Ejercicio Abstracción
El juego Buscaminas

El objetivo de este ejercicio es que el estudiante se familiarice con el proceso de documentar código y aprender a crear y usar clases plantillas.

EL JUEGO BUSCAMINAS

En esta sección se explicará la lógica del juego. El juego se plantea en un tablero con un número de filas 9 y columnas 9. Las casillas pueden ocultar minas esparcidas por todo el tablero. Es decir algunas casillas no ocultan nada y otras ocultan minas. El objetivo del jugador es limpiar el tablero de minas sin detonar ninguna mina. Para tal fin, el jugador tiene información o pistas sobre el número de minas vecinas en las casillas circundantes a una dada.

Las casillas tienen tres estados:

- Sin abrir. Una casilla sin abrir está en blanco y se puede hacer clic sobre ella.
- Abiertas. Las casillas abiertas, sabemos su estado. Quedan expuestas.
- Marcadas. Son aquellas marcadas por el jugador para indicar una posible ubicación de una mina

LÓGICA DEL JUEGO

El jugador selecciona una casilla para abrirla. En la primera jugada se le indica al jugador una casilla que es segura para abrir (ver la Figura 1). A lo largo del juego, si el jugador abre una casilla minada, la partida termina (ver la Figura 3). En otro caso, la celda queda abierta y muestra un número que indica la cantidad de minas circundantes (adyacentes a ella máximo 8); o una casilla en blanco (tiene un 0) y todas las casillas adyacentes no minadas a ella se abrirán automáticamente (es decir las 8 adyacentes). Esto se puede observar en la Figura 2. Los jugadores también pueden marcar una casilla (en la Figura 3 se muestran con una 'F'), lo cual se visualiza colocando una bandera para indicar que creen que hay una mina en ese lugar. Las casillas marcadas todavía se consideran sin abrir y un jugador puede hacer clic en ellas para abrirlas.

Al jugador se le da el número de minas y el numero de casillas que les queda por abrir.

Para ganar una partida, todas las casillas que no sean minas deben abrirse sin abrir una mina (ver Figura 5). Un tablero predefinido podría ser un 9x9 que contiene 10 minas (el que vamos a programar). Tableros más avanzados podrían ser de 16x16 con 40 minas o 30x16 con 99 minas. Una configuración de un posible tablero se puede ver en la Figura 4.

Para más información sobre estrategias y patrones de juego podéis consultar este link.

