Let's Bingo!

**Descripción del juego**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 |  |  |  | 6 |  |  |  | 10 | 11 |  |  |
| 14 |  |  | 17 |  |  | 20 |  |  |  |  | 25 |  |
|  |  | 29 | 30 | 31 |  |  |  |  |  |  |  | 39 |

Un tablero o cartón de Bingo, como se ve en la imagen, posee algunos casilleros con números y otros casilleros vacíos. La distribución de números y casilleros vacíos es distinta en cada tablero, con lo cual no existen dos tableros iguales. En nuestra versión del juego, cada tablero tiene capacidad para 100 casilleros, y posee 15 casilleros con números (entre el 1 y el 100).



Al comenzar cada partida, se reparte 1 tablero para cada jugador. A continuación, se pone en movimiento el bolillero y comienzan a sacarse los números uno a uno. Cada vez que sale un número, los jugadores deben verificar si posee ese número en su tablero, y en ese caso deberá hacer una marca sobre el número, indicando que el número ha salido. El jugador que complete primero su tablero obtendrá tres puntos, el que complete segundo obtendrá dos puntos, y el tercero obtendrá un punto. En caso de empate, ambos recibirán los puntos. Por ejemplo, si dos jugadores completan sus tableros, y son los primeros en hacerlo, ambos recibirán los 3 puntos.

La persona que saca los números del bolillero los coloca en un espacio aparte, para asegurarse que no vuelvan a salir durante la partida. El juego finaliza cuando han surgido los ganadores (primero, segundo y tercer puesto).

## Para pensar… un bingo extendido

Podemos extender las reglas del juego, considerando que un juego consta de varias partidas, en las que cada jugador suma los puntos obtenidos al finalizar cada partida. En este bingo extendido, una vez finalizada la partida (al menos un jugador obtuvo 3 puntos, al menos un jugador obtuvo 2 puntos y al menos un jugador obtuvo 1 punto), los jugadores devuelven los tableros y se reparten nuevamente. Además, todas las bolillas vuelven al bolillero para la siguiente partida.

El juego finaliza cuando un jugador alcanza los 20 puntos.

**Actividad**

El objetivo de esta actividad es implementar en Pascal el juego de Bingo descrito arriba. Para facilitar la implementación, vamos a pensar el juego en 4 etapas, que se describen a continuación:

1. Inicialización del juego (para 10 jugadores):
   1. Carga de datos de los jugadores: nombre y número (1..10)
   2. Inicializar los contadores de puntos para cada jugador.
2. Comenzar una partida
   1. Inicializar el bolillero (poner todas las bolillas dentro)
   2. Repartir los tableros a los jugadores.
3. Jugar
   1. Obtener un número del bolillero
   2. Todos los jugadores verifican si lo tienen sus cartones, y hacen una marca en tal caso
   3. Cada jugador debe verificar si completó su cartón. En caso de hacerlo, recibirá los puntos correspondientes
   4. Volver al punto 3.a hasta que hayan surgido los 3 ganadores
   5. Volver al punto 2.a hasta que algún jugador gane el juego
4. Podio
   1. Informar el jugador que ha ganado el juego
   2. Informar en orden descendente las posiciones de los restantes jugadores



**Para tener en cuenta**

El juego posee algunas actividades aleatorias: la generación de cada tablero, la repartición de tableros entre los jugadores, la extracción de números del bolillero. Para simular esta aleatoriedad, vamos a utilizar la función Random(N) disponible en Pascal. Esta función recibe un parámetro de entrada N, y retornará un valor aleatorio entre 0 y N-1 . Además, para maximizar la aleatoriedad, debemos invocar al módulo Randomize cada vez que comienza el programa. Por ej:

**program** aleatorios;

**var**

num, cant : integer;

**begin**

Randomize; *//con esto generamos secuencias diferentes en cada ejecución*

cant := 0;

num := Random(100); //genero un número entre 0 y 99

**while** (num <> 20) **do** **begin**

cant := cant + 1;

num := Random(100); //*genero otro número*

**end**;

writeln('Tuve que generar ',cant, ' números hasta que apareció el 20');

**end**.



**Atención**

Una secuencia aleatoria puede generar un mismo número más de una vez. Por lo tanto, cada vez que se genera un valor aleatorio, debe verificarse que el mismo sea válido. Por ejemplo, si se obtiene un número del bolillero, deberá verificarse que dicho número no haya salido en esta partida. O si se reparten cartones entre los jugadores, debe verificar que un mismo cartón no vaya a dos jugadores en una misma partida.

**Para pensar entre todos:**

* Datos y estructuras de datos para representar:
  + Tableros
  + Jugadores y puntajes
  + El bolillero
* Generación de valores aleatorios correctos (descartando los repetidos)
* Detectar al ganador de cada partida

**Cómo vamos a resolver la actividad?**

* Presentamos la estructura general del programa
* armamos 3 grupos por aula:
  + grupo 1: Inicialización del juego y de cada partida,
  + grupo 2: jugar una partida
  + grupo 3: detectar que ya terminó el juego, y generar el podio
* En el encuentro de práctica del martes 26/5, los ayudantes presentan el problema, discuten estructuras de datos con los alumnos, arman los grupos y distribuyen las tareas entre los alumnos
* Un alumno de cada grupo deberá presentar, en el encuentro del viernes 29/5, una propuesta para resolver la tarea que se le asignó a su grupo.
* El ayudante evaluará la solución, y también seleccionará a otro grupo para que le diga qué le parece (esto no lo comunicamos antes, así nos aseguramos que todos estén presentes)

hector

1. Inicialización del juego (para 10 jugadores):

**procedure** cargarJugadores(var v : vectorJugadores);

* 1. Cargar datos de los jugadores: nombre y número (1..10)
  2. Inicializar los contadores de puntos para cada jugador.

1. Comenzar una partida:

**procedure** jugarPartida(var jugadores : vectorJugadores);

* 1. Inicializar el bolillero (poner todas las bolillas dentro) y repartir los tableros a los jugadores.

**procedure** inicializarBolillero(var v : vectorPartida; var numerosQueSalieron: TCarton);

1. Jugar:
   1. Obtener un número del bolillero

**function** obtenerNumeroNoRepetido(carton : TCarton) : integer;

* 1. Todos los jugadores verifican si lo tienen sus cartones, y hacen una marca en tal caso

**procedure** procesarNumeroBolilla(num : integer; var v : vectorPartida);

* 1. Cada jugador debe verificar si completó su cartón. En caso de hacerlo, recibirá los puntos correspondientes

**procedure** buscarGanadorPartida(var v : vectorPartida; var puntos: integer; var jugadores : vectorJugadores);

* 1. Volver al punto 3.a hasta que hayan surgido los 3 ganadores
  2. Volver al punto 2.a hasta que algún jugador gane el juego

1. Podio:

**procedure** imprimirPuntos(jugadores : vectorJugadores);

* 1. Informar el jugador que ha ganado el juego
  2. Informar en orden descendente las posiciones de los restantes jugadores