

# Manual del programador

---

## Juego de la vida V1.0

---

## Manual del programador

---

### Modulos del programa

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

- OperacionesLogicas  
Este modulo tiene funciones logicas y de computos matematicos del programa.
- Estructuras  
Contiene funciones de inicializacion de las estructuras que se usan en el programa. Las estructuras se encuentran definidas en el Header homonimo.
- Impresiones  
Todas las funciones que se encargan tanto de imprimir informacion por consola como de pedirle ingreso de datos al usuario se encuentran en este modulo
- Integracion  
En este modulo se lleva a cabo la integracion de las funciones de los tres modulos anteriores. Tiene las funciones de inicializacion del juego, el menu principal y la comunicacion de las diferentes estructuras y funciones.
- Principal  
Solo contiene la funcion main del programa. No hay definiciones de funciones y hace llamada a la funcion juegoDeLaVida. Que se encarga de correr el juego.

### Implementacion del juego

#### Estructuras del juego

El juego posee tres estructuras principales. Declaradas en el header *estructuras.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 proposito de grilla esta en guardar las posiciones en las cuales se encuentran las celulas vivas mientras que grillaInicial contiene las posiciones de las celulas 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 informacion general del juego. las posiciones actuales e iniciales de las celulas vivas y las cantidades que se usan para calcular promedios a lo largo del juego.

### 3. EstadísticasTurno

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

## Primer impresion de la grilla

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

[illegible]

