# Trabajo práctico propuesto

**Objetivos del trabajo:**

- Adquirir la capacidad de programar aplicaciones algo complejas en lenguaje C.
- Aplicar los conocimientos adquiridos en las clases teóricas y prácticas a un ejemplo práctico.
- Adquirir la capacidad de estructurar problemas complejos para facilitar su resolución.

## 1. Introducción

Como trabajo para preparar el examen de la asignatura se implementará un programa que permita simular el juego del bingo.

## 2. Indicaciones

- El trabajo es individual.
- Se recomienda realizarlo para superar la asignatura.
- Se recomienda utilizar el foro de dudas del moodle para resolver los problemas que os vayan surgiendo. Como no cuenta para nota, vamos a usar el foro aunque envieis código para que os asesoremos.
- Copiar no os sirve para aprender. Tenéis que intentarlo vosotros de forma individual. Todos los profesores de la asignatura estamos a vuestra disposición para ayudaros a conseguirlo.
- Tendréis un assignment en github habilitada para subir código. Podéis ir subiéndolo, actualizando el archivo tantas veces como queráis y, si nos escribís, os echaremos un vistazo para deciros cómo lo tenéis, posibles fallos, posibles mejoras, etc.

## 3. Desarrollo del juego

El juego consiste en implementar el juego del bingo. Inicialmente se ha de mostrar un menú que permita al usuario elegir entre "1.- Jugar contra el ordenador", "2. - Jugar contra otro usuario" o "3.- Salir del programa". Si la opción

elegida no es correcta, se mostrará un mensaje de error y se volverá a mostrar el menú. Sólo se saldrá cuando la opción elegida sea "Salir del programa".

Si la opción elegida es "Jugar contra el ordenador", lo primero que se pedirá al usuario es que indique el tamaño de filas y de columnas de cartón. Además, se leerá un entero positivo menor que el número de columnas: indicará cuántas posiciones blancas habrá en cada fila. En las posiciones restantes se almacerán números aleatorios no repetidos de 1 a 99 ordenados por columnas de menor a mayor.

Si la opción elegida es "Jugar contra otro usuario", lo primero que se pedirá al usuario es que indique el tamaño de filas y de columnas de cartón. Además, se leerá un entero positivo menor que el número de columnas: indicará cuántas posiciones blancas habrá en cada fila. En las posiciones restantes se almacerán los números que indique el usuario. Dichos números tiene que comprobarse que están entre 1 y 99 y que no están repetidos. Además, tienen que almacenarse en el cartón ordenados por columnas de menor a mayor.

Una vez generados los cartones, se imprimirán por pantalla, indicando cada cartón a qué jugador pertenece. A continuación, comenzará el juego del bingo. El desarrollo del juego es el siguiente: se generará un número aleatorio de 1 a 99, se mostrará por pantalla el número generado (no puede haber salido antes). Además, se comprobará si está en alguno de los cartones. En caso de que esté, se guardará dicha información en el cartón. Cuando hayan salido todos los números que aparecen en una fila de un cartón, se indicará por pantalla "El Jugador X ha cantado FILA", especificará qué jugador lo ha cantado (el 1 o el 2). Además, se imprimirá el cartón de dicho usuario en el estado actual: en las posiciones donde haya un blanco aparecerá un símbolo X, en las que haya un número se imprimirá el número, si no ha salido, y el símbolo $ si dicho número ya ha salido.

A continuación, se seguirán generando números aleatorios sin repetición de 1 a 99. Cuando hayan salido todos los números del cartón, se indicará "BINGO", especificando por pantalla qué jugador ha cantado bingo. Se mostrará por pantalla el cartón del jugador que ha perdido en su estado actual (con los blancos ó X, los números que sí han salido con $ y los que no han salido) y el cartón del usuario que ha ganado. Después, se mostrará de nuevo el menú inicial.

## 3.1. Cartones

1. El programa tendrá 2 punteros a enteros. Uno para cada jugador. En ellos se almacena información relacionada con los números del cartón con los que juega el jugador y las posiciones en blanco.

