# Práctica 5. Vectores, matrices y números aleatorios

**Objetivos de la práctica:**

- Adquirir soltura resolviendo problemas sencillos utilizando el lenguaje C
- Comprender cómo se usan los vectores y matrices
- Comprender cómo se usan los números aleatorios

## 1. La librería `stdlib.h`

La librería `stdlib.h` contiene funciones útiles cuando queremos obtener un número aleatorio. Los números aleatorios se usan en programación para simular por ejemplo el resultado de tirar un dado, distribuir los premios ocultos en un juego (arkanoid), inicializar una jugada (posición de barcos en hundir la flota), etc. Mediante la función `rand()` obtenemos un número aleatorio entre `0` y el valor especificado por la constante `RAND_MAX`. El valor de la constante `RAND_MAX` depende del compilador, pero como mínimo vale `0x7FFF`.

Generalmente se quiere un número dentro de un rango, es decir, por ejemplo de 0 a 100. Lo que se hace es dividir entre el máximo más 1 y quedarnos con el resto. Así garantizamos que el número obtenido está entre 0 y 100. Genéricamente, para obtener un número aleatorio entre *0* y *N*, tendríamos que hacer lo siguiente:

```
num_aleatorio=rand()% (N+1);
```

El problema de la función `rand()` es que para calcular el número aleatorio usa un algoritmo que parte de un número (semilla) y a partir de él calcula los números aleatorios. Por eso, si en nuestro programa la llamamos 4 veces, saldrán 4 números aleatorios. Sin embargo, si volvemos a ejecutar el mismo programa, nos saldrán los mismos números en la misma secuencia. Esto es porque siempre usa la misma semilla (número inicial).

La solución a esto nos la proporciona la función `srand()`. Esta función nos permite asignar una semilla (la que se le pase como parámetro). Así, si cambiamos la semilla, la secuencia de números aleatorios no será siempre la misma.

El número que le pasamos a `srand()` no puede ser un valor fijo, porque estaríamos en el mismo caso de antes. Lo que se hace es pasarle un valor que sea distinto en cada ejecución del programa, como por ejemplo la hora/fecha del sistema (cambia en cada instante).

Para obtener la hora del sistema se usa la función `time()`, que está en la librería `time.h`.

En resumen, para calcular un número aleatorio en un programa deberemos:

- Indicar una semilla mediante la instrucción `srand(time(NULL));` al principio de nuestro programa.

– Obtener los números aleatorios que queramos mediante la instrucción `num_aleatorio=rand()% (N+1);`

# 2. Los vectores en C

Los vectores nos permiten almacenar datos de un mismo tipo. En C se declaran de la siguiente forma:

`tipo_de_datos nombre_vector[TAMANYO];`

Podemos inicializarlos dándole valores que van entre llaves y separados por comas:

`tipo_de_datos nombre_vector[TAMANYO]={valor1, valor2, valor3, valorn};`

Para acceder a una posición del vector, se utiliza la siguiente notación, teniendo en cuenta que el índice va desde 0 hasta el tamaño del vector menos 1:

`nombre_vector[indice]`

# 3. Las matrices en C

Las matrices nos permiten almacenar datos de un mismo tipo en organizaciones 2D. En C se declaran de la siguiente forma:

`tipo_de_datos nombre_matriz[NUMFILAS][NUMCOLS];`

Podemos inicializarlos dándole valores que van entre llaves y separados por comas:

```
tipo_de_datos nombre_matriz[NUMFILAS][NUMCOLS]={{valor1fila1, valor2fila1,
valor3fila1, valornfila1}, {valor1fila2, valor2fila2, valor3fila2,
valornfila2},{valor1filam, valor2filam, valor3filam, valornfilam}};
```

Para acceder a una posición de la matriz, se utiliza la siguiente notación, teniendo en cuenta que el índice va desde 0 hasta el tamaño de la fila/columna menos 1:

`nombre_matriz[indicefila][indicecolumna]`

# 4. Ejercicios propuestos

## 4.1. Ejercicio 1

Realice un programa `passwords.c` en lenguaje C que genere contraseñas aleatorias de 8 minúsculas. Para ello tiene que contener una función que inicialice un vector de 26 elementos con las 26 letras del vocabulario inglés (de la letra 'a', con valor ascii 97, a la letra 'z', con valor ascii 122, sin almacenar la 'ñ').