Rosa Mª Rodríguez Sánchez

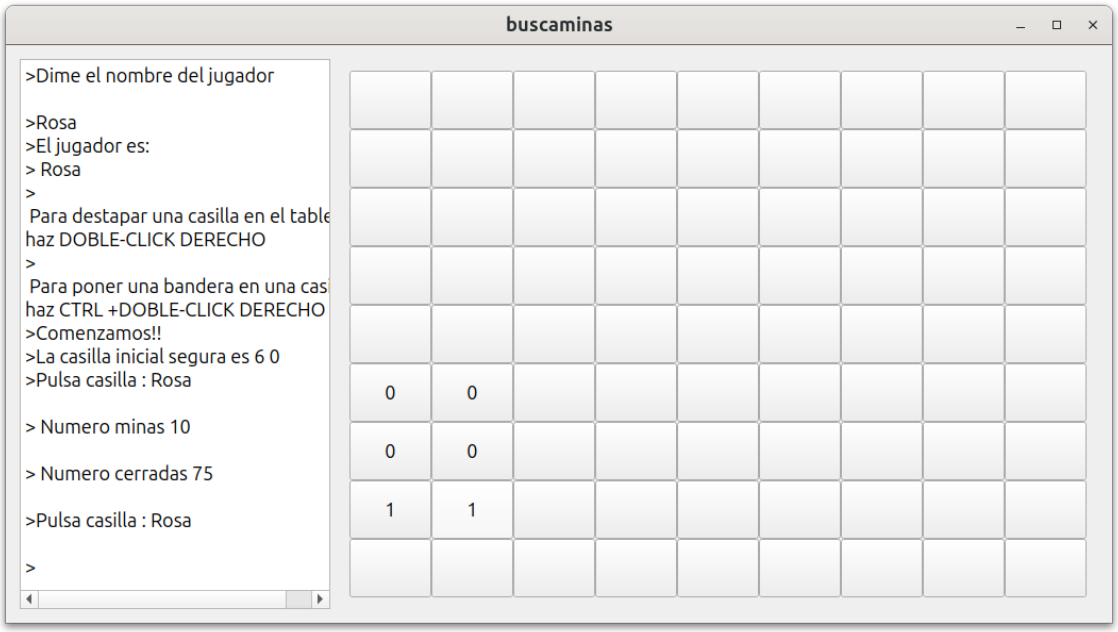

Figura 1: Inicio del juego

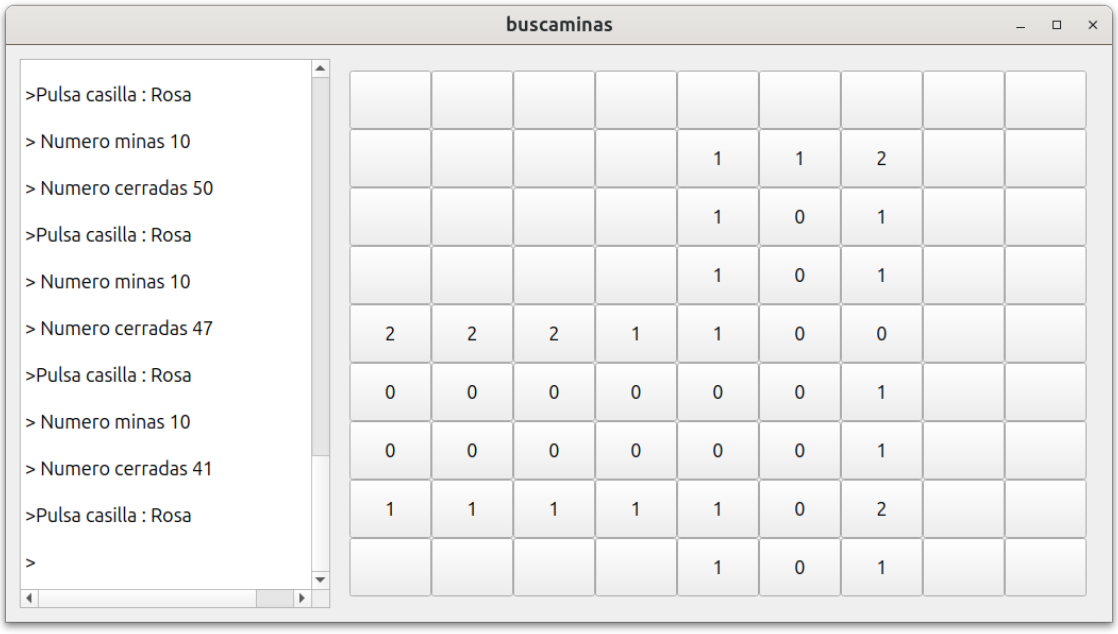

Figura 2: Evolucion del Juego Buscaminas.