2. Se puede utilizar un código numérico para representar los distintos valores en cada tabla. Por ejemplo, -1 indica que hay un blanco en esa posición, 0 indica que esa posición había un número y ya salió su bola, un número de 1 a 99 indica que hay un número pero todavía no ha salido la bola. Para imprimirlo, por ejemplo, se puede hacer lo siguiente: si el valor es -1 se imprimirá una X, si es 0 un símbolo $, si no, el número almacenado.

3. Las tablas serán matrices bidimensionales de nf x nc posiciones. Hay que utilizar punteros.

4. Cada fila del cartón tiene que tener el número de posiciones en blanco que indique el usuario (sin número).

5. Los números que pueden aparecer en un cartón van de 1 a 99 *inclusive* y no pueden estar repetidos.

6. Dos jugadores pueden tener los mismos números en sus cartones.

7. Los números del cartón han de estar ordenados de menor a mayor por columnas. Es decir, el número que aparezca en la posición (0,0) ha de ser menor que el que aparezca en la posición (1,0) y así sucesivamente (el que esté en la posición (5,2) ha de ser menor que el de la posición (6,0), etc).

## 3.2. Reglas del juego

Inicialmente se mostrará un menú con las siguientes opciones:

1. Jugar (ordenador vs ordenador).

2. Jugar (jugador vs jugador).

3. Salir.

El usuario podrá elegir una de esas 3 opciones.

### 3.2.1. Opción 1. Jugar (ordenador vs ordenador).

Si se elige la opción 1, se deberán generar dos cartones de forma automática. Para generar un cartón, se decidirá aleatoriamente qué posiciones del cartón van a tener una posición en blanco, teniendo en cuenta que habrá el número de blancos que indique el usuario por fila. Además, se rellenarán las posiciones "no blancas" con números aleatorios no repetidos de 1 a 99, ordenados de menor a mayor por columnas.

Una vez que se han inicializado los cartones, se imprimen por pantalla y comienza el juego.

### 3.2.2. Opción 2. Jugar (jugador vs jugador).

Si se elige la opción 2, se deberán generar dos cartones de forma manual. Para generar un cartón, el usuario indicará en qué posiciones del cartón va a haber una posición en blanco. Además, se rellenarán las posiciones "no blancas" con los números que el usuario indique, no repetidos y 1 a 99, ordenados de menor a mayor por columnas.

Una vez que se han inicializado los cartones, se imprimen por pantalla y comienza el juego.

### 3.2.3. Opción 3. Salir.

Al elegir esta opción, finaliza la ejecución del programa.

# 4. Funciones

En el código deben aparecer como mínimo las siguientes funciones:

- Función main:

  - ¿Qué hace? Debe abrir un archivo y comprueba si hay un error en la apertura. Si lo hay, muestra un mensaje de error, si no, muestra el menú mientras la opción seleccionada no sea "Salir del programa". Antes de finalizar el programa, cierra el archivo.

- Función menú:
  - Devuelve: la opción leída
  - Recibe: nada
  - ¿Qué hace? Muestra un menú por pantalla y lee una opción, comprobando si es correcta (valor entre 1 y 3). Si no es correcta, muestra un mensaje de error e imprime de nuevo el menú por pantalla hasta que la opción leída sea correcta. La función devuelve la opción leída.
- Función juego_bingo:
  - Devuelve: nada
  - Recibe: la opción elegida
  - ¿Qué hace? Se lee el tamaño del cartón (número de filas y número de columnas) y el número de blancos por fila. Si los valores indicados son negativos o 0 o si el número de blancos es negativo o mayor que el número de columnas, se muestra un mensaje de error y se vuelven a leer hasta que sean correctos. Según el argumento (la opción de menú que se ha leído por teclado) se juega al bingo automáticamente o manualmente, indicando además el tamaño de cartón y el número de blancos por fila.
- Función jugar_ordenador:
  - Devuelve: nada
  - Recibe: el número de filas del cartón, el número de columnas del cartón y el número de blancos en cada fila.
  - ¿Qué hace? Reserva espacio para dos cartones del tamaño apropiado y los inicializa automáticamente. A continuación, los imprime por pantalla. Por último, juega al bingo con esos cartones.
- Función jugar_usuarios:
  - Devuelve: nada
  - Recibe: el número de filas del cartón, el número de columnas del cartón y el número de blancos en cada fila.
  - ¿Qué hace? Reserva espacio para dos cartones del tamaño apropiado y los inicializa manualmente. A continuación, los imprime por pantalla. Por último, juega al bingo con esos cartones.
- Función generarTableroAutomatico:
  - Devuelve: nada
  - Recibe: el cartón, el número de filas del cartón, el número de columnas del cartón y el número de blancos en cada fila.

