

# Informe de trabajo

---

## Juego de la Vida Recargado V1.0

---

### Informe de trabajo

---

Para la realizacion de esta primera entrega se realizaron las siguientes suposiciones:

1. Que por reiniciar el juego se entiende la carga de los datos contenidos en el archivo de entrada.
2. Que cuando el usuario decide terminar con el juego, la ejecución del programa no tiene por que seguir adelante.
3. Que el usuario solo ingresara datos numéricos para la posicion de una celula dada, por lo que no es necesario verificar si lo ingresado no son numeros.
4. Que la salida, en esta primer entrega sin generacion de imagenes, se da por consola. Sin delimitacion de la grilla con caracteres para tal fin
5. Que cuando el usuario deba ingresar una opcion, ingresada un unico caracter o una cadena cuando sea pertinente.
6. Que el ejemplo del archivo para la carga del tablero debe ser tomado como referencia exacta.
7. Que si durante un turno no ocurren cambios en el tablero, el mismo tuvo la misma composicion a lo largo de dos turnos. Por lo que el juego se considera congelado
8. Que un gen, cuya intensidad disminuye a 0 puede volver a aparecer nuevamente en el tablero, en tal caso, la edad del gen sera la edad correspondiente a su primera aparicion en el tablero.
9. Que si un gen mutado se encuentra en el tablero, la edad del mismo sera tomada de la edad del tablero. Ademas, si un gen mutado, se encuentra en la carga genetica de la celula, el mismo no es agregado a la carga genetica. Por ende se prioriza la transferencia sobre la mutacion.
10. Que unicamente se debe imprimir el tablero, y no las cantidades de celulas en las imagenes bmp.

### Manual del usuario

---

## Carga de datos

La carga de datos debe realizarse en el archivo *infoTablero* que se encuentra en la carpeta *celulas* dentro de *JuegoDeLaVida*. El formato del archivo debe respetarse para la correcta ejecucion del programa, a continuacion se ejemplifica la disposicion de los datos dentro del mismo:

```
tablero 20 80
celula 10 30
gen 1000100010001111000001 20
gen 10001001010101011110010001 80
gen 1001111 90
fin
celula 11 30
gen 1000000111 70
gen 1001111 20
fin
celula 12 30
gen 1000100010001111000001 30
gen 10001001010101011110010001 90
gen 1001111 10
fin
celula 11 29
gen 10001001010101011110010001 15
fin
celula 10 31
gen 1000100010001111000001 30
gen 10001001010101011110010001 90
gen 1001111 10
fin
```

Como se puede ver, primero se pasan las dimensiones del tablero (filas, columnas) para luego comenzar a informar las posiciones en las cuales se encontraran las celulas en el estado inicial del mismo, tambien con el mismo formato (filas, columnas).

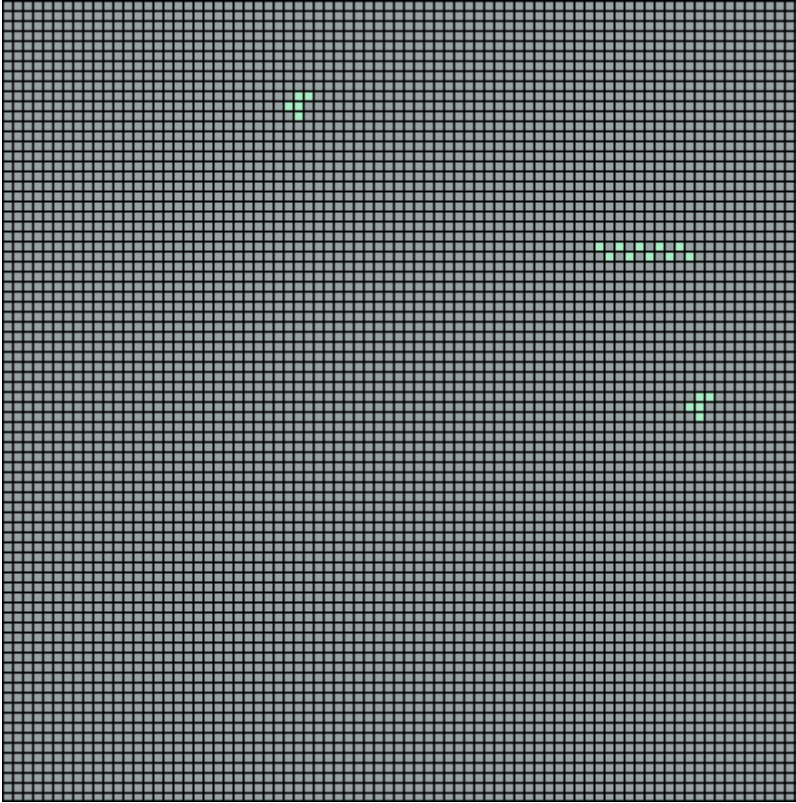
A continuacion de informar la posicion de una celula, se procede a ingresar la carga genetica de la misma. EL nombre del gen, con su cadena de bits y posteriormente la intensidad del mismo dentro de esa celula particular.

Una vez finalizado el ingreso de los genes, se debe explicitar el fin de la carga de datos para esa celula particular.