La función principal llama a esa función y le pasa un vector como argumento, en el cual quedan almacenadas las letras. Después, la función principal genera 8 números aleatorios entre 0 y 25. Cada uno de ellos corresponde con una letra, la cual forma una contraseña de 8 letras minúsculas que se guarda en un vector de datos de tipo char y tamaño 9.

Por ejemplo, siendo el vector (a,b,c,d,e,f...z), y generando los aleatorios 3,2,4,1,0,4,25,24 la contraseña sería *dcebaezy*.

Compílalo con el compilador `gcc`. Después, ejecútalo para comprobar que funciona correctamente.

Para imprimir un vector de char, no es necesario usar un bucle, se puede imprimir todo el vector almacenando el carácter '\0' al final del vector (en la posición siguiente a la última letra, en este caso, en la posición 8) y usando la instrucción printf(" %s", vector_char);

### 4.2. Ejercicio 2

Realice un programa `identidad.c` en lenguaje C cuya función principal llame a una función que inicializa la matriz que se le pasa como argumento, cuyo tamaño es 9x9, con la matriz identidad (en la diagonal 1s y 0s en el resto). La función principal imprimirá la matriz una vez inicializada por la función. Hay que utilizar bucles. Compílalo con el compilador `gcc`. Después, ejecútalo para comprobar que funciona correctamente.

## 5. Ejercicios extra

### 5.1. Ejercicio 1

Realice un programa `matriz.c` en lenguaje C cuya función principal llame a una función que inicializa la matriz que se le pasa como argumento, cuyo tamaño es 5x5, con números aleatorios entre 3 y 15. La función principal imprimirá la matriz una vez inicializada por la función. Compílalo con el compilador `gcc`. Después, ejecútalo para comprobar que funciona correctamente.

### 5.2. Ejercicio 2

Realice un programa `suma.c` en lenguaje C cuya función principal llame a una función que inicializa la matriz que se le pasa como argumento, cuyo tamaño es 6x6, con números aleatorios entre 1 y 10. Luego la función principal llamará a una función `suma` que recibe una matriz como argumento y devuelve la suma de todos los elementos de la matriz. La función principal imprimirá la matriz una vez inicializada por la función y después el resultado devuelto por la función `suma`. Compílalo con el compilador `gcc`. Después, ejecútalo para comprobar que funciona correctamente.

### 5.3. Ejercicio 3

Realice un programa `sumaColumnas.c` en lenguaje C cuya función principal llame a una función que inicializa la matriz que se le pasa como argumento, cuyo tamaño es 3x6, con números aleatorios entre 1 y 10. Luego la función principal llamará a una función `sumaColumnas` que recibe una matriz y un vector como argumento y devuelve la suma de todos los elementos de cada columna de la matriz. La función principal imprimirá la matriz una vez inicializada por la función y después el resultado de la suma de las columnas. Compílalo con el compilador `gcc`. Después, ejecútalo para comprobar que funciona correctamente.

### 5.4. Ejercicio 4

Realice un programa `producto.c` en lenguaje C que calcule el producto de 2 matrices 3x3. Para ello tiene que usar una función `inicializa` que inicializa

la matriz que se le pasa como argumento, cuyo tamaño es 3x3, con números aleatorios entre 1 y 5 (se llamará 2 veces a la función para inicializar las 2 matrices). Luego la función principal llamará a una función `producto` que recibe 3 matrices como argumento y calcula el producto de dos de ellas almacenando el resultado en la tercera. La función principal imprimirá las matrices una vez inicializadas por la función y después la matriz resutado del producto. Para ello usará la función `imprime` que dada una matriz 3x3 la imprime por pantalla. Compílalo con el compilador `gcc`. Después, ejecútalo para comprobar que funciona correctamente.