Manual del programador

Juego de la vida V1.0

Manual del programador

Módulos del programa

El programa se encuentra divido en 4 módulos diferentes y el modulo principal.
Cada modulo contiene funciones que se encargan de hacer operaciones similares.
Los módulos del programa son:

- OperacionesLogicas
  Este modulo tiene funciones lógicas y de cómputos matemáticos del programa.
- Estructuras
  Contiene funciones de inicialización de las estructuras que se usan en el programa. Las estructuras se encuentran definidas en el Header homónimo.
- Impresiones
  Todas las funciones que se encargan tanto de imprimir información por consola como de pedirle ingreso de datos al usuario se encuentran en este modulo
- Integracion
  En este modulo se lleva a cabo la integración de las funciones de los tres módulos anteriores. Tiene las funciones de inicialización del juego, el menú principal y la comunicación de las diferentes estructuras y funciones.
- Principal
  Solo contiene la función main del programa. No hay definiciones de funciones y hace llamada a la función juegoDeLaVida. Que se encarga de correr el juego.

Implementación del juego

Estructuras del juego

El juego posee tres estructuras principales. Declaradas en el header *estruturas.h*. Estas estructuras son:

1. Tablero
   Se compone de dos arrays de dos dimensiones, sus nombres respectivos son grilla y grillaInicial. Ambas de 20 arrays de 80 elementos cada uno y de tipo booleano. El propósito de grilla esta en guardar las posiciones en las cuales se encuentran las células vivas mientras que grillaInicial contiene las posiciones de las células vivas ingresadas por el

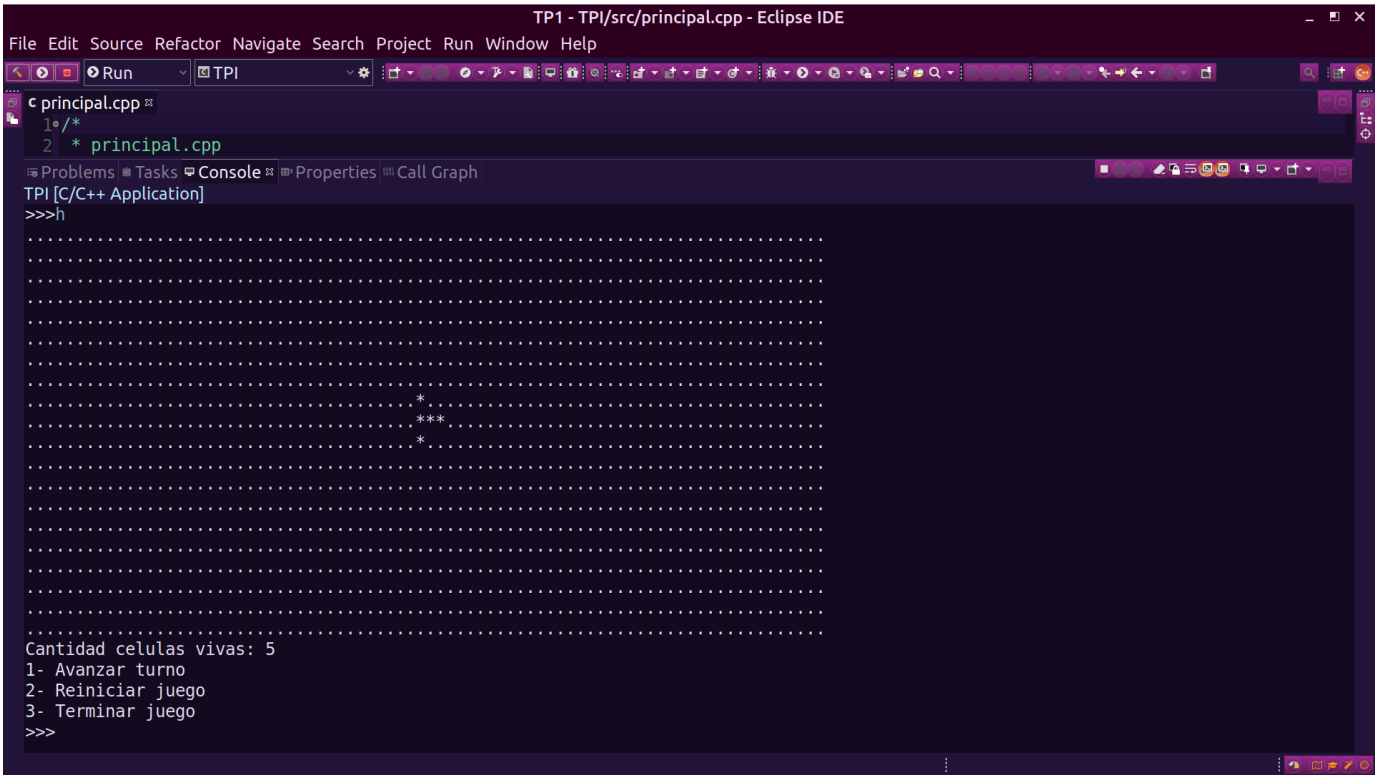
usuario, con el objetivo de poder guardar las mismas dado el caso en que el usuario desee reiniciar el juego.