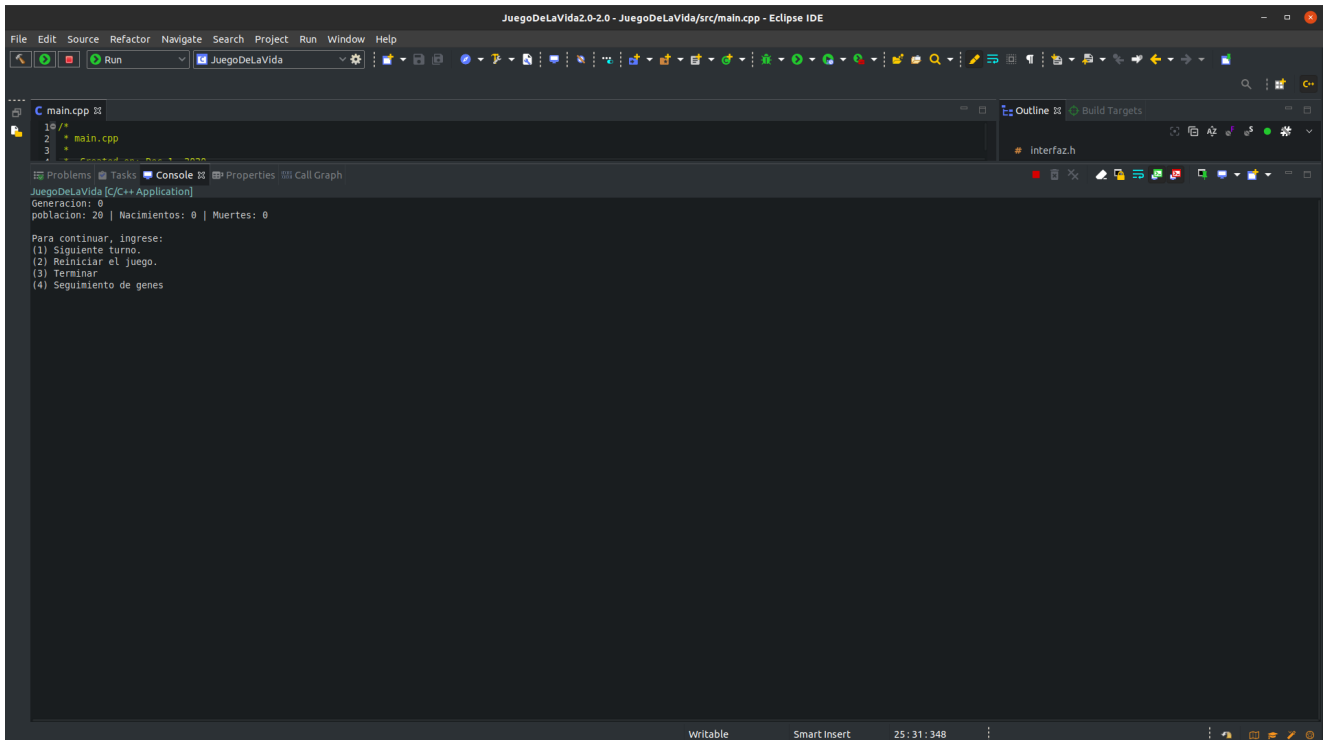
## Primera impresion del tablero

Si la carga de datos, se realizo de forma exitosa. Se procedera a observar la impresion inicial del tablero al archivo bmp correspondiente junto con la presentacion del menu del juego, como se muestra a continuacion:

- Imagen del estado inicial del tablero(se encuentra, como todas las imagenes, en la carpeta turnos)



- Imagen del menu de juego, mostrando el estado actual del tablero



## Avanzando de turno

Si la opcion elegida por el usuario es la primera, entonces se avanzara de turno. Esto implica la actualizacion del tablero de juego y la nueva impresion del mismo. Al realizarse este proceso. Las celulas, nacen mueren, se transfieren genes y algunos pueden mutar. Luego de un salto de turno, al usuario se le presenta informacion acerca de:

1. La generacion actual de celulas(equivalente al numero de turno en el que se encuentra)
2. La cantidad de celulas que murieron en el cambio de turno
3. La cantidad de celulas que nacieron en el cambio de turno
4. Los promedios de muertes y nacimientos a lo largo de todo el juego
5. El estado del juego

La ultima pieza de informacion, el estado del juego. Puede indicarse como activo o congelado. En caso de que este *activo*, Indica que durante el ultimo turno, hubo al menos una muerte o un nacimiento. Si el estado del juego se encuentra en *congelado* esto indica que durante el cambio de turno, no ocurrieron cambios en el tablero.

