Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2025

LABORATORIO EPIDEMIOLOGICO DE GUATEMALA

PROYECTO 1

Carnet 202401753 – Bily Estuardo Vallecidos Folgar

RESUMEN

¿Qué es?

Una aplicación de consola en C# que simula cómo se comportan enfermedades en tejidos celulares humanos, usando las mismas reglas del famoso "Juego de la Vida de Conway".

Cómo funciona?

Se carga una rejilla cuadrada de células (por ejemplo 10x10) desde un archivo XML. Cada célula puede estar sana o contagiada. En cada período se aplican dos reglas:

- Una célula contagiada sobrevive si tiene exactamente 2 o 3 vecinos contagiados, si no, sana.
- Una célula sana se contagia si tiene exactamente 3 vecinos contagiados.

El sistema repite esto período a período buscando si algún patrón se repite

Qué detecta?

- **LEVE** → la enfermedad desaparece sola, nunca se repite ningún patrón
- **GRAVE** → detecta un patrón que se repite cada N períodos (N > 1)
- **MORTAL** → el patrón se repite cada 1 período, la enfermedad es incurable

INTRODUCCIÓN

El presente documento describe el desarrollo del Proyecto 1 del curso de Introduccion a la Programacion y Computacion 2, el cual consiste en una aplicacion de simulacion epidemiologica desarrollada en el lenguaje de programacion C#.

La aplicacion tiene como objetivo simular el comportamiento de enfermedades en tejidos celulares humanos, utilizando un modelo matematico basado en el Juego de la Vida de Conway. A traves de esta simulacion, el sistema es capaz de determinar si una enfermedad sera leve, grave o mortal para un paciente especifico, basandose en los patrones que se forman en una rejilla cuadrada de celulas sanas y contagiadas.

El proyecto implementa conceptos fundamentales de la Programacion Orientada a Objetos (POO), estructuras de datos como listas enlazadas creadas desde cero, manejo de archivos XML para la entrada y salida de datos, y visualizacion grafica mediante la herramienta Graphviz.

Descripcion del Problema

El laboratorio de investigacion epidemiologica de Guatemala necesita una herramienta que permita analizar el comportamiento de enfermedades en tejidos celulares. El sistema trabaja con rejillas cuadradas de M x M celdas, donde cada celda contiene una celula que puede estar sana (0) o contagiada (1).

Reglas de Simulacion

El comportamiento de las celulas se rige por las siguientes reglas aplicadas en cada periodo:

- Regla 1: Una celula contagiada continua contagiada si tiene exactamente 2 o 3 celulas contagiadas vecinas. De lo contrario, sana en el siguiente periodo.

Regla 2: Una celula sana se contagia si tiene exactamente 3 celulas contagiadas vecinas en el periodo actual.

Clasificacion de Resultados

Luego de ejecutar los periodos de simulacion, el sistema clasifica la enfermedad en uno de los siguientes casos:

Resultado	Descripcion
LEVE	La enfermedad desaparece antes de cumplir los periodos configurados. No se detecta ningun ciclo repetitivo.
GRAVE	Se detecta un patron que se repite cada N periodos, donde N es mayor a 1.
MORTAL	El patron se repite cada 1 periodo, lo que indica que la enfermedad es incurable.

Figura 1. Clasificacion y Niveles

Fuente: elaboración propia

Solucion Implementada

La solucion fue desarrollada en C# utilizando .NET 10, siguiendo los principios de la Programacion Orientada a Objetos. El proyecto esta organizado en cuatro capas principales: Estructuras, Modelos, Logica y Archivos.

Estructuras de Datos

Se implementaron dos clases fundamentales para el manejo de datos, sin utilizar ninguna estructura nativa de C# como List, Queue o Stack:

- Clase Nodo<T>: Clase generica que sirve como nodo base para la lista enlazada. Contiene un dato de tipo generico T y una referencia al siguiente nodo.
- Clase ListaEnlazada<T>: Implementacion propia de una lista enlazada simple generica. Provee operaciones de agregar, obtener por indice, obtener tamano, verificar si esta vacia, limpiar y obtener la cabeza para recorridos manuales.

Modelos

- Celda: Representa una celula individual en la rejilla. Contiene la fila, columna y estado (sana o contagiada). Incluye un metodo Clonar() para copiar su estado sin modificar el original.
- Rejilla: Estructura principal de M x M celdas. Almacena las celdas en una ListaEnlazada propia. Implementa las reglas de Conway, cuenta celulas sanas y contagiadas, genera el patron como cadena para deteccion de ciclos, y permite clonar la rejilla completa.
- Paciente: Contiene los datos personales (nombre, edad), configuracion (periodos, rejilla) y los resultados del analisis (resultado, N, N1).

