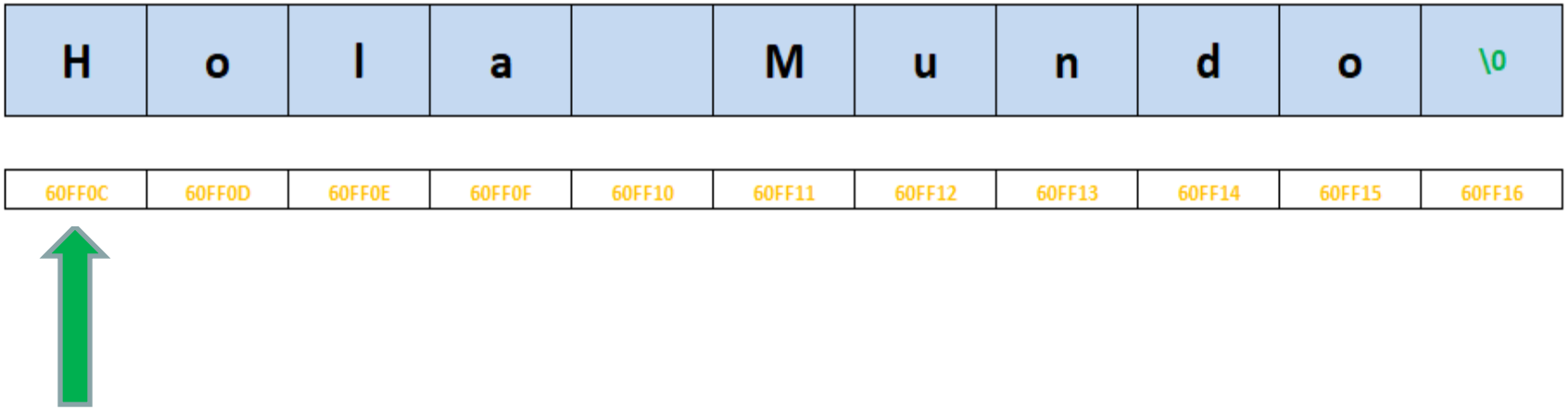
Arrays y su relación con cadena de caracteres

• En programación denominamos ‘string’ o cadenas a una secuencia de caracteres. Por ejemplo:
”Hola mundo!”

• C no soporta el tipo string por lo tanto tiene una librería: string.h con funciones para el tratamiento de las cadenas.

Arrays y cadena de caracteres

- Una cadena es un array de caracteres o un puntero a una porción de memoria, que contiene caracteres ASCII.
- Finalizan con un carácter nulo o '\0' que muchas funciones utilizan para saber donde finaliza la cadena de caracteres.
- Todos los elementos son de tipo char.



Declaración:

```
char  mi_cadena[11];
char mi_cadena [ ];
```

O, puede emplearse una constante simbólica:

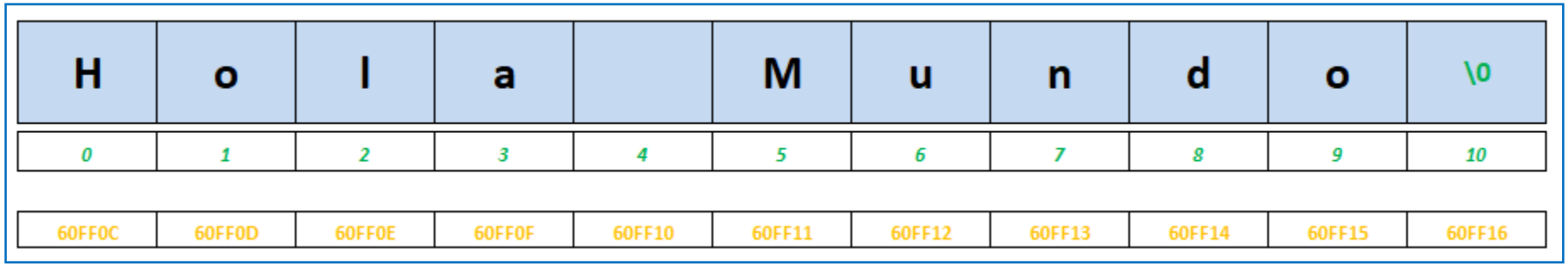
```
#define TAM 11
. . .
char  mi_cadena[TAM];
```

Inicialización:

Las cadenas de caracteres se declaran e inicializan como cualquier array.