Esto se muestra a continuacion:

The image shows the Eclipse IDE interface with the following components:

- Top Bar:** Title bar reads "JuegoDeLaVida2.0-2.0 - JuegoDeLaVida/src/main.cpp - Eclipse IDE".
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Toolbar:** Standard Eclipse development icons including Run, Save, and various navigation tools.
- Editor Area:** The main window shows "main.cpp" with the following code:

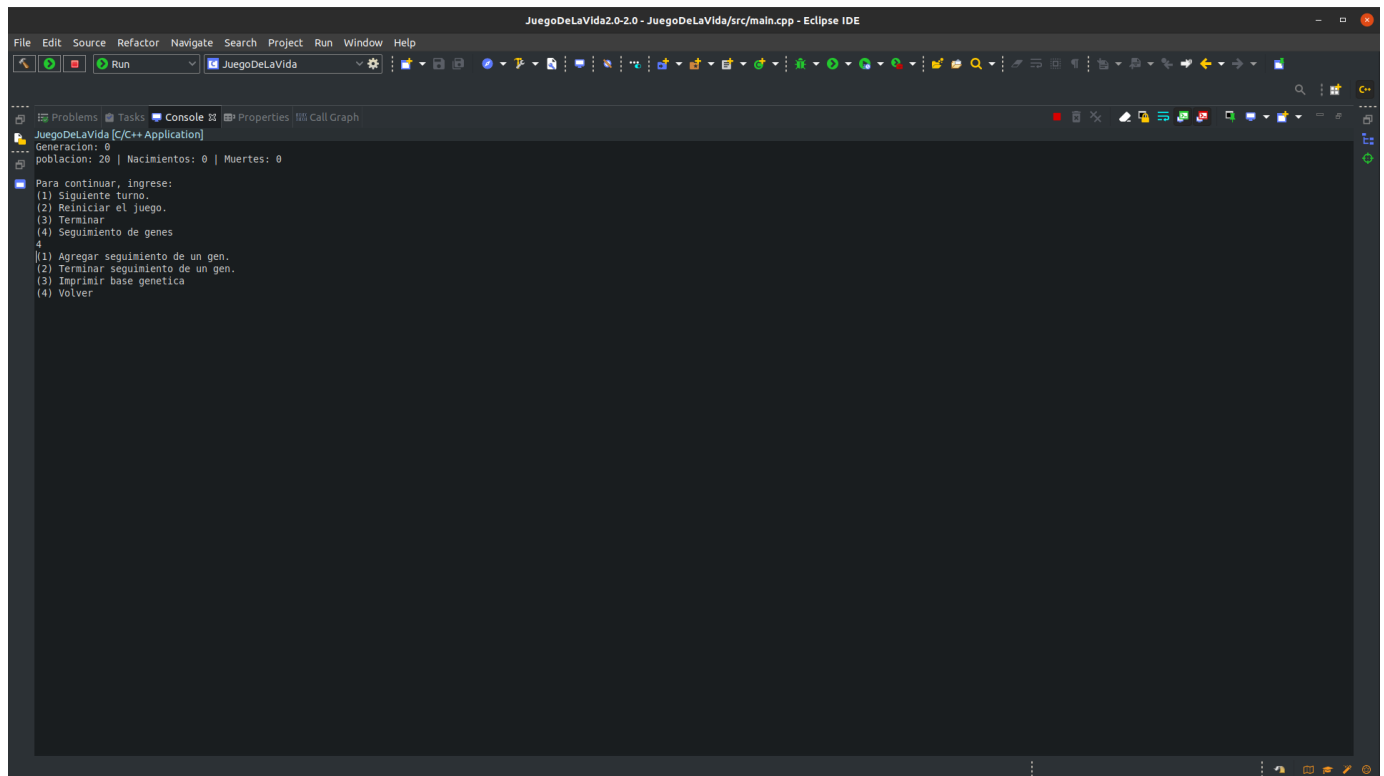
```
1 // *  
2 * main.cpp  
3 *  
4 * Created on: Dec 1, 2020
```
- Outline View:** Located on the right, it shows a file named "Interfaz.h".
- Console View:** The bottom panel displays the output of the application. It shows the game starting at Generation 0 with a population of 20 and 0 deaths. It then proceeds to Generation 1 (population 28, 4 deaths) and Generation 2 (population 30, 16 deaths). After each generation, a menu is displayed with options: (1) Siguiente turno, (2) Reiniciar el juego, (3) Terminar, and (4) Seguimiento de genes. The user has selected option 1 for the first two generations and option 4 for the third.

## reiniciando el Juego

Si la opcion elegida es la segunda, el reincio del juego. El programa volvera a leer nuevamente el archivo de entrada. Por lo que si no se realizaron cambios sobre el mismo por parte del usuario, el estado inicial del juego sera el mismo.

## Siguiendo genes

El menu le ofrece al usuario la opcion de comenzar o terminar con el seguimiento de un gen. Ademias, se ofrece la posibilidad de observar la base genetica completa del tablero. Esto es, todos los genes que alguna vez hayan estado presentes en alguna celula. Estas opciones se presentan en el menu de seguimiento de genes, que es la opcion 4 del mismo. Una muestra de como se presenta el mismo se encuentra a continuacion:



```
File Edit Source Refactor Navigate Search Project Run Window Help
JuegoDeLaVida2.0-2.0 - JuegoDeLaVida/src/main.cpp - Eclipse IDE
Run
JuegoDeLaVida
Problems Tasks Console Properties Call Graph
JuegoDeLaVida [C/C++ Application]
Generacion: 0
poblacion: 20 | Nacimientos: 0 | Muertes: 0
Para continuar, Ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
4
(1) Agregar seguimiento de un gen.
(2) Terminar seguimiento de un gen.
(3) Imprimir base genetica
(4) Volver
```

Al momento de iniciar el seguimiento de un gen, el programa verifica que el mismo se encuentre en la base genetica del tablero. El seguimiento comienza en el turno posterior al cual se encuentra actualmente el tablero.

