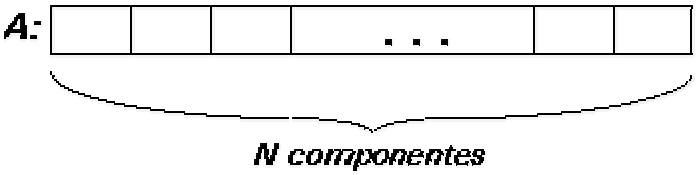
Arrays y Cadenas en C

Los arrays son variables estructuradas, donde cada elemento se almacena de forma consecutiva en memoria.

Las cadenas de caracteres son declaradas en C como arrays de caracteres y permiten la utilización de un cierto número de notaciones y de funciones especiales.

Arrays de una dimensión

Un array (unidimensional, también denominado vector) es una variable estructurada formada de un número "n" de variables simples del mismo tipo que son denominadas los componentes o elementos del array. El número de componentes "n" es, entonces, la dimensión del array. De igual manera que en matemáticas, decimos que "A" es un vector de dimensión "n".



El formato para declarar un array unidimensional es:

```
tipo nombre[n];
```

donde: n >= 1

Para acceder a un elemento del array:

```
nombre[i];
```

donde: 0 <= i < n

Por ejemplo, la declaración:

```
int A[4];
```

define un array de tipo entero de dimensión 4. Y ya podríamos acceder al primer componente del array por medio de: A[0], al segundo elemento por: A[1] y al último elemento por A[3].

En C, un array se utiliza básicamente cuando queremos tener, por ejemplo, una secuencia de números reunidos en una sola variable.

Para inicializar un array, podemos hacer lo siguiente:

```
for (i = 0; i< 4; i++)
   A[i] = i;
```

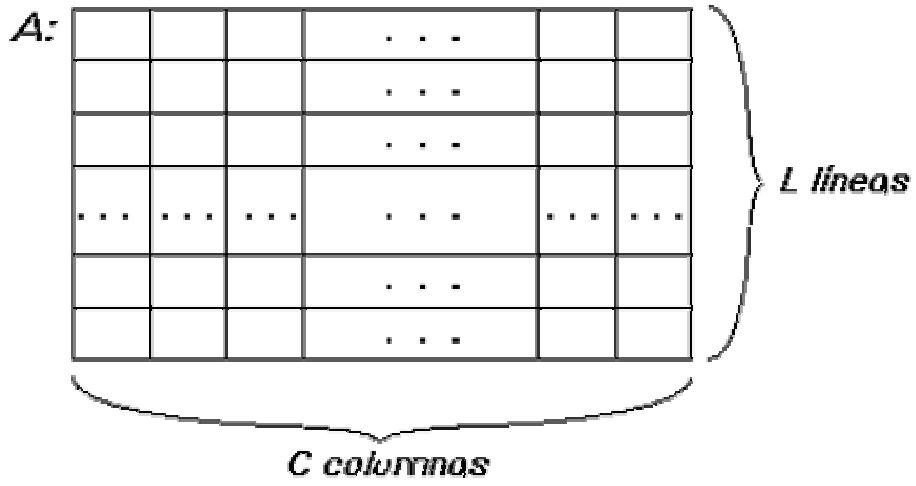
Otra manera de inicializar un array es asignándole los valores iniciales entre llaves de la siguiente:

```
int A[4] = {0, 1, 2, 3};
```

Si no se inicializa explícitamente el array no se puede estar seguro del valor que contienen los elementos del mismo.

Arrays de dos dimensiones

Un array en C puede tener una, dos o más dimensiones. Por ejmplo, un array de dos dimensiones también denominado matriz, es interpretado como un array (umidimensional) de dimensión "f" (número de filas), donde cada componente es un array (unidimensional) de dimensión "c" (número de columnas). Un array de dos dimensiones, contiene, pues, "f*c" componentes.