Arrays y cadena de caracteres

Ejemplo: **char mi_cadena[11]={‘H’, ’o’, ’l’, ’a’, ’ ’, ’m’, ’u’, ’n’, ’d’, ’o’};**



- ‘H’ es un carácter, ‘o’ también, y así sucesivamente.
- «Hola mundo» es una cadena de caracteres.
- El carácter ‘\0’ (barra cero) lo agrega el sistema. Es no imprimible, es decir que no lo podemos ver, pero sí podemos recorrer la cadena de caracteres con un ciclo hasta el ‘\0’

Arrays y cadena de caracteres

Se puede omitir la dimensión de un array si se inicializa en la declaración, en caso contrario es forzoso declarar la dimensión.

Otro ejemplo: **char mi_cadena[ ] = “Hola mundo”;**

H	o	l	a		M	u	n	d	o	\0
0	1	2	3	4	5	6	7	8	9	10

Otro ejemplo: **char mi_cadena[11]= “Hola”;**

H	o	l	a	\0						
0	1	2	3	4	5	6	7	8	9	10

Espacios no utilizados

Arrays y cadena de caracteres – Ejemplo 1

Se puede ingresar una cadena *caracter por carácter* utilizando la función scanf:

scanf(“%c”, &mi_cadena [i]);

```
#include <stdio.h>

int main(){
const int dim = 11;
char mi_cadena[dim];
int cont=0;

  while (cont < (dim - 1)){
    printf( "Ingrese una letra de su cadena: \n" );
    scanf( "%c", &mi_cadena[cont] );
    fflush(stdin);
    cont++;
}

printf ("Mi cadena es: %s\n", mi_cadena);
printf ("contador lleg%c a: %d\n", 162, cont);

return 0;
}
```

```
Ingrese una letra de su cadena:
H
Ingrese una letra de su cadena:
o
Ingrese una letra de su cadena:
l
Ingrese una letra de su cadena:
a
Ingrese una letra de su cadena:

Ingrese una letra de su cadena:
m
Ingrese una letra de su cadena:
u
Ingrese una letra de su cadena:
n
Ingrese una letra de su cadena:
d
Ingrese una letra de su cadena:
o
Mi cadena es: Hola
mundo
contador llegó a: 10
```

Evidentemente es poco práctico…

Arrays y cadena de caracteres – Ejemplo 2

O se puede ingresar el texto completo utilizando la función scanf:

scanf(“%s”, mi_cadena);

En este caso, el nombre del array no lleva el operador de dirección de memoria &, porque lleva implícito la dirección de memoria.

```
#include <stdio.h>

int main(){
const int dim = 11;
char mi_cadena[dim];

printf( "Ingrese una cadena: \n" );
scanf( "%s", mi_cadena);
printf ("Mi cadena es: %s\n", mi_cadena);

printf( "Ingrese una cadena: \n" );
scanf( "%s", mi_cadena);
printf ("Mi cadena es: %s\n", mi_cadena);

printf( "Ingrese una cadena: \n" );
scanf( "%s", mi_cadena);
printf ("Mi cadena es: %s\n", mi_cadena);

return 0;
}
```

```
Ingrese una cadena:
Hola
Mi cadena es: Hola
Ingrese una cadena:
Hola_Mundo
Mi cadena es: Hola_Mundo
Ingrese una cadena:
Hola Mundo
Mi cadena es: Hola
```

Pero en el último caso no se emitió la cadena completa... Qué pasó?