2. InformacionJuego
   Posee el tablero, el total de muertes(unsigned int), el total de nacimientos(unsigned int) y la cantidad de turnos(unsigned int). Es la estructura que guarda, como su nombre lo indica, la información general del juego. las posiciones actuales e iniciales de las células vivas y las cantidades que se usan para calcular promedios a lo largo del juego.

3. EstadisticasTurno
   Esta estructura guarda información, siempre como unsigned int sobre, la cantidad de células vivas, la cantidad de muertes, la cantidad de nacimientos y la cantidad de cambios(nacimientos + muertes) que ocurren en un turno en particular.

Primer impresión de la grilla

Una vez finalizada la carga de datos por parte del usuario. El programa imprime la grilla e indica, la cantidad de células vivas que se encuentran actualmente en el tablero. Como se muestra a continuación:



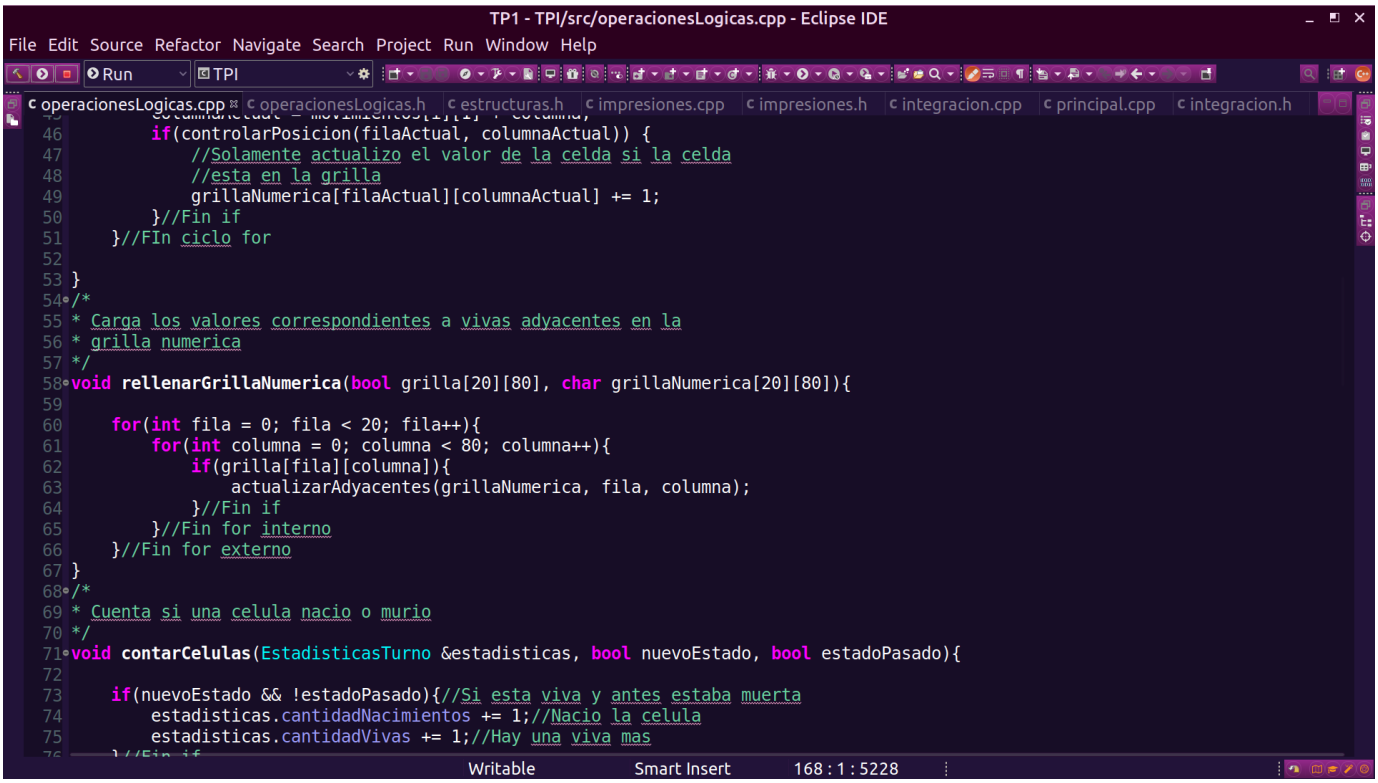
Actualización de la grilla

Luego de cada turno, se debe actualizar el tablero después de cada turno. El método mediante el cual se realiza dicha operación puede ser descrito como:

1. Se evaluá el estado de una célula
2. Si la célula se encuentra viva
   1. Le sumo a las células adyacentes
      uno

3. Si no se vuelve a *1.*

La función *rellenarGrillaNumerica* se muestra a continuación:



Para realizar dicho proceso, la función *actualizarGrilla* del modulo aritmética genera una grilla de 20 por 80 de caracteres(dado que ningun valor sera mayor a 80) llena de ceros. Se rellena según el proceso descripto en la función *rellenarGrillaNumerica* y posteriormente, según el estado de cada celula y siguiendo las reglas del juego, se actualiza la grilla del juego.

Durante este proceso, se realiza también la carga de la información del turno. Por ello, *actualizarGrilla* recibe un parámetro de *Tablero* como referencia y otro parámetro de *EstadisticasTurno* también como referencia.

Determinación del estado actual de una celula

Con lo obtenido luego de *rellenarGrillaNumerica* en *actualizarGrilla* se itera sobre la misma y se define el nuevo estado de la célula en la función *determinarCaso*.
Esta función recibe tres parámetros:

1. El estado actual de la célula
2. La cantidad de células vivas adyacentes(Se toma de *grillaNumerica*)
3. Las estadísticas del turno(se utilizan en la función *contarCelulas*, explicada a continuación)

La función *determinarCaso* evaluá el estado actual de la célula y le asigna a una variable booleana el estado futuro de la misma. Según las reglas del juego. Esto lo hace con una estructura if-else. Dado que hay dos posibilidades, que la célula este muerta o que la célula este viva.

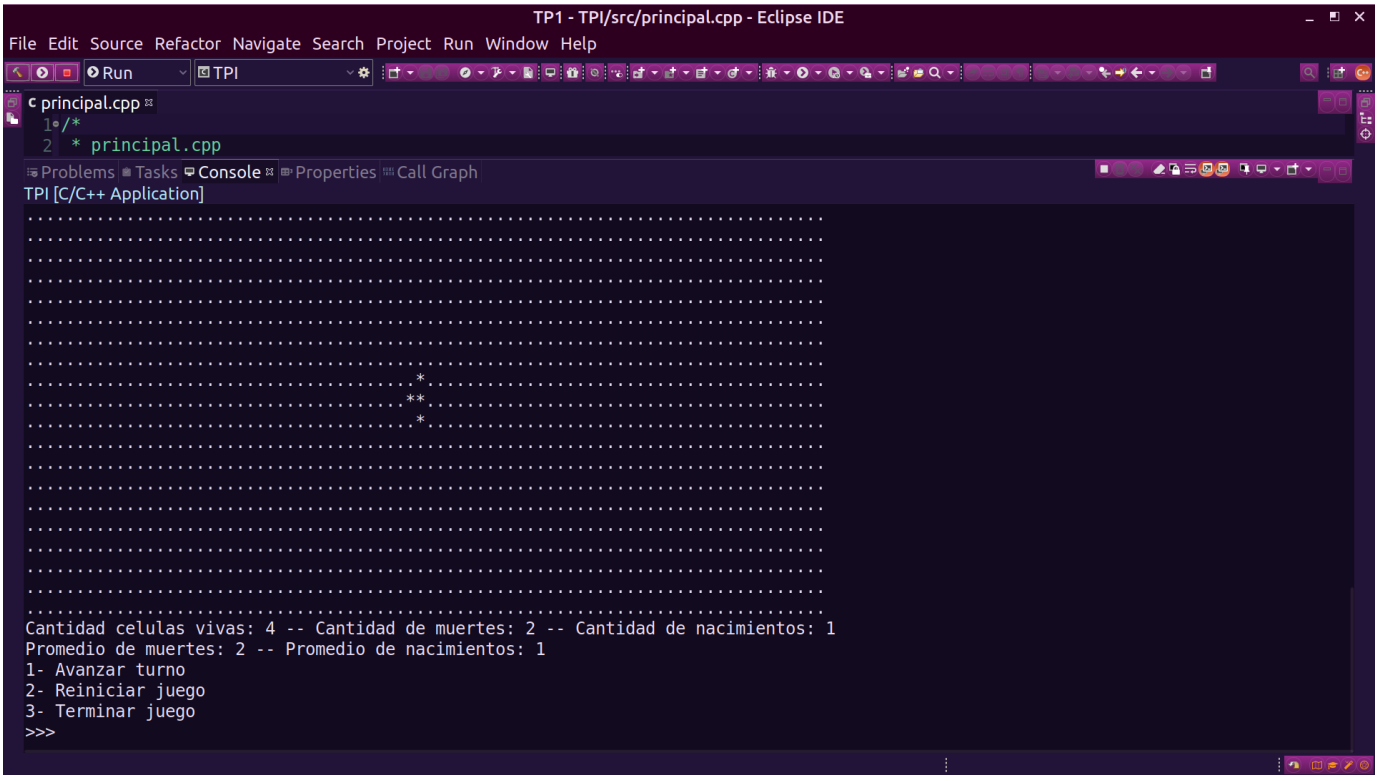
determinarCaso devuelve el estado futuro de la célula, y el mismo se le asigna a la célula correspondiente en la grilla(La cual es célula que estoy evaluando en ese momento de la iteración).