Por otro lado, al momento de terminar el seguimiento de un gen, el mismo finaliza inmediatamente. Imprimiendose en el mismo momento los datos obtenidos a lo largo del seguimiento del gen.

Los seguimientos de genes, recolectan turno a turno, la intensidad acumulada del gen en particular. El programa permite que el usuario realice la cantidad de seguimientos que requiera.

A continuacion se muestran imagenes de la impresion de la base genetica, el inicio de un seguimiento y la finalizacion del mismo:

## Impresion base genetica

```
File Edit Source Refactor Navigate Search Project Run Window Help
JuegoDeLaVida2.0-2.0 - JuegoDeLaVida/src/main.cpp - Eclipse IDE

Problems Tasks Console Properties Call Graph
JuegoDeLaVida [C/C++ Application]
Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
1
Generacion: 13
poblacion: 32 | Nacimientos: 16 | Muertes: 10
Promedio de nacimientos: 12.1538 | Promedio de muertes: 11.2308
estado del juego: activo

Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
1
Generacion: 14
poblacion: 38 | Nacimientos: 12 | Muertes: 6
Promedio de nacimientos: 12.1429 | Promedio de muertes: 10.8571
estado del juego: activo

Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
1
Generacion: 15
poblacion: 38 | Nacimientos: 12 | Muertes: 12
Promedio de nacimientos: 12.1333 | Promedio de muertes: 10.9333
estado del juego: activo

Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
4
(1) Agregar seguimiento de un gen.
(2) Terminar seguimiento de un gen.
(3) Imprimir base genetica
(4) Volver
3
100011->100000->110001->100100->110010->1000100010000110001001->110110->110000->100110->10001001010101111101110->10100->100001->10001->10000001110111011001010111->110011->100010010101010110010110->100010010101010110110011001->
(1) Agregar seguimiento de un gen.
(2) Terminar seguimiento de un gen.
(3) Imprimir base genetica
(4) Volver
```

Inicio seguimiento de un gen

```
File Edit Source Refactor Navigate Search Project Run Window Help
JuegoDeLaVida2.0-2.0 - JuegoDeLaVida/src/main.cpp - Eclipse IDE

Problems Tasks Console Properties Call Graph
JuegoDeLaVida [C/C++ Application]
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
1
Generacion: 23
poblacion: 60 | Nacimientos: 26 | Muertes: 40
Promedio de nacimientos: 16.5217 | Promedio de muertes: 14.7826
estado del juego: activo

Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
1
Generacion: 24
poblacion: 92 | Nacimientos: 44 | Muertes: 12
Promedio de nacimientos: 17.6667 | Promedio de muertes: 14.6667
estado del juego: activo

Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
1
Generacion: 25
poblacion: 78 | Nacimientos: 42 | Muertes: 56
Promedio de nacimientos: 18.64 | Promedio de muertes: 16.32
estado del juego: activo

Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
4
(1) Agregar seguimiento de un gen.
(2) Terminar seguimiento de un gen.
(3) Imprimir base genetica
(4) Volver
1
Ingresar el gen a seguir.
0
(1) Agregar seguimiento de un gen.
(2) Terminar seguimiento de un gen.
(3) Imprimir base genetica
(4) Volver
```

Fin seguimiento de un gen

Al finalizar el seguimiento de un gen, se generara un archivo en la carpeta seguimiento con el siguiente nombre: gen-turnoInicial-turnoFinal.bmp.

Tener en cuenta que si el seguimiento es finalizado en la generacion 4, entonces el seguimiento sera hasta la generacion 3.

Acontinuacion, se muestra la finalizacion de un seguimiento y el archivo generado, a modo de ejemeplo.

JuegoDeLaVida2.0-2.0 - JuegoDeLaVida/src/main.cpp - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Run JuegoDeLaVida

Problems Tasks Console Properties Call Graph

```
Restore
JuegoDeLaVida [C/C++ Application]
del juego: activo

Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
1
Generación: 49
poblacion: 96 | Nacimientos: 46 | Muertes: 40
Promedio de nacimientos: 23.8367 | Promedio de muertes: 22.2857
estado del juego: activo

Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
1
Generación: 50
poblacion: 128 | Nacimientos: 64 | Muertes: 32
Promedio de nacimientos: 24.64 | Promedio de muertes: 22.48
estado del juego: activo

Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
1
Generación: 51
poblacion: 90 | Nacimientos: 56 | Muertes: 94
Promedio de nacimientos: 25.2549 | Promedio de muertes: 23.8824
estado del juego: activo

Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
4
(1) Agregar seguimiento de un gen.
(2) Terminar seguimiento de un gen.
(3) Imprimir base genetica
(4) Volver
2
ingresar gen para terminar seguimiento
0
|
```