Arrays y cadena de caracteres - Ejemplo 3

La función scanf guarda lo que pulsemos por teclado hasta el primer espacio en blanco o enter, por lo cual, si ingresamos una cadena compuesta, sólo se emitirá la cadena hasta el primer espacio blanco. Podemos evitarlo utilizando:

scanf("%[^\n]", mi_cadena);

O más sencillo

gets(mi_cadena);

```
#include <stdio.h>

int main(){
const int dim = 11;
char mi_cadena[dim];

printf( "Ingrese una cadena: \n" );
scanf("%[^\n]", mi_cadena);
fflush(stdin);
printf ("Mi cadena es: %s\n", mi_cadena);

printf( "Ingrese una cadena: \n" );
gets(mi_cadena);
fflush(stdin);
printf ("Mi cadena es: %s\n", mi_cadena);

return 0;
}
```

```
Ingrese una cadena:
Hola Mundo
Mi cadena es: Hola Mundo
Ingrese una cadena:
Hola Mundo
Mi cadena es: Hola Mundo
```

Arrays y cadena de caracteres - Ejemplo 4

Y las cadenas pueden emitirse caracter a caracter:

printf ("%c", mi_cadena[i]);

```
#include <stdio.h>

int main(){
const int dim = 11;
char mi_cadena[dim];
int cont=0;

printf( "Ingrese una cadena: \n" );
gets(mi_cadena);
printf ("\nEmitiendo caracter a caracter....\n");

  while (mi_cadena[cont] != '\0'){
    printf( "%c", mi_cadena[cont] );
    cont++;
}

printf ("\n\nMi cadena es: %s\n", mi_cadena);
printf ("\nContador lleg%c a: %d\n", 162, cont);

return 0;
}
```

```
Ingrese una cadena:
Hola Mundo

Emitiendo caracter a caracter....
Hola Mundo

Mi cadena es: Hola Mundo

Contador llegó a: 10
```

Arrays y cadena de caracteres - Ejemplo 5

O pueden emitirse completas:

printf ("%s", mi_cadena);

```
#include <stdio.h>

int main(){
const int dim = 11;
char mi_cadena[dim];

printf( "Ingrese una cadena: \n" );
gets(mi_cadena);

printf ("\nMi cadena es: %s\n", mi_cadena);

return 0;
}
```

```
Ingrese una cadena:
Hola Mundo

Mi cadena es: Hola Mundo
```

printf recibe un apuntador al inicio de la cadena, es decir, se tiene acceso a una cadena constante por un puntero a su primer elemento. Emitir la cadena es una característica propia de la función.

%s indica que interprete esa dirección como el comienzo de una cadena y la función interpreta caracter a caracter hasta encontrar el '\0'.

Arrays y cadena de caracteres

Otra forma de declaración es la siguiente:

char * mi_cadena;

Entonces:

otra_ cadena = “Hola Mundo”;

Asigna un puntero al array de caracteres. No es la copia de una cadena, es un puntero. ¿Existe alguna diferencia entre ellas?, sí:

char mi_cadena[ ] = “Hola mundo”; — **Es un array**

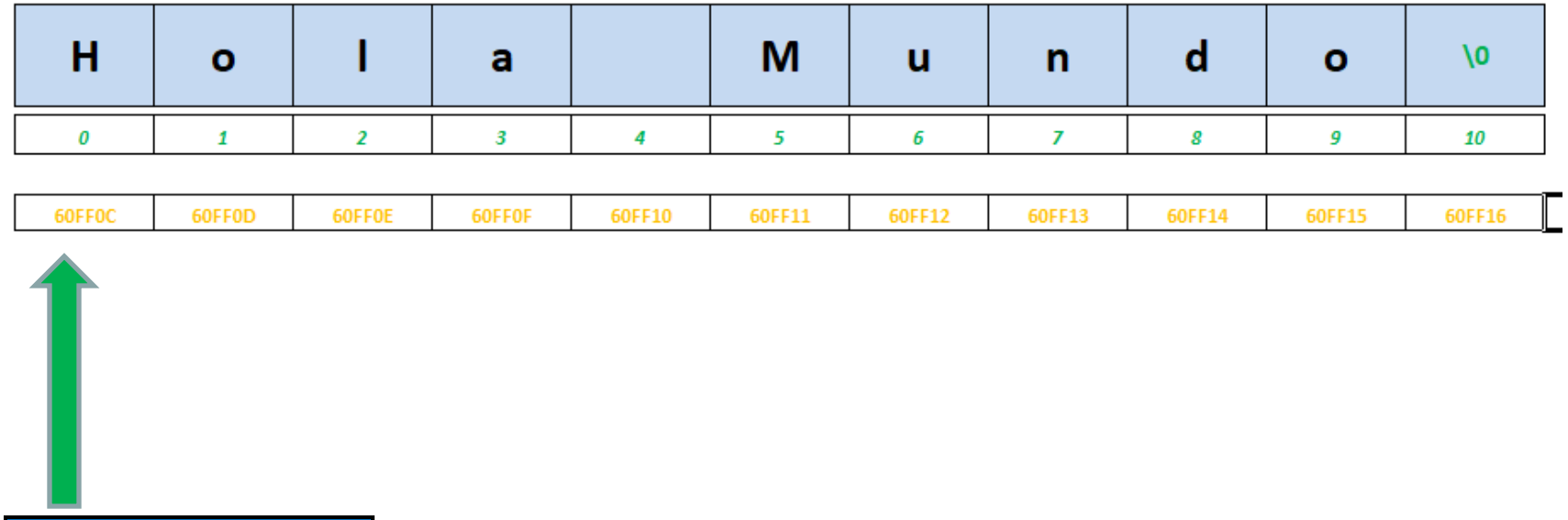
char * mi_cadena = “Hola mundo”; — **Es un puntero al array**

