

Programación II  
Curso 2020-2021

*Entrega 1*

“Game of Life”

Daniel Aramburu, Jorge Salas, Marc Pastor

NIA: 100430509

Madrid, 17 de octubre de 2020

Índice

[1. Estructuras de datos utilizadas 4](#_Toc53870402)

[2. Estructuras de programación utilizadas 4](#_Toc53870403)

[4. Funciones 5](#_Toc53870404)

# **1. Estructuras de datos utilizadas**

Las principales estructuras de datos que hemos utilizado son:

1. **Vectores**: hemos utilizado vectores principalmente para la entrada por teclado de los datos por parte del usuario. Mediante el uso *readline()* obteníamos un vector de *characters* , que posteriormente convertíamos a vector numérico, en caso que *grepl()* detectara dígitos enteros (aunque en formato carácter). También hemos usado muchos vectores de carácter lógico (*boolean*), para crear bucles *while* y condiciones que aseguran el buen funcionamiento del programa y detectan los errores del usuario al insertar los valores que se le piden.
2. **Matrices**: Hemos utilizado matrices para la creación de los tableros (tanto el inicial como los sucesivos). El motivo de su uso ha sido que el juego solo trata con un tipo de valores (en nuestro caso caracteres), y las matrices son la estructura óptima cuando se trata de datos que comparten clase (y tienen 2 dimensiones), ya que nos permite extender las propiedades de los vectores a 2 dimensiones. En el caso que tuviéramos que introducir tipos de datos distintos (por ejemplo números y caracteres), habríamos optado por el *data.frame()*, pero al no darse el caso, hemos creído que la matriz era más adecuada. En nuestro código hay 2 matrices: *grid\_gen\_0* (tablero 0) y *grid\_gen\_0* (tablero 1), sus dimensiones son las mismas y las especifica el usuario.

# **2. Estructuras de programación utilizadas**

Las principales estructuras de programación que hemos usado son:

1. **Bucles**: Hemos utilizado bucles (principalmente *whiles*) para volver a pedir por teclado los datos al usuario, en caso de que los valores que ha introducido sean incorrectos y también para seguir en el menú de opciones, mientras el usuario no seleccione la opción de “Terminar”. El motivo es que no conocemos el número de veces que el usuario se equivocará al introducir los datos, o el número de células vivas que quiere añadir o eliminar. También hemos usado bucles *for()* principalmente para localizar a una célula y a sus vecinas (iterando por filas y columnas el tablero).
2. **Condicionales**: Esta ha sido la estructura de programación que más hemos usado. La hemos empleado para decidir si el programa debe continuar con su ejecución (cuando los datos introducidos son correctos) o meterse dentro de un bucle hasta que el usuario los introduzca correctamente, así como para determinar si una célula es vecina de otra, o si está viva o muerta.
3. **Funciones**: hemos utilizado muchas funciones base de R, para facilitarnos el trabajo como: *strsplit(), readline(), grepl(), as.integer(), nrow(), ncol(),* etc. Además hemos creado 2 funciones propias, en las que se basa el funcionamiento del programa.

# **4. Funciones**

Hemos creado dos funciones para diseñar nuestro programa:

1. ***game\_set\_up()*** : Esta función es la que sustenta el programa. Nos permite pedir los datos necesarios al usuario, establecer las condiciones de como deben ser dichos datos (y en caso de que no sean correctos avisar al jugador), para posteriormente crear el tablero inicial y añadir/eliminar células vivas. Esta función contiene muchas variables, que principalmente se dividen en vectores de tipo *carácter ¸* mediante los que guardamos los datos que introduce el usuario, vectores lógicos para aplicar bucles y condiciones y matrices para crear las tablas. No depende de parámetros, sino que usa datos introducidos por teclado y al final de su ejecución nos devuelve el tablero inicial que hemos creado.
2. ***crear\_tabla\_1()*** : Esta función coge como input el tablero inicial generado por *game\_set\_up(),* al cual le aplica las reglas de reproducción, soledad, supervivencia y sobrepoblación, para luego devolver como output el tablero de la generación 1 en una ventana nueva.