Logica de Negocio

- Simulador: Clase central que maneja la simulacion de periodos. Mantiene un historial de patrones como cadenas de texto en una ListaEnlazada. En cada periodo ejecuta las reglas de Conway, verifica si el patron actual ya existio en el historial para detectar ciclos, y clasifica la enfermedad como leve, grave o mortal segun corresponda.
- GeneradorGraphviz: Genera archivos .dot y los convierte a imagenes PNG utilizando el motor de Graphviz. Visualiza la rejilla como una cuadricula donde las celulas contagiadas se muestran en azul y las sanas en blanco.

MANEJO DE ARCHIVOS

Se implementaron dos clases para el manejo de archivos XML:

- LectorXML: Lee el archivo XML de entrada y carga todos los pacientes con sus rejillas en una ListaEnlazada<Paciente>.
- EscritorXML: Genera el archivo XML de salida con los resultados de cada paciente, incluyendo el resultado (leve/grave/mortal) y los valores de N y N1 cuando aplican.

Algoritmos Principales:

ALGORITMO DE SIMULACION POR PERIODO:

El algoritmo de simulacion de un periodo funciona de la siguiente manera:

1. Se ejecuta EjecutarPeriodo() en la rejilla del paciente activo.
2. Para cada celula, se cuenta el numero de vecinos contagiados (8 vecinos posibles).
3. Se aplican las reglas de Conway y se almacena el nuevo estado en una lista temporal.
4. Se actualiza el estado de todas las celulas con los valores calculados.
5. Se verifica si el nuevo patron ya existe en el historial para detectar ciclos.

ALGORITMO DE DETECCION DE CICLOS:

La deteccion de ciclos es el algoritmo central del proyecto. Funciona de la siguiente manera:

- Al iniciar la simulacion, se guarda el patron inicial (periodo 0) en el historial de patrones.
- Despues de cada periodo, se convierte el estado de la rejilla a una cadena de 0s y 1s (patron).
- Si la rejilla queda completamente vacia (0 celulas contagiadas), se clasifica como LEVE.
- Se busca el patron actual en el historial. Si se encuentra en el indice I, se calcula N1 = periodo_actual - I.
- Si N1 = 1, la enfermedad es MORTAL. Si N1 > 1, la enfermedad es GRAVE.
- Si se alcanza el limite de periodos sin encontrar ciclo, se clasifica como LEVE.

Funcionalidades del Sistema:

La aplicacion provee las siguientes opciones a traves de un menu de consola interactivo:

CONCLUSIONES

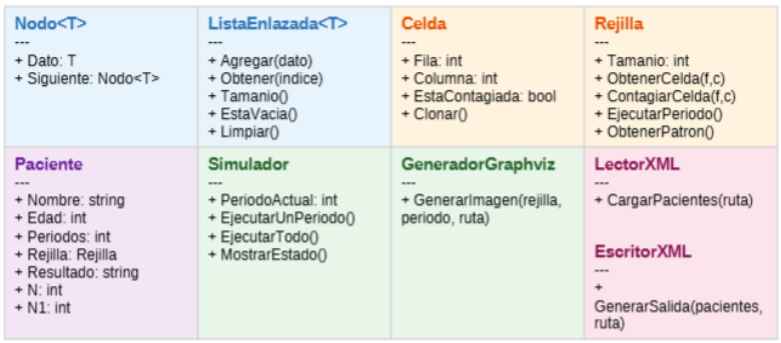
- Se implemento exitosamente una aplicacion en C# que simula el comportamiento de enfermedades celulares, cumpliendo con todos los requisitos del proyecto.
- La implementacion de listas enlazadas desde cero sin usar estructuras nativas de C# permitio comprender mejor el funcionamiento interno de las estructuras de datos.
- El algoritmo de deteccion de ciclos basado en comparacion de patrones como cadenas de texto resulto eficiente para rejillas de tamano moderado.

Referencias bibliográficas

- Conway, J. (1970). The Game of Life. Scientific American.
- Microsoft. (2024). Documentacion de C# - Microsoft Docs. https://docs.microsoft.com/es-es/dotnet/csharp/
- Graphviz. (2024). Graphviz - Graph Visualization Software. https://graphviz.org/
- Enunciado Proyecto 1 - Introduccion a la Programacion y Computacion 2, USAC 2026.

APENDICE A: DIAGRAMA DE CLASES

El siguiente diagrama muestra la estructura de clases implementada en la solucion:



Opcion	Descripcion
1	Cargar archivo XML de entrada con los datos de los pacientes.
2	Seleccionar un paciente y visualizar su rejilla inicial.
3	Ejecutar un periodo a la vez con visualizacion grafica en Graphviz.
4	Ejecutar todos los periodos automaticamente hasta detectar el resultado.
5	Generar archivo XML de salida con los resultados de todos los pacientes.
6	Generar imagen Graphviz del estado actual de la rejilla.
7	Limpiar la memoria del sistema eliminando todos los pacientes cargados.
8	Salir del programa.