  - ¿Qué hace? Esta función inicializa el cartón a 0, pone el número de blancos que indique el argumento en cada fila del cartón de forma aleatoria y por último, rellena las posiciones restantes del cartón con números aleatorios de 1 a 99 no repetidos ordenados de menor a mayor.

- Función generarTablero:
  - Devuelve: nada
  - Recibe: el cartón, el número de filas del cartón, el número de columnas del cartón y el número de blancos en cada fila.
  - ¿Qué hace? Esta función inicializa el cartón a 0, pone el número de blancos que indique el argumento en cada fila del cartón en las posiciones que indique el usuario (comprobando que son válidas) y por último, rellena las posiciones restantes del cartón con los números que indique el usuario, comprobando que no se repitan y que estén de 1 a 99 ordenados de menor a mayor.

- Función pone_blancos_carton:
  - Devuelve: nada
  - Recibe: el cartón, el número de filas del cartón, el número de columnas del cartón y el número de blancos en cada fila.
  - ¿Qué hace? Esta función genera aleatoriamente las posiciones donde irá un "blanco" en cada fila del cartón. Como resultado el cartón tendrá en cada fila los blancos especificados por el número pasado como argumento.

- Función pone_blancos_carton_manual:
  - Devuelve: nada
  - Recibe: el cartón, el número de filas del cartón, el número de columnas del cartón y el número de blancos en cada fila.
  - ¿Qué hace? Esta función lee las posiciones donde irá un "blanco" (tantas como indica el argumento) indicadas por el usuario. La función ha de comprobar que las posiciones indicadas por el usuario son válidas.

- Función pone_nums_carton:
  - Devuelve: nada
  - Recibe: el cartón, el número de filas del cartón, el número de columnas del cartón y el número de blancos en cada fila.
  - ¿Qué hace? Genera números aleatorios distintos de 1 a 99 y los coloca en el cartón ordenados de menor a mayor por columnas donde no haya blancos. Para ello usamos un vector de tamaño igual al número de posiciones del cartón que no tienen blanco. Se inicializa a 0 el vector y luego se rellena de números aleatorios de 1 a 99 no repetidos. Después se ordena y finalmente se almacena en el cartón.

- Función pone_nums_carton_manual:

- Devuelve: nada
- Recibe: el cartón, el número de filas del cartón, el número de columnas del cartón y el número de blancos en cada fila.
- ¿Qué hace? Lee números distintos de 1 a 99 y los coloca en el cartón ordenados de menor a mayor por columnas donde no haya blancos. Para ello usamos un vector de tamaño igual al número de posiciones del cartón que no tienen blanco. Se inicializa a 0 el vector y luego se rellena de con números indicados por el usuario de 1 a 99 no repetidos. Después se ordena y finalmente se almacena en el cartón.

– Función comprueba_num_en_vector:

- Devuelve: un valor si el número está en el vector y otro en caso contrario
- Recibe: un vector, su tamaño y un número.
- ¿Qué hace? Se comprueba si el número está en el vector. Devuelve un valor si el número está y otro en caso contrario.