## Finalizando la ejecucion

Si se elige la opcion 3, finaliza la ejecucion del programa. Imprimiendose consigo un mensaje que explicita el hecho.



```
JuegoDeLaVida2.0-2.0 - JuegoDeLaVida/seguimientos/gen-0-25-50.bmp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Run
C:\main.cpp gen-0-25-50.bmp 1100x1100 pixels
file:///home/francisco/Documents/algo2/TP/JuegoDeLaVida2.0-2.0/JuegoDeLaVida/seguimientos/gen-0-25-50.bmp

Problems Tasks Console Properties Call Graph
<terminated> (exit value: 0) JuegoDeLaVida [C/C++ Application] /home/francisco/Documents/algo2/TP/JuegoDeLaVida2.0-2.0/JuegoDeLaVida/Debug/JuegoDeLaVida (2/15/21, 5:48 PM)
(1) Terminar
(4) Seguimiento de genes
1
Generacion: 50
poblacion: 128 | Nacimientos: 64 | Muertes: 32
Promedio de nacimientos: 24.64 | Promedio de muertes: 22.48
estado del juego: activo

Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
1
Generacion: 51
poblacion: 98 | Nacimientos: 56 | Muertes: 94
Promedio de nacimientos: 25.2549 | Promedio de muertes: 23.8824
estado del juego: activo

Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
4
(1) Agregar seguimiento de un gen.
(2) Terminar seguimiento de un gen.
(3) Imprimir base genetica
(4) Volver
2
Ingresar gen para terminar seguimiento
0
(1) Agregar seguimiento de un gen.
(2) Terminar seguimiento de un gen.
(3) Imprimir base genetica
(4) Volver
4
Para continuar, ingrese:
(1) Siguiente turno.
(2) Reiniciar el juego.
(3) Terminar
(4) Seguimiento de genes
3
----- FIN DEL JUEGO -----
```

# Manual del programador

## TDAs de la implementacion

- TDA: Nodo
  - Realizado como un template para listas simplemente enlazadas.
- TDA: lista
  - Realizado como un template de lista, tiene algunas particularidades expuestas posteriormente
- TDA: Pila
  - Realizado en base a metodos de lista, hecho como template de pila
- TDA: Cola
  - Unicamente realizado para *unsigned int* dada su utilizacion puntual para ese tipo de dato
- TDA: Juego
  - Contiene al tablero de juego y se encarga de manejar el menu del mismo.
- TDA: Tablero
  - Contiene dentro de si, toda la informacion general relacionada al juego esto incluye:
    1. Cantidad de filas y columnas
    2. Poblacion total de celulas

3. Muertes y nacimientos del ultimo turno
  4. Totales de muertes y nacimientos a lo largo del juego
  5. Numero de turno actual
  6. La base genetica total del juego
  7. Las estrtucturas de seguimiento de genes
  8. El tablero en si mismo
- TDA: Celula
    - Se encarga de representar a una celula en el tablero, por lo que contiene:
      1. El estado de una celula
      2. Los genes de la misma
      3. Si hubo un cambio de estado
  - TDA: Genes
    - Se encarga de manejar la informacion genetica de una celula particular por lo que contiene:
      1. La cantidad de progenitores (utilizada tambien como cantidad de celulas vivas adyacentes)
      2. La lista con la informacion genetica de cada uno de los genes en particular
  - TDA: InformacionGenetica
    - Se encarga de manejar el estado de un gen en un lugar particular, contiene entonces:
      1. Los bits del gen
      2. La edad del gen
      3. La intensidad de esa instancia del gen
  - TDA: ListaSeguimiento
    - Se encarga de manejar los seguimientos de los genes, utiliza nodos especiales para este proposito
  - TDA: NodoGen
    - Nodos para listas simplemente enlazadas, que poseen los bits de un gen y ademas una cola con la informacion del seguimiento en cada turno

## Implementacion general del juego

La forma mas general para exponer los algoritmos que son utilizados en esta implementacion del juego de la vida se pueden exponer de la siguiente manera.

En primer lugar, se inicializa una instancia de Juego con la ruta del archivo donde se encuentran

los datos de las células. Al inicializarse esta instancia, se genera un tablero. Llamado *Partida* de ahora en adelante. *Partida* se encarga de leer el archivo, de forma secuencial y guardando la información pertinente. Primero genera el tablero, con su cantidad de filas y columnas. Luego agrega una a una las células informadas en el archivo, con sus cargas genéticas correspondientes, guardadas en primer lugar en una *Lista de InformacionGenetica*. Con la carga de cada célula, se agregan, además, los genes que correspondan a la *baseGenetica* del juego. Finalizada la carga de datos, se muestra el tablero y se le presenta al usuario un menú de opciones:

### 1. Avanzar de turno

Si el usuario opta por esta opción, el proceso se puede describir de la siguiente manera. Se itera el tablero hasta encontrar la primera célula viva. Las ocho células a su alrededor, se les suma un progenitor a sus genes y si los mismos no fueron nunca transferidos y hay menos de tres hasta el momento se agregan los genes de la célula aledaña.