Calculo de información del turno

Al final de cada turno, el programa imprime junto con la grilla actualizada la siguiente información:

1. Cantidad de células vivas en el tablero
2. Cantidad de células que murieron en el turno anterior
3. Cantidad de células que nacieron en el turno anterior
4. Promedio de nacimiento de células histórico
5. Promedio de muertes de células histórico

La forma en la que se presenta la misma se muestra a continuación:



Para los puntos 1, 2 y 3. El programa lleva a cabo el conteo en paralelo con la actualización de la grilla. Esto se hace siguiendo la siguiente lógica en la función *contarCelulas*, que toma como parámetros. El estado actual de la célula y el estado anterior de la célula. De esta forma si:

1. La célula estaba viva y ahora esta muerta
    - Sumala a la cantidad de muertes
    - Restale a la cantidad de vivas
2. La célula estaba muerta y ahora esta viva
    - Sumala a la cantidad de nacimientos

- Restale a la cantidad de células vivas

Como se puede observar, la cantidad de células vivas se actualiza de forma dinámica. Por ende, luego del conteo inicial, según lo ingresado por el usuario. La cantidad de vivas se actualiza de esta manera.

El calculo de los promedios, los puntos 4 y 5. Se llevan a cabo luego de haberse imprimido por pantalla los primeros tres puntos. Previo a su calculo, en la función *actualizarValores*(que no devuelve nada) se actualizan:

1. La cantidad de turnos(se los incrementa en uno)
2. La cantidad de muertes(se le suman las muertes del turno actual)
3. La cantidad de nacimientos(ídem que para las muertes)

Ademas, los valores de cantidad de muertes y nacimientos del turno, se devuelven a 0. Para ello, la función toma como parámetros InformacionJuego y EstadisticasTurno, ambos pasados por referencia.

Aclaración sobre promedios

Dado que las células, se consideran entes enteros, no existen 0.5 células. Los promedios se calculan por medio de división entera. Que puede verse como un truncamiento del valor de la división de punto flotante. La función encargada de hacer este calculo, se llama *calcularPromedio* y toma como parámetros dos ints(por cantidad de turnos y por cantidad de muertes/nacimientos).

Implementación de la estabilidad en el tablero

Cuando pasan dos turnos seguidos durante los cuales no hubo ni muertes, ni nacimientos. Se le informa al usuario que el mismo hecho a ocurrido.
Para poder informar esto, se hace uso de la variable *cantidadCambios* de la estructura *EstadisticasTurno*. El procedimiento es el siguiente. Se evaluan la cantidad de cambios que hubo en un turno particular, sumando la cantidad de muertes y de nacimientos. Luego, esta cantidad se compara con la variable *cantidadCambios* que posee información del turno anterior. Si ambas son iguales a cero luego, la función *evaluarEstabilidad* devuelve el valor **true** indicando que durante dos turnos consecutivos no hubo cambios.
Esta misma función, luego de hacer la comparación descrita, le asigna a *cantidadCambios* el valor calculado para el turno actual. Por ende, *evaluarEstabilidad* toma como parametro una variable de tipo *EstadisticasTurno* pasada por referencia.

Implementación de reinicio de juego

Dado el caso en que el usuario decida reiniciar el juego a su estado inicial. Se reinicia la variable de Tablero, grilla. Copiándose los valores guardados en la variable de Tablero, grillaInicial.

Implementación de fin de juego

Cuando el usuario decide finalizar la ejecución del juego de la vida actual. Ingresando la opción correspondiente. El programa finaliza su ejecución.