El formato para declarar un array multidimensionales:

```
int nombre[f][c]…;
```

donde: f,c...>= 1;

Para acceder a un elemento del array multidimensional:

```
nombre[i][j];
```

donde: 0 <= i < f ; 0 <= j < c;

Durante la declaración de un array multidimensional también podemos inicializar sus componentes indicando la lista de los valores entre llaves. En el interior de la lista, los componentes de cada línea del array son encerrados nuevamente entre llaves. Para hacer más cara la visibilidad de los elementos del array, podemos indicarlos en varias líneas.

```
int A[3][4] ={{ 0,1,2,3},
              { 1,2,3,4},
              { 2,3,4,5}};
```

Sin embargo, es mucho más conveniente anidar dos ciclos para inicializar un array de dos dimensiones:

```
for (i = 0; i< 3; i++)
   for (j = 0; j< 4; j++)
      A[i][j] = i+j;
```

Cadenas de caracteres

En C no existe un tipo predefinido para manipular cadenas de caracteres (*string*). Sin embargo, el estándar de C define algunas funciones de biblioteca para tratamiento de cadenas.

Una cadena en C es un array de caracteres de una dimensión (vector de caracteres) que termina con el carácter especial '\0' (cero).

El formato para declarar una cadena es:

```
char nombre[n];
```

donde: n >= 1 y representa a la longitud-1 real de la cadena.

Un ejemplo de declaración de cadena:

```
char cadena [5];
```

Debido a que en la representación interna de una cadena de caracteres es terminada por el símbolo '\0', para un texto de "n" caracteres, debemos reservar "n+1". El carácter '\0', aunque pertenece a la cadena, no aparece al utilizar funciones como printf.

En el caso especial de los arrays de caracteres, podemos utilizar varias formas de inicialización:

```
char cadena[] = "Hola";
char cadena[] = {'H','o','l','a',0};
char cadena[] = {'H','o','l','a','\0'};
```

sin especificar el tamaño de la cadena, o especificando el tamaño:

```
char cadena[5] = "Hola";
char cadena[5] = {'H','o','l','a',0};
char cadena[5] = {'H','o','l','a','\0'};
```

Durante la inicialización, se reserva automáticamente el número de bytes necesarios para la cadena, esto es, el número de caracteres más uno. Por ejemplo:

```
char TXT[] = "Hello";
```

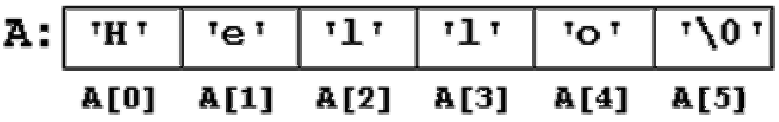
Para acceder a un elemento de una cadena de caracteres puede hacerse de la misma manera que el acceso al elemento de un array.

```
cadena[i];
```

donde: 0 <=i < n

Por ejemplo:

```
char A[6] = "Hello";
```



La biblioteca “string” tiene una gran cantidad de funciones prácticas para trabajar con cadenas de caracteres. Para utilizarlas debemos de incluir el fichero que define los prototipos de dichas funciones:

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

Algunas de las funciones más importantes son:

- `strlen(<cadena>)`: Devuelve la longitud de la cadena sin tomar en cuenta el caracter de final de cadena.
- `strcpy(<cadena_destino>, <cadena_origen>)` : Copia el contenido de `<cadena_origen>` en `<cadena_destino>`.
- `strcat(<cadena_destino>, <cadena_origen>)` : Concatena el contenido de `<cadena_origen>` al final de `<cadena_destino>`.
- `strcmp(<cadena1>, <cadena2>)` : Compara las dos cadenas y devuelve un 0 si las dos cadenas son iguales, un número negativo si

`<cadena1>` es menor que (precede alfabéticamente a) `<cadena2>` y un número positivo (mayor que cero) si `<cadena1>` es mayor que `<cadena2>`.

A diferencia de los arrays de tipos de datos numéricos (arrays de enteros, de números con punto decimal, etc.), en donde cada elemento del array se debe considerar como una variable independiente de los demás, los arrays de caracteres (cadenas) se pueden manipular de dos maneras: de forma conjunta o separada.

Por ejemplo, para mostrar en pantalla un array de caracteres podemos hacerlo dentro de un bucle, desde el primer caracter (indice 0) hasta el último carácter (lo que nos devuelve la función strlen):

```
for(i=0; i<strlen(cadena); i++)
  printf("%c",cadena[i]);
```

Existe una mejor manera de mostrar en pantalla una cadena, y es utilizando el carácter de conversión %s:

```
printf("%s",cadena);
```