– Función ordena_vector:

- Devuelve: nada
- Recibe: un vector y su tamaño
- ¿Qué hace? La función ordena los números de dicho vector de menor a mayor.

– Función imprime_carton:

- Devuelve: nada
- Recibe: el cartón y su tamaño
- ¿Qué hace? La función imprime el cartón por pantalla distinguiendo las posiciones donde hay números que no han salido, de los que sí han salido y de las posiciones en blanco. Por ejemplo, se puede hacer lo siguiente: en las posiciones donde hay un blanco se imprime una X. En las posiciones donde hay un número se imprime el número. En las posiciones donde hay un número que ya ha salido, se imprime un $. El cartón se imprime de forma organizada (cada fila ocupa una línea y los valores que están en la misma columna, están alineados).

– Función jugar:

- Devuelve: nada.
- Recibe: dos cartones y su tamaño.
- ¿Qué hace? Primero inicializa un vector de 99 enteros (las bolas que salen) a 0. Después, mientras no haya fila, genera un número aleatorio de 1 a 99 sin repeticiones y se imprime por pantalla. A continuación, se comprueba si ese número está en los cartones. También ha de comprobar si se ha cantado fila (sólo se puede cantar una fila, por el primer jugador que lo haga) y si se ha cantado bingo. Si se canta fila, imprime por pantalla el cartón del jugador que ha cantado fila e

indica quién es. Una vez que se ha cantado fila, se siguen generando números aleatorios de 1 a 99 sin repeticiones, se siguen imprimiendo por pantalla y se comprueba si se canta bingo. Si se canta bingo, se imprime el cartón del jugador que ha perdido e indica quién ha ganado, acabando el juego (se vuelve a mostrar el menú).

- Función comprobar_ganador:
  - Devuelve: un valor en función de si dicho cartón tiene o no tiene bingo.
  - Recibe: un cartón y su tamaño.
  - ¿Qué hace? Devuelve un valor en función de si dicho el cartón pasado como argumento tiene o no tiene bingo.
- Función comprobar_fila:
  - Devuelve: un valor en función de si el cartón tiene una fila completa o no.
  - Recibe: un cartón y su tamaño.
  - ¿Qué hace? Devuelve un valor en función de si dicho cartón tiene o no tiene una fila completa (han salido todos los números de esa fila).
- Función comprueba_carton:
  - Devuelve: nada.
  - Recibe: un cartón y su tamaño.
  - ¿Qué hace? Si el cartón tiene el número, lo marca como que ha salido (puede almacenar un 0 en esa posición para indicar que ha salido).
- Función comprueba_si_ha_salido_bola:
  - Devuelve: un valor en función de si dicho número está o no está en el vector.
  - Recibe: un vector de 99 enteros y un número.
  - ¿Qué hace? Devuelve un valor en función de si dicho número está o no está en el vector.

# 5. Ejemplo

Visualización de nuestro cartón con los números y posiciones en blanco inicialmente:

```
X 12 38  X 55 73 82 X  X
2  X 39  X 58  X 85 X 94
7 32 46 52 X   X  X X 97
```

Las Xs significan posición en blanco. Los números están ordenados de menor a mayor por columnas.

Visualización del cartón con las bolas que han aparecido en un punto intermedio del juego:

```
X  $ 38  X  55 73 82 X  X
2  X  $  X  58  X 85 X 94
7 32 46  $  X   X  X X 97
```

Los símbolos $ significan que ese número ya ha salido.

Visualización del cartón cuando se ha cantado fila:

```
X  $  $ X  $   $  $ X  X
2  X  $ X  58  X 85 X 94
7 32 46 $  X   X  X X 97
```

En una fila sólo habrá símbolos $ ó X.

Visualización del cartón cuando se ha cantado bingo:

```
X  $  $  X  $  $  $  X  X
$  X  $  X  $  X  $  X  $
$  $  $  $  X  X  X  X  $
```

En una fila sólo habrá símbolos $ ó X.