() {
    char s1[11] = "Hola ", s2[6] = "amigos";
    printf( "s1=%s\n", s1 ); printf( "s2=%s\n", s2 );
    strcat( s1, s2 );
    printf( "s1=%s\n", s1 );
    return 0;
}
```



Ejemplo 24.- Función `strlen()`

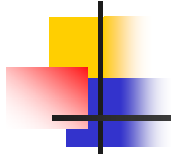
size_t strlen(const char *s);

- Calcula el número de caracteres de la cadena apuntada por **s**
- Valor de retorno:
 - La función retorna el número de caracteres hasta el carácter nulo, que no se incluye

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

```
int main() {
    char s[13] = "Hola a todos";

    printf( "s=%s\n", s );
    printf( "strlen(s) = %d\n", strlen( s ) );
    return 0;
}
```



Ejemplo 25.- Función strcpy()

char *strcpy(char *s1, const char *s2);

- Copia la cadena apuntada por **s2** (incluyendo el carácter nulo) a la cadena apuntada por **s1**
- Valor de retorno:
 - La función retorna el valor de **s1**. Si al copiar una cadena a la otra se superponen, entonces el comportamiento no está definido

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

int main() {
    char s2[7] = "abcdefg";
    char s1[7];

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



Ejemplo 26.- Función strcmp()

int strcmp(const char *s1, const char *s2);

- Compara la cadena apuntada por **s1** con la cadena apuntada por **s2**.
- Valor de retorno:
 - La función retorna un número entero mayor, igual, o menor que cero, apropiadamente según la cadena apuntada por **s1** es mayor, igual, o menor que la cadena apuntada por **s2**.

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

int main() {
    char s1[5] = "Abeja", s2[5] = "abeja"; int i;

    printf( "s1=%s\n", s1 ); printf( "s2=%s\n", s2 );
    i = strcmp( s1, s2 ); printf( "s1 es " );
    if( i < 0 ) printf( "menor que" );
    else if( i > 0 ) printf( "mayor que" );
    else printf( "igual a" );
    printf( " s2\n" );
}
```