De esta forma se itera a lo largo de todo el tablero. Luego, se repite la operación, pero esta vez, se buscan células, vivas o muertas. Al encontrarse con una célula se pueden dar las siguientes situaciones:

#### 1. Que ya este viva

en tal caso, si debe permanecer viva, no se realizan cambios sobre la misma. Pero si debe morir, la célula es destruida del tablero, dando la posibilidad de que luego nazca una nueva célula en su lugar.

#### 2. Que la célula este muerta

En este caso, se verifica el puntaje de la misma. Si cumple con los requerimientos, se pasa al proceso de transferencia de los genes (que se realiza mediante una mezcla de los mismos) y luego a la mutación de los genes pertinentes, si es que existen. Quedando al final de este proceso como una célula viva en el tablero.

Luego de finalizada la operación, se procede a imprimir nuevamente el tablero, junto con la información recogida a lo largo del cambio de turno.

### 2. Reiniciar el juego

En este caso, la operación es simplemente eliminar la instancia actual de *Partida* para proceder a armar una nueva, con los datos del mismo archivo.

### 3. Seguimiento de genes

Se puede optar por varias opciones, de donde destaca iniciar un seguimiento nuevo o finalizar uno existente.

En el primer caso, Se agrega a la *listaSeguimiento* del tablero, el seguimiento del gen en particular, agregando al

*NodoGen* nuevo, su cadena de bits y una cola vacía. A lo largo de los turnos subsiguientes, se va a buscar en los genes de las células vivas esta cadena de bits, si se encuentra, se suma su intensidad.

En el segundo caso, se procede a la eliminación del nodo de la *ListaSeguimiento* para

imprimir la cola con los resultados obtenidos a lo largo de los turnos durante los cuales estuvo activo el seguimiento de ese gen.

## Implementacion de Nodo

Hay una particularidad de los nodos en esta implementacion de los mismos, la llave a los nodos es en este caso particular. El valor guardado en los mismos, se encuentra almacenado en memoria dinamica. Por ende, las llave de estos nodos esta implementada como un puntero a su valor particular.

Esta implementacion, permite la devolucion por referencia del valor contenido en los nodos. Considerado de utilidad a la hora de modificar valores en los nodos de *InformacionGenetica* que requieren de metodos.

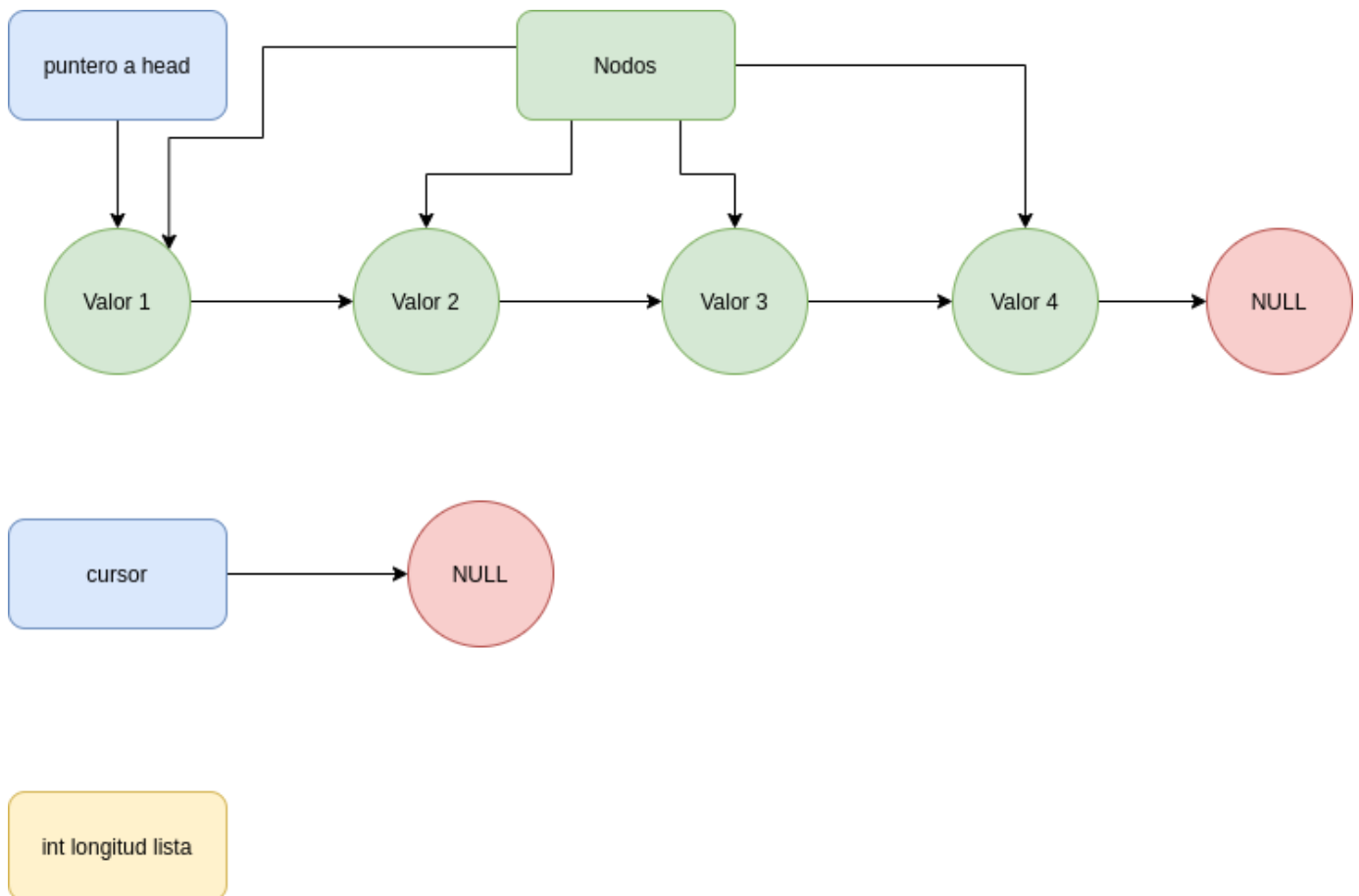
Ademas, se permite la devolucion del puntero a la llave de los nodos, nuevamente esto permite el cambio de un valor en un TDA contenido en los nodos, ya que de otra forma lo que obtendriamos seria una copia del mismo.

