Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2do. Semestre 2022.

PROYECTO 1 – DETERMINACION DE UNA ENFERMEDAD

202000562 – Pedro Luis Pu Tavico

Resumen

El laboratorio de investigación epidemiológica de Guatemala ha estado investigando la forma en que las enfermedades infectan las células del cuerpo humano y se expanden produciendo enfermedades graves e incluso la muerte. Los científicos, luego de hacer experimentos han logrado identificar patrones que pueden determinar si una enfermedad producirá una enfermedad leve, una enfermedad grave, o si el paciente morirá a causa de la enfermedad.

Para esta aplicación se hará uso de un algoritmo que analizará celda por celda en cada rejilla identificando si en la celda posicionado actualmente está sana o contagiada y así poder aplicar cada uno de los casos expuestos y encontrar un patrón determinado.

Palabras clave

TDA: es un conjunto de datos u objetos creado de manera personalizada por un programador para un fin específico.

XML: es un lenguaje de marcado similar a HTML.

Nodo: es un registro que contiene un dato de interés y al menos un puntero para referenciar (apuntar) a otro nodo.

POO: La Programación Orientada a Objetos es un paradigma de programación que parte del concepto de "objetos" como base.

Apuntador: es una variable que contiene la dirección de memoria de otra variable.

Abstract

Guatemala's epidemiological research laboratory has been investigating how diseases infect the cells of the human body and spread to produce serious illness and even death. The scientists, after conducting experiments, have been able to identify patterns that can determine if a disease will produce a mild illness, a severe illness, or if the patient will die from the disease.

For this application, an algorithm will be used to analyze cell by cell in each grid, identifying whether the currently positioned cell is healthy or infected and thus be able to apply each of the exposed cases and find a certain pattern.

Keywords

TDA: is a set of data or objects created in a personalized way by a programmer for a specific purpose. A TDA is an abstraction that allows modeling the characteristics of a particular element.

XML: It is a markup language similar to HTML.

Node: is a record that contains a data of interest and at least one pointer to reference (point) to another node.

OOP: Object Oriented Programming is a programming paradigm that starts from the concept of "objects" as a base.

Pointer: is a variable that contains the memory address of another variable.

Introducción

Esta aplicación software consiste en verificar celda por celda en la rejilla inicial del periodo inicial dada por el paciente y así poder aplicar cada uno de los casos expuestos. Al terminar de analizar celda por celda en la rejilla inicial el programa genera un patrón y en el siguiente periodo vuelve a verificar celda por celda en la rejilla correspondiente al periodo actual y luego verifica si el patrón generado en el periodo anterior es igual al del periodo actual y poder así determinar el caso del paciente.

TDA

Un Tipo de dato abstracto (en adelante TDA) es un conjunto de datos u objetos al cual se le asocian operaciones. El TDA provee de una interfaz con la cual es posible realizar las operaciones permitidas, abstrayéndose de la manera en como están implementadas dichas operaciones. Esto quiere decir que un mismo TDA puede ser implementado utilizando distintas estructuras de datos y proveer la misma funcionalidad.

El paradigma de orientación a objetos permite el encapsulamiento de los datos y las operaciones mediante la definición de clases e interfaces, lo cual permite ocultar la manera en cómo ha sido implementado el TDA y solo permite el acceso a los datos a través de las operaciones provistas por la interfaz.

Memoria dinámica

Se refiere a aquella memoria que no puede ser definida ya que no se conoce o no se tiene idea del número de la variable a considerarse, la solución a este problema es la memoria dinámica que permite solicitar memoria en tiempo de ejecución, por lo que cuanta más memoria se necesite, más se solicita al sistema operativo. El sistema operativo maneja la memoria gracias al uso de punteros, por la misma naturaleza del proceso nos impide conocer el tamaño de la memoria necesaria en el momento de compilar.

Un dato importante es que como tal este tipo de datos se crean y se destruyen mientras se ejecuta el programa y por lo tanto la estructura de datos se va dimensionando de forma precisa a los requerimientos del programa, evitándonos así perder datos o desperdiciar memoria si hubiéramos tratado de definir la cantidad de memoria a utilizar en el momento de compilar el programa.

XML

XML es el acrónimo de Extensible Markup Language, es decir, es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos.

El lenguaje de marcado es un conjunto de códigos que se pueden aplicar en el análisis de datos o la lectura de textos creados por computadoras o personas. El lenguaje XML proporciona una plataforma para definir elementos para crear un formato y generar un lenguaje personalizado.

Un archivo XML se divide en dos partes: prolog y body. La parte prolog consiste en metadatos administrativos, como declaración XML, instrucción de procesamiento opcional, declaración de tipo de documento y comentarios. La parte del body se compone de dos partes: estructural y de contenido (presente en los textos simples).

El diseño XML se centra en la simplicidad, la generalidad y la facilidad de uso y, por lo tanto, se utiliza para varios servicios web. Tanto es así que hay sistemas destinados a ayudar en la definición de lenguajes basados en XML, así como APIs que ayudan en el procesamiento de datos XML - que no deben confundirse con HTML.