Diferencia entre:
char mi_cadena[ ]
y
char * mi_cadena

Arrays y cadena de caracteres

char mi_cadena[ ] = “Hola mundo”;

- ‘mi_cadena’ es un puntero *constante*, no se puede cambiar, no es una variable.
- Puede cambiarse el contenido de la cadena mientras no se cambie el tamaño.
- Usando la notación de arrays, los bytes de almacenamiento - en el bloque de memoria estática - son tomados uno para cada caracter y uno para el caracter de terminación’\0’.



Arrays y cadena de caracteres - Ejemplo 6

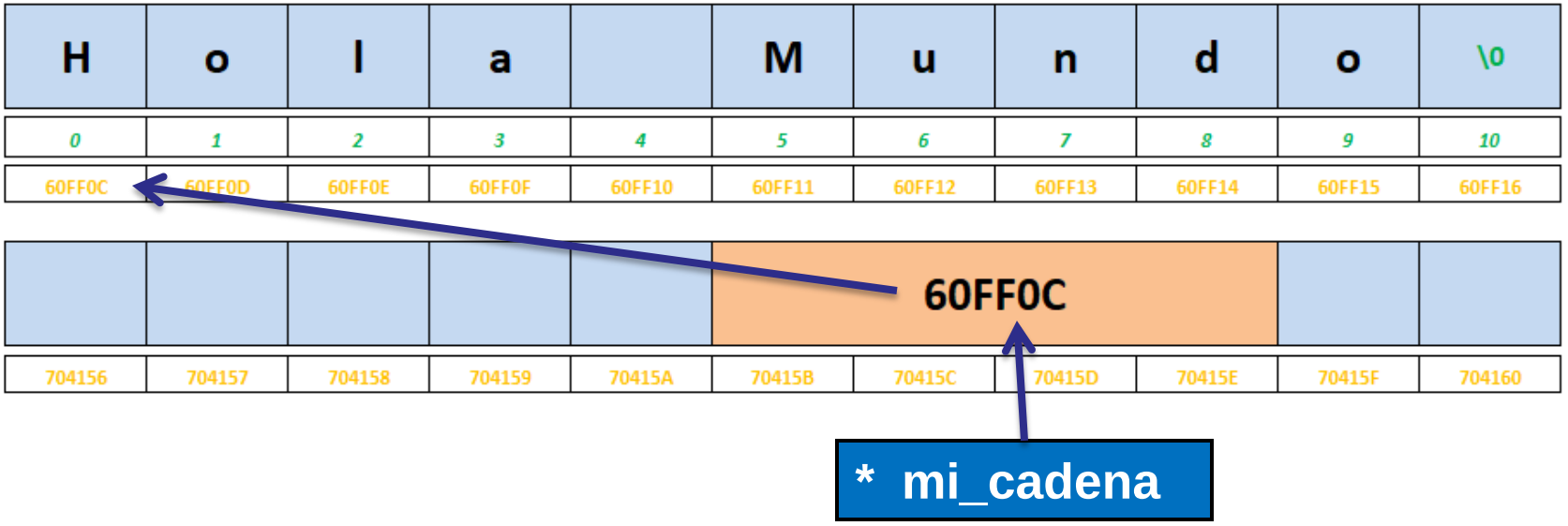
```
#include <stdio.h>

int main(){
const int dim = 11;
char mi_cadena[dim];
int cont=0;

printf( "Ingrese una cadena: \n" );
gets(mi_cadena);
printf ("\nMi cadena original es: %s\n", mi_cadena);

printf ("\nCambiando algunos caracteres....\n");
  while (mi_cadena[cont] != '\0'){

    if ((mi_cadena[cont] == 'a') || (mi_cadena[cont] == 'o') || (mi_cadena[cont] == 'u'))
      mi_cadena[cont] = 'e';
    cont++;
}

printf ("\n\nMi cadena ahora es: %s\n", mi_cadena);
printf ("\nContador lleg%c a: %d\n", 162, cont);

return 0;
}
```

```
Ingrese una cadena:
Hola Mundo

Mi cadena original es: Hola Mundo

Cambiando algunos caracteres....


Mi cadena ahora es: Hele Mende

Contador llegó a: 10
```