## Implemetacion de Lista

La lista en este caso, se encuentra implementada como una lista simplemente enlazada con cursor. Posee los metodos tipicos de lista, como agregar, eliminar y buscar elementos en la misma. Dada la particularidad de los nodos, el metodo para cambiar un valor de la lista, no esta disponible. Dado que el mismo cambio se puede realizar mediante la utilizacion del operador `[]`.

Ademas, la lista tiene dentro de su implementacion, un metodo *insertar* particular, dado que el mismo se puede utilizar para mantener el orden de los elementos dentro de la lista. Este metodo, inserta un elemento en la lista, luego de encontrarse con otro que cumpla la condicion de ser menor a el mismo. Esto permite, preparar listas ordenadas de forma decreciente. Este hecho, es de gran ayuda a la hora de la transferencia de los genes. Dado que la misma operacion se puede realizar mediante la mezcla de los mismos y no se requiere de una clasificacion mas complicada de los genes.

A continuacion, se muestra un esquema de como estan conectados los nodos y de las variables de la lista:



## Implementacion de Pila

La pila, esta implementada como un puntero a una lista. Donde los metodos que se pueden realizar sobre la misma se encuentran acotados para realizar las operaciones basicas de una pila: apilar, desapilar, y saber si esta vacia.

El uso de la misma dentro del programa se da durante la transferencia de los genes. Aqui, se apilan aquellos genes de intensidad 0 para finalmente, mezclarlos en el gen mutado final.

## Implementacion de Cola

La cola posee todas las primitivas basicas de la misma: acolar, desalocar y obtenerFrente. En este caso, dado que unicamente se utilizan colas para guardar intensidades de los genes, se opto por implementarla unicamente para el tipo de dato *unsigned int*.

## Implementacion del Tablero

El *Tablero* esta implementado como un puntero a puntero a puntero de *Celula*. La razon por la cual se opto por esta implementacion particular, es que permite mantener punteros a NULL, lo que en cierto sentido, mejora la eficiencia del uso de la memoria.

Aquellos punteros a NULL son considerados como celulas muertas, mientras que por otro lado, las *Celulas* pueden estar muertas tambien. En este sentido, durante la actualizacion del tablero entre turno y turno, aquellas celulas que hayan quedado muertas, son eliminadas del tablero. Quedando sus punteros en NULL.

El proceso mas importante que posee el tablero, en su actualizacion. La cual se realiza en la

funcion *actualizarTablero*

en primer lugar, se itera a lo largo del mismo y se definen los puntajes finales de cada una de las *Celulas* que posean alguna otra adyacente. Luego de realizado este proceso, la funcion llama a *definirTablero*.

*definirTablero* cumple el rol de definir que celula queda viva y cual no. Para lo cual itera sobre el tablero y llama a metodos de *Celula*. Luego, si una celula permanece viva, *definirTablero* se encarga de llamar al envejecimiento de sus genes y de llamar a metodos de *ListaSeguimiento* para llevar adelante a los procesos pertinentes.

## Implementacion de Celula

La *Celula* tiene la capacidad de nacer, morir, de agregar progenitores si es que sus genes no fueron mezclados (a este proceso se lo conoce como *transferencia* de ahora en mas) y de poder almacenar la cantidad de celulas vivas vecinas que tiene.

Durante el proceso de carga, si la celula no transfirio sus genes y la cantidad de sus progenitores es menor a 3. Se puede agregar na carga genetica nueva a la misma. Caso contrario solamente puede sumar otro progenitor. Luego de la transferencia de genes y si la celula permanece viva, la cantidad de progenitores pasa a ser una buena forma de contar la cantidad de celulas vivas adyacentes, por eso todas las celulas pueden agregar progenitores en cualquier estado en el que se encuentren.