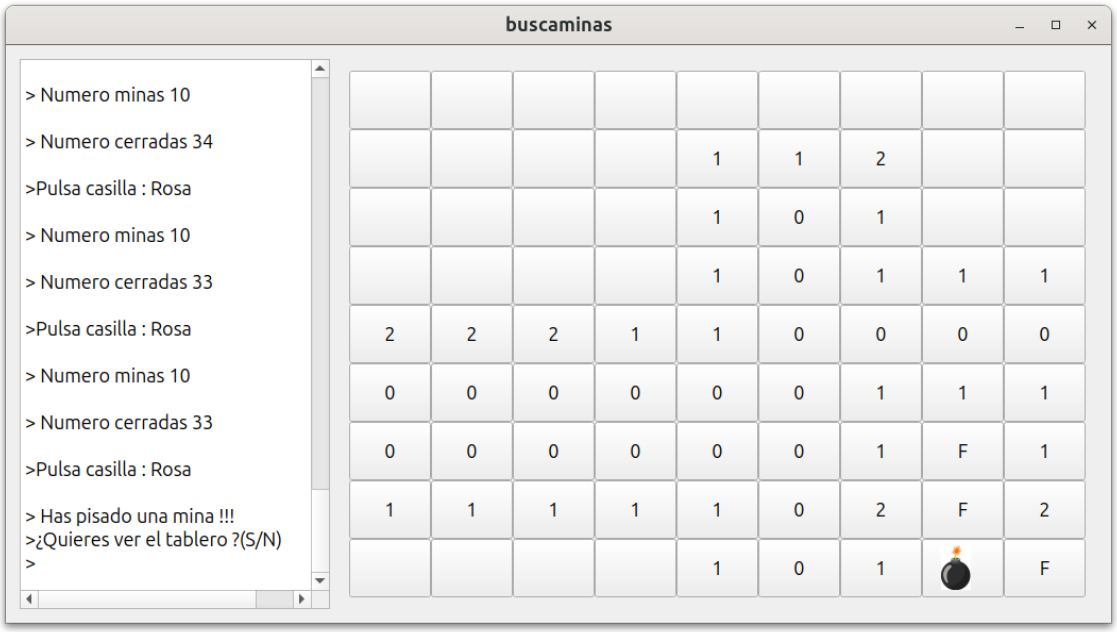

Figura 3: Al destapar una mina el juego termina

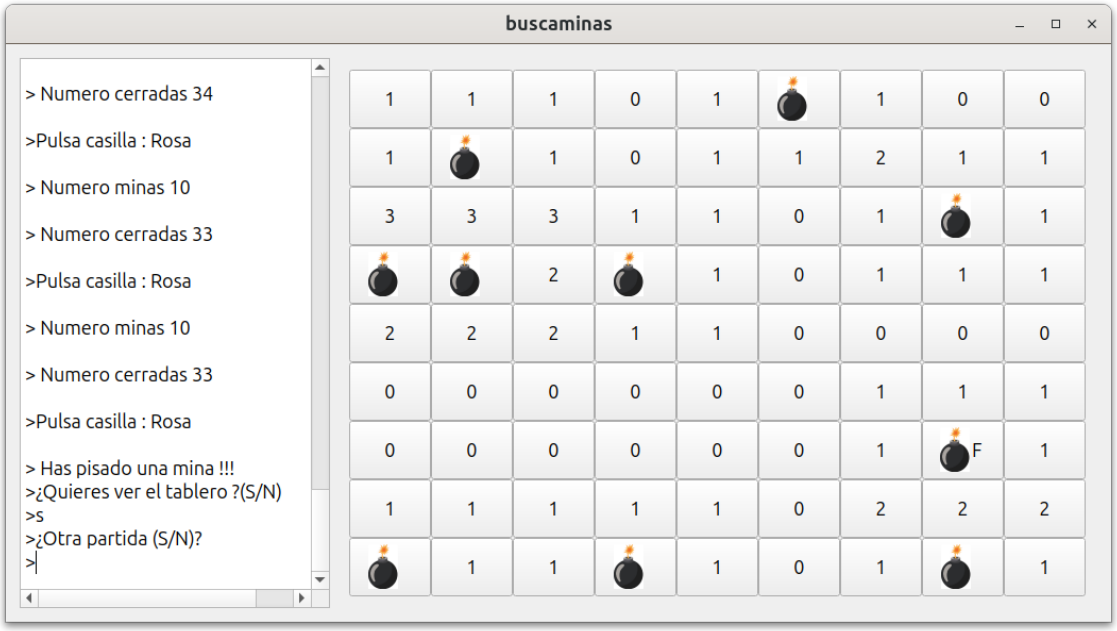

Figura 4: Tablero con todas las casillas descubiertas

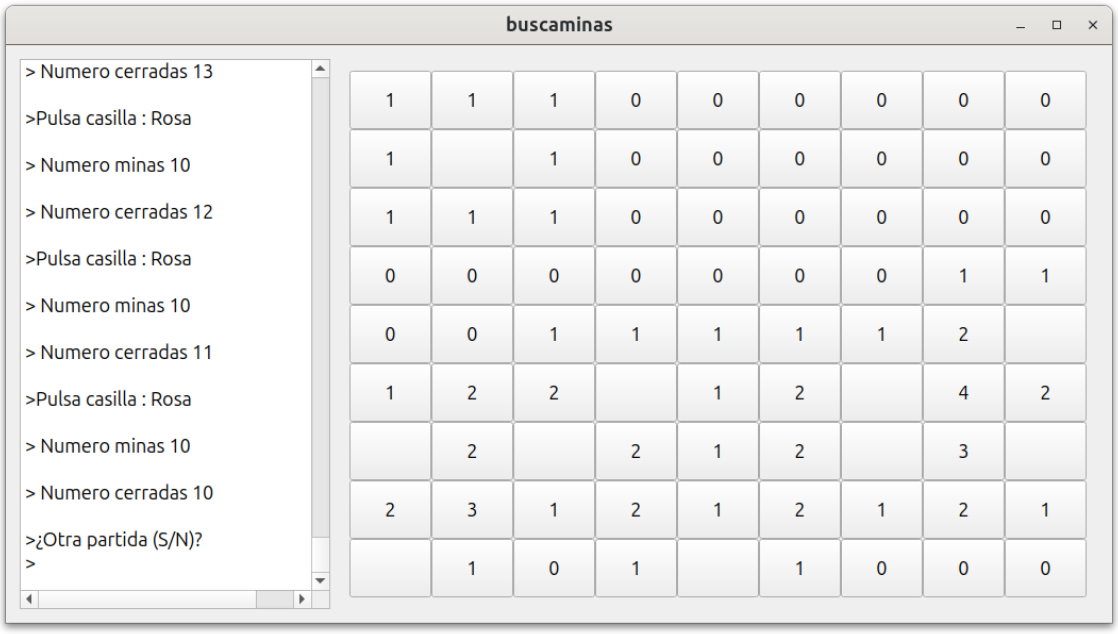

Figura 5: Ejemplo de una partida en la que el jugador descubre todas las minas

DISEÑO

Los módulos que se ha implementado son los siguientes:

- Tablero: contiene la información del tablero. La reprsentación de tablero incluye cuatro matrices de n filas y m columnas. Las matrices en concreto son:
  - **tab**: es una matriz de enteros que contiene la información de donde se encuentran la minas. Si tab[i][j]==-1 en la casilla (i,j) existe una mina en caso contrario tab[i][j]=0
  - **marcadas**: es una matriz de bool indicando si el usuario ha insertado una marca o no. Asi marcadas[i][j]=true si el usuario ha colocado una marca (en el juego se vera una F). En caso contrario marcadas[i][j]=false
  - **entorno**: es una matriz de enteros que indica para cada posición (i,j) el numero de bombas que existe en su 8 entorno.
  - **cerradas**: es una matriz de bool indicado si la casilla está cerrada (true, no ha sido destapada por el jugador) o si esta abierta (false).

```
//Representación de Tablero
class Tablero {

  private:
    int **tab;      //tablero que indica si existe mina (valor -1) o no (valor 0)
    bool ** cerradas; //tablero indicando si la casilla esta abierta o cerrada
    bool **marcadas ; //tablero para poner banderas en las casillas.True puesta
              //false no puesta
    int ** entorno ; //numero de minas en el entorno de cada casilla

    int numminas;  // Número de  minas
    int numfilas;  // Número de filas
    int numcols;           //Numero de columnas del tablero
    TableroGrafico *TG;   //puntero al tablero grafico
    int inicio_row,inicio_col; //Fila y Columna para iniciar el juego