Arrays y cadena de caracteres

char * mi_cadena = “Hola mundo”; //literal de cadena

- Aquí ‘mi_cadena’ es una variable puntero inicializada para apuntar a un literal de cadena que se aloja en una zona solo-lectura de la memoria. Puede cambiarse el apuntador.
- Intentar cambiar el contenido de la cadena puede dar error o un resultado indefinido, esto significa que el programa puede fallar intermitentemente, de formas distintas, y en condiciones poco esperadas
- En este tipo de notación, son requeridos los bytes de la cadena más n bytes para alojar la variable puntero.
- Los literales de cadena se almacenan, en el segmento de datos, es el mismo sitio que se reserva para los valores constantes y variables globales. Un literal es tratado como una constante.



Arrays y cadena de caracteres - Ejemplo 7

```
#include <stdio.h>

int main(){

char * mi_cadena = "Hola Mundo";
char * otra_cadena = "Hola Mundo";

printf ("\nMi otra_cadena original es: %s\n", otra_cadena);
otra_cadena = NULL; //el puntero a char no es constante
printf ("\nMi otra_cadena original es: %s\n", otra_cadena);

printf ("\nMi cadena original es: %s\n", mi_cadena);
mi_cadena+=2;
printf ("\nMi cadena original ahora es: %s\n", mi_cadena);

*mi_cadena = 'x'; // resultado inesperado...
printf ("\nMi cadena original ahora es: %s\n", mi_cadena);

return 0;
}
```

```
Mi otra_cadena original es: Hola Mundo

Mi otra_cadena original es: (null)

Mi cadena original es: Hola Mundo

Mi cadena original ahora es: la Mundo
```

Arrays y cadena de caracteres

Entonces:

```
void mi_funcion_A (char * ptr) {
char mi_cadena[ ] = “ABCDE”;
... }
```

En el caso de mi_funcion_A, el contenido, o valor(es), del array ‘mi_cadena[ ]’ son considerados como los datos. Se dice que ha sido inicializado a los valores ABCDE.

```
void mi_funcion_B (char * ptr) {
char * mi_cadena  = “FGHIJ”;
…}
```

En el caso de mi_funcion_B, el valor del puntero ‘mi_cadena’ se considera como dato. El puntero ha sido inicializado para apuntar a la cadena FGHIJ.

En ambas funciones las definiciones son variables locales y por tanto la cadena ABCDE se guarda en la pila (stack), así como el valor del apuntador mi_cadena. La cadena FGHIJ se guarda en el segmento de datos.