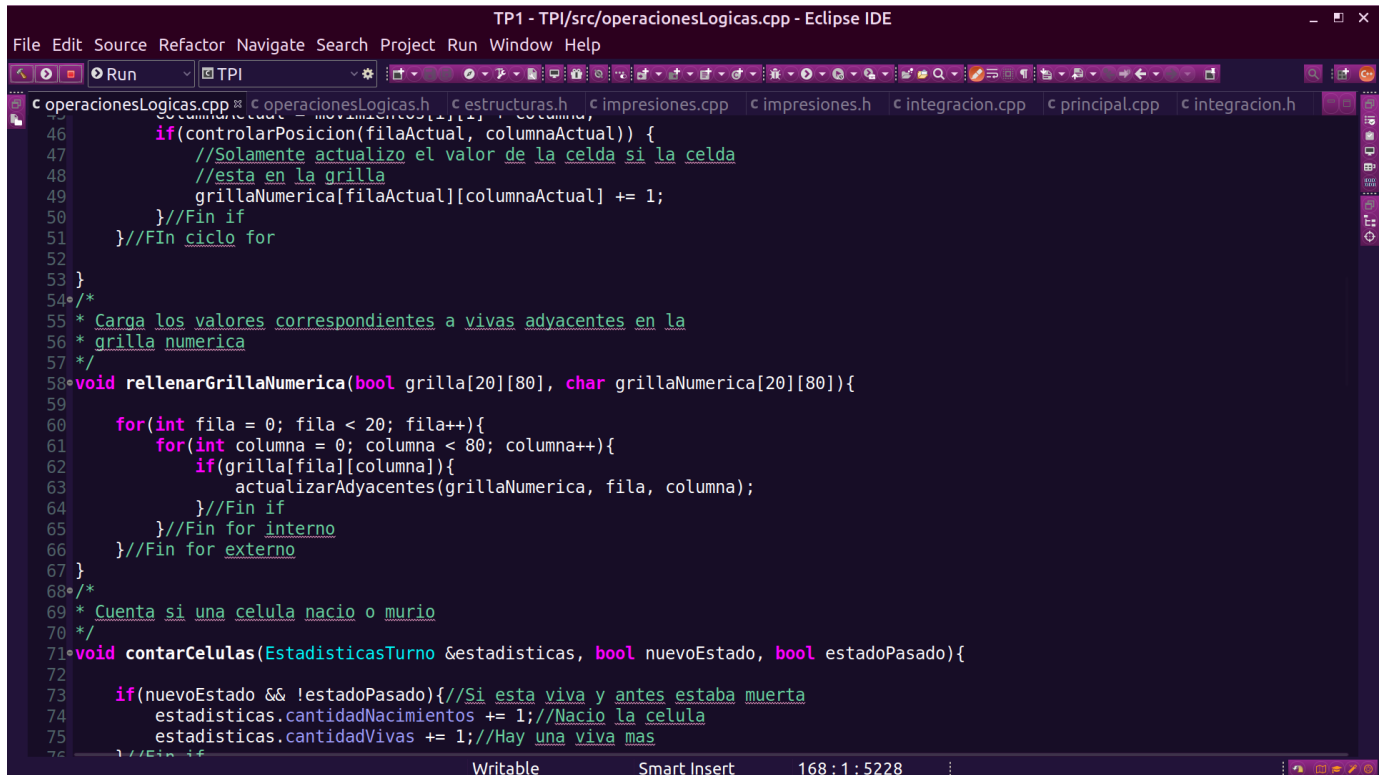
### Actualizacion de la grilla

Luego de cada turno, se debe actualizar el tablero despues de cada turno. El metodo mediante el cual se realiza dicha operacion puede ser descrito como:

1. Se evalua el estado de una celula
2. Si la celula se encuentra viva
  1. Le sumo a las celulas adyacentes uno

3. Si no se vuelve a 1.

La funcion *rellenarGrillaNumerica* se muestra a continuacion:



```
TP1 - TPI/src/operacionesLogicas.cpp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
TP1
operacionesLogicas.cpp | operacionesLogicas.h | estructuras.h | impresiones.cpp | impresiones.h | integracion.cpp | principal.cpp | integracion.h
46     if(controlarPosicion(filaActual, columnaActual)) {
47         //Solamente actualizo el valor de la celda si la celda
48         //esta en la grilla
49         grillaNumerica[filaActual][columnaActual] += 1;
50     } //Fin if
51 } //Fin ciclo for
52
53 }
54 /*
55  * Carga los valores correspondientes a vivas adyacentes en la
56  * grilla numerica
57  */
58 void rellenarGrillaNumerica(bool grilla[20][80], char grillaNumerica[20][80]){
59     for(int fila = 0; fila < 20; fila++){
60         for(int columna = 0; columna < 80; columna++){
61             if(grilla[fila][columna]){
62                 actualizarAdyacentes(grillaNumerica, fila, columna);
63             } //Fin if
64         } //Fin for interno
65     } //Fin for externo
66 }
67 }
68 /*
69  * Cuenta si una celula nacio o murio
70  */
71 void contarCelulas(EstadisticasTurno &estadisticas, bool nuevoEstado, bool estadoPasado){
72     if(nuevoEstado && !estadoPasado){ //Si esta viva y antes estaba muerta
73         estadisticas.cantidadNacimientos += 1; //Nacio la celula
74         estadisticas.cantidadVivas += 1; //Hay una viva mas
75     } //Fin if
76 }
```

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

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

### Determinacion 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 celula en la funcion *determinarCaso*.

Esta funcion recibe tres parametros:

1. El estado actual de la celula
2. La cantidad de celulas vivas adyacentes(Se toma de *grillaNumerica*)
3. Las estadisticas del turno(se utilizan en la funcion *contarCelulas*, explicada a continuacion)

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

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

## Calculo de informacion del turno

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

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

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

[illegible]

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

- Restale a la cantidad de celulas vivas

Como se puede observar, la cantidad de celulas vivas se actualiza de forma dinamica. Por ende, luego del conteo inicial, segun 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 funcion *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(idem que para las muertes)

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

---

### Aclaracion sobre promedios

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

---

### Implementacion 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 informacion del turno anterior. Si ambas son iguales a cero luego, la funcion *evaluarEstabilidad* devuelve el valor **true** indicando que durante dos turnos consecutivos no hubo cambios.

Esta misma funcion, luego de hacer la comparacion 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.

### Implementacion 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. Copiandose los valores guardados en la variable de Tablero, *grillaInicial*.

## **Implementacion de fin de juego**

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