Las celulas se aplican a ellas mismas las reglas del juego, esto se realiza en la funcion *decidirEstado* que llama a *aplicarReglas* y a *reseteearPuntaje*. La primera, se encarga de permitir el nacimiento o la continuacion en el estado de *viva* si cumple con las condiciones establecidas. *reseteearPuntaje* pone el numero de progenitores nuevamente a cero para que a futuro se pueda nuevamente actualizar el *Tablero*.

## Implementacion de Genes

Los *Genes* son un TDA que posee una *Lista<InformacionGenetica>* un booleano que indica si los genes fueron transferidos y un contador de progenitores/puntaje.

La razon de explicitar si los genes fueron o no transferidos, se debe a que la *Lista<InformacionGenetica>* se encuentra en caso de no haber sido transferidos como un array de punteros, con tres lugares disponibles. Caso contrario es simplemente un puntero a una unica *Lista*.

La mezcla de los genes se realiza en la funcion homonima *mezclarGenes* con una *Lista* que representa la carga genetica final y un *Pila* que contiene los genes que van a sufrir el proceso de combinacion. El proceso que se lleva a cabo es la mezcla de los genes. Los cuales se encuentran ordenados, de forma descendente(Se explica posteriormente). Durante la mezcla, existen tres posibilidades de transferencia con sus funciones homonimas *transferenciaSimple*, *transferenciaDoble* y *transferenciaTriple*. En caso de *transferenciaSimple* el gen transferido tendra intensidad 0 y sera apilado. En los otros dos casos, se calcula en primer lugar la intensidad final del gen, en caso de que sea cero, el gen se apila. Caso contrario se lo inserta en la *cargaFinal*.

una vez obtenido el gen mutado, se verifica que el mismo no se encuentre ya presente en el tablero, en caso de que lo este, se verifica que no se encuentre en la carga genetica final y recién ahí se lo inserta en la misma. Si el gen se encuentra presente en el tablero entonces se cambia su edad por la del gen en el tablero, si no se mantiene como 1. Si el gen además de estar en el tablero, esta en la célula, entonces se prioriza el gen previamente existente en las células.

## Implementacion de InformacionGenetica

*InformacionGenetica* es un TDA que posee la informacion acerca de la instancia de un gen en una situacion particular. por lo que posee su edad, su intensidad particular y su cadena de bits. La *InformacionGenetica* se puede combinar con otras, puede cambiar su intensidad, su edad y puede compararse mediante los siguientes operadores con otras *\_InformacionGenetica\_s*: `==`, `!=`, `>` y `<`. La razon de la sobrecarga de tales operadores es que de esta forma, se nos permite comparar las cadenas de bits de varios genes y ordenarlos en una Lista utilizando el metodo *insertar* de las mismas, lo que nos permite realizar transferencias de forma mas simple. Además, *InformacionGenetica* utiliza la sobrecarga del operador `<<` de *Ostream* para permitirle usar a *Lista<InformacionGenetica>* el metodo *imprimir* de forma directa.

## Implementacion NodoGen

*NodoGen* es un nodo que posee, un puntero a *Cola*, además de poseer una clave, que en este caso es la cadena de bits correspondiente a un gen. *NodoGen* tiene sobrecargado el operador `==`, pero para su comparacion con instancias de *InformacionGenetica*. Lo que se compara en este caso es la cadena de bits de *NodoGen* con la cadena de bits de *InformacionGenetica*. Estos nodos, se encargan cada uno, de llevar adelante y almacenar los datos del seguimiento de un gen en particular.

## Implementacion de ListaSeguimiento

*ListaSeguimiento* es una implementacion particular de *Lista*, utiliza un puntero a una *Lista* de *NodoGen*. De esta forma, *ListaSeguimiento* se encarga de realizar las operaciones relativas al seguimiento de los genes.

## Implementacion de Juego

*Juego* maneja unicamente a la partida, en este sentido. *Juego* se encarga de implementar el menu y las interacciones con el usuario. Si bien maneja la partida, la misma no es un atributo del mismo. Partida se genera en memoria dinamica dentro del metodo *iniciarPartida* y se libera su espacio dentro del mismo metodo. *IniciarPartida* llama al metodo *desarrolloDelJuego* donde se encuentra el menu con las opciones correspondientes y las acciones que pueden ocurrir durante el desarrollo del *JuegoDeLaVida*.

## Impresiones en el juego

Si se desean cambiar los colores de las impresiones, ya sea el color del tablero y/o las células, o el color de las impresiones de seguimiento, referirse a las funciones:

1. *dibujarLineasSeparadoras* en el archivo `tablero.cpp`

- La variable **colorLineas** contiene los datos del color de las líneas de separación del tablero impreso (Negras por defecto)

2. *dibujarCélulaViva* en el archivo `tablero.cpp`

- La variable **verde** contiene el color de las células vivas en el tablero

3. *dibujarCélulaMuerta* en el archivo `tablero.cpp`

- La variable **oscuro** contiene el color de las células muertas en el tablero

4. *crearGrafico* en el archivo `listaSeguimiento.cpp`

- La variable **negro** contiene el color de las líneas de los ejes
- La variable **gris** contiene el color de las líneas indicadoras

5. *detenerSeguimiento* en el archivo `listaSeguimiento.cpp`

- La variable **rojo** contiene el color la línea de intensidades acumuladas