```

- Jugador: contiene la información del jugador
- Juego: comunica el tablero con el jugador para realizar la dinámica del juego
- Ventana, Consola y Tablerografico: módulos que permiten programar el juego de modo gráfico.

EJERCICIO

Se pide que el estudiante cree la clase plantilla Matriz y que las matrices en Tablero sean redefinidas como instancias de la clase Matriz con los tipo base correspondientes. Además se pide que se documente este módulo y el módulo Tablero con doxygen. El estudiante debe tener en cuenta que deberá adaptar el código de Tablero para que se usen los métodos de Matriz donde corresponda.

Así el módulo matriz tendrá:

```
//Representación de Matriz fichero matriz.h
#ifndef __MATRIZ__H
#define __MATRIZ__H
template <class T>
 class Matriz{
  private:
   T** datos;
   int nf,nc;
  public:
    void Copiar(const Matriz<T> &M);//copia una matriz
    Void Libera();//libera la memoria
    Matriz();
    Matriz(int f,int c);
    Matriz(const Matriz<T> & M);
    ~Matriz();
    Matriz<T> & operator=(const Matriz<T> &M);
    T get(int f,in c)const;
    void set(int f,int c, const T &v);
    int getNumFilas()const;
    int getNumCols()const;
    template <class U>
    friend
    ostream & operator<<(ostream &os, const Matriz<U> &M);
};
#include “matriz.cpp”
#endif
```

Ahora el tablero quedaría como:

```
//Representación de Tablero
class Tablero {

  private:
    Matriz<int> tab; //almacena si existe (-1) o no mina (0)
    Matriz<bool> cerradas; //si la casilla esta cerrada (true) o no (false)
    Matriz<bool> marcadas ; //si la casilla contiene una marca o no
    Matriz<int> entorno ; //numero de minas en el entorno de cada casilla
    int numminas;         // Número de  minas

    TableroGrafico *TG;   //puntero al tablero grafico
    int inicio_row,inicio_col; //Fila y Columna para iniciar el juego
...
```