Arrays y cadena de caracteres

Para realizar operaciones con caracteres existen funciones de biblioteca definidas en **ctype.h**

Ejemplos de algunas funciones para tratamiento de caracteres son:

isalnum(caracter): devuelve cierto (un entero cualquiera distinto de cero) si caracter es una letra o dígito, y falso (el valor entero 0) en caso contrario.
isalpha(caracter): devuelve cierto si caracter es una letra, y falso en caso contrario.
isblank(caracter): devuelve cierto si caracter es un espacio en blanco o un tabulador.
isdigit(caracter): devuelve cierto si caracter es un digito, y falso en caso contrario.
isspace(caracter): devuelve cierto si caracter es un espacio en blanco, un salto de línea, un retorno de carro, un tabulador, etc., y falso en caso contrario.
islower(caracter): devuelve cierto si caracter es una letra minúscula, y falso en caso contrario.
isupper(caracter): devuelve cierto si caracter es una letra mayúscula, y falso en caso contrario.
toupper(caracter): devuelve la mayúscula asociada a caracter, si la tiene; si no, devuelve el mismo caracter.
tolower(caracter): devuelve la minúscula asociada a caracter, si la tiene; si no, devuelve el mismo caracter.

Arrays y cadena de caracteres

Para realizar operaciones con cadenas hay que usar funciones de biblioteca. El lenguaje no dispone de operadores para cadenas pero se utilizarán funciones definidas en **string.h**

Algunos ejemplos de funciones para tratamiento de cadenas son:

Función	Objetivo	Valor de retorno
char *strcpy(char *s1, const char *s2);	**Copia** la cadena apuntada por s2 (incluyendo el carácter nulo) a la cadena apuntada por s1.	La función retorna el valor de s1.
int strcmp(const char *s1, const char *s2);	**Compara** la cadena apuntada por s1 con la cadena apuntada por s2.	La función retorna un número entero mayor, igual, o menor que cero, según si la cadena apuntada por s1 es mayor, igual, o menor que la cadena apuntada por s2.
char *strcat(char*s1, const char *s2);	**Agrega** 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	La función retorna el valor de s1.
size_t strlen(const char *s);	**Calcula** el número de caracteres de la cadena apuntada por s.	La función retorna el número de caracteres hasta el carácter nulo, que no se incluye.

Arrays y cadena de caracteres – Ejemplo 8

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

int main() {
char cad_1[ ] = "Abeja";
char cad_2[ ] = "Abejita";
int valor;

printf( "cad_1 = %s\n", cad_1 );
printf( "cad_2 = %s\n", cad_2 );

valor = strcmp( cad_1, cad_2 );

printf( "\nPrestar atenci%cn a este valor devuelto por strcmp: %d\n ", 162,valor);

    if( valor < 0 )
        printf( " %s es menor que %s\n", cad_1, cad_2 );
    else if( valor > 0 )
        printf( " %s es mayor que %s\n", cad_1, cad_2 );
    else
        printf( "Son iguales\n" );

return 0;
}
```

```
cad_1 = Abeja
cad_2 = Abejita

Prestar atención a este valor devuelto por strcmp: -1
  Abeja es menor que Abejita
```

Arrays y cadena de caracteres - Ejemplo 9

```
#include <stdio.h>
#include <string.h>
int main()
{
char cad_1[30] = "Me encantan las cadenas en C";
char cad_2[30];

printf( "Cadena 1: %s\n", cad_1 );
printf( "Cadena 2: %s\n", cad_2 );

strcpy( cad_2, cad_1 );

printf( "Cadena 1: %s\n", cad_1 );
printf( "Cadena 2: %s\n", cad_2 );

printf("Tama%co de cadena 1: %d\n", 164,strlen(cad_1));
printf("Tama%co de cadena 2: %d\n", 164, strlen(cad_2));

return 0;
}
```

```
Cadena 1: Me encantan las cadenas en C
Cadena 2:
Cadena 1: Me encantan las cadenas en C
Cadena 2: Me encantan las cadenas en C
```

Arrays y cadena de caracteres - Ejemplo 10

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

int main() {
char cad_1[ ] = "Me encantan ";
char cad_2[ ] = "las cadenas en C";
int i;

printf( "cad_1 = %s\n", cad_1 );
printf( "cad_2 = %s\n", cad_2 );

strcat( cad_1, cad_2 );

printf( "cad_1 = %s\n", cad_1 );
printf( "cad_2 = %s\n", cad_2 );


return 0;
}
```

```
cad_1 = Me encantan
cad_2 = las cadenas en C
cad_1 = Me encantan las cadenas en C
cad_2 = las cadenas en C
```

Fin arrays unidimensionales