Listas Enlazadas

Una lista enlazada es una de las estructuras de datos fundamentales, y puede ser usada para implementar otras estructuras de datos. Consiste en una secuencia de nodos, en los que se guardan campos de datos arbitrarios y una o dos referencias, enlaces o punteros al nodo anterior o posterior. El principal beneficio de las listas enlazadas respecto a los vectores convencionales es que el orden de los elementos enlazados puede ser diferente al orden de almacenamiento en la memoria o el disco, permitiendo que el orden de recorrido de la lista sea diferente al de almacenamiento.

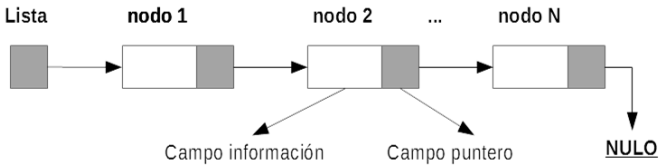


Figura 1. Ejemplo de lista simple enlazada.
Fuente: elaboración propia,

Listas Doblemente Enlazadas

Es una estructura de datos que consiste en un conjunto de nodos enlazados secuencialmente. Cada nodo contiene tres campos, dos para los llamados enlaces, que son referencias al nodo siguiente y al anterior en la secuencia de nodos, y otro más para el almacenamiento de la información (en este caso un entero). El enlace al nodo anterior del primer nodo y el enlace al nodo siguiente del último nodo, apuntan a un tipo de nodo que marca el final de la lista, normalmente un nodo centinela o puntero null, para facilitar el recorrido de la lista. Si existe un único nodo centinela, entonces la lista es circular a través del nodo centinela.

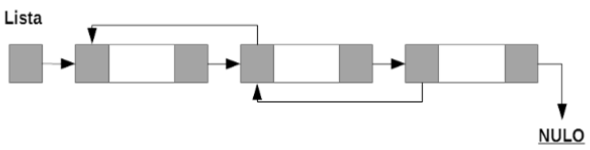


Figura 2. Ejemplo de lista doble enlazada.
Fuente: elaboración propia,

El doble enlace de los nodos permite recorrer la lista en cualquier dirección. Mientras que agregar o eliminar un nodo en una lista doblemente enlazada requiere cambiar más enlaces que en estas mismas operaciones en una lista enlazada simple, las operaciones son más simples porque no hay necesidad de mantener guardado el nodo anterior durante el recorrido, ni necesidad de recorrer la lista para hallar el nodo anterior, la referencia al nodo que se quiere eliminar o insertar es lo único necesario.

Lista Enlazada Pacientes

Esta clase almacena todos los datos de los pacientes ingresados por el usuario. En ella se encuentran las posiciones de cada una de las celdas infectadas, como también el tamaño de la rejilla y los datos personales del paciente.

Lista Doble Enlazada Periodos

Esta clase almacena todos los periodos pertenecientes a un paciente en específico. Debido a que es doblemente enlazada es posible verificar periodo por periodo para poder identificar si algún patrón se repite de una manera más sencilla.

Lista Doble Enlazada Números

Esta clase almacena todas las posiciones de las celdas infectadas que se encuentran en la rejilla perteneciente a un periodo.

Cargar Archivo

En esta clase se encuentra la lógica que utiliza el programa para almacenar los datos del paciente dados por el usuario a través de un archivo con extensión .xml.
Al ingresar un paciente por el usuario, el programa almacena en la lista doble enlazada números todas las posiciones de las celdas infectadas, luego almacena en la lista doble enlazada periodos la lista doble enlazada números la cual corresponde como periodo inicial ya que es el periodo dado por el usuario.

Por último, almacena la lista doble enlazada periodos en la lista enlazada pacientes la cual también contendrá los datos personales del paciente.

Paciente

En esta clase se encuentra la mayoría de la lógica del programa.
En ella se encuentra el algoritmo utilizado para determinar cuál será el caso del paciente, el cual consiste en lo siguiente:
El programa verifica cuales son las celdas que se encuentran contagiadas en la rejilla perteneciente al periodo inicial. Al determinar dichas celdas procede a almacenar las celdas que seguirán contagiadas en el siguiente periodo. Luego almacena las posiciones de las celdas sanas que se contagiaran al siguiente periodo.
Al terminar de estudiar cada uno de los casos pertenecientes a las celdas, genera un patrón que identificara al periodo en el que se encuentra actualmente. Por último, verifica si el patrón del periodo en el que se encuentra actualmente es idéntico al del periodo inicial, de no ser así, verifica con los demás periodos y si no encuentra parecido con ninguno de los periodos realiza los pasos mencionados anteriormente hasta encontrar similitud con algún patrón o hasta que se termine la cantidad de periodos a ejecutar ingresadas por el usuario a través del archivo (.xml).

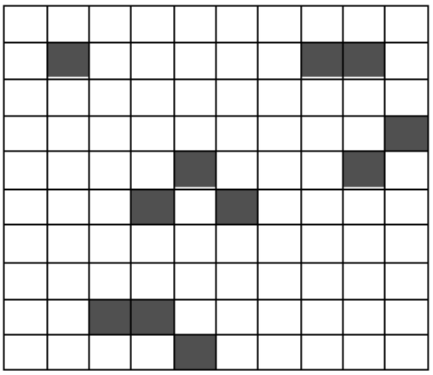


Figura 3. Ejemplo de rejilla con celdas contagiadas.
Fuente: elaboración propia,

Si el programa determina que un patrón se repite, verifica el numero del periodo de dicho patrón y concluye identificando el caso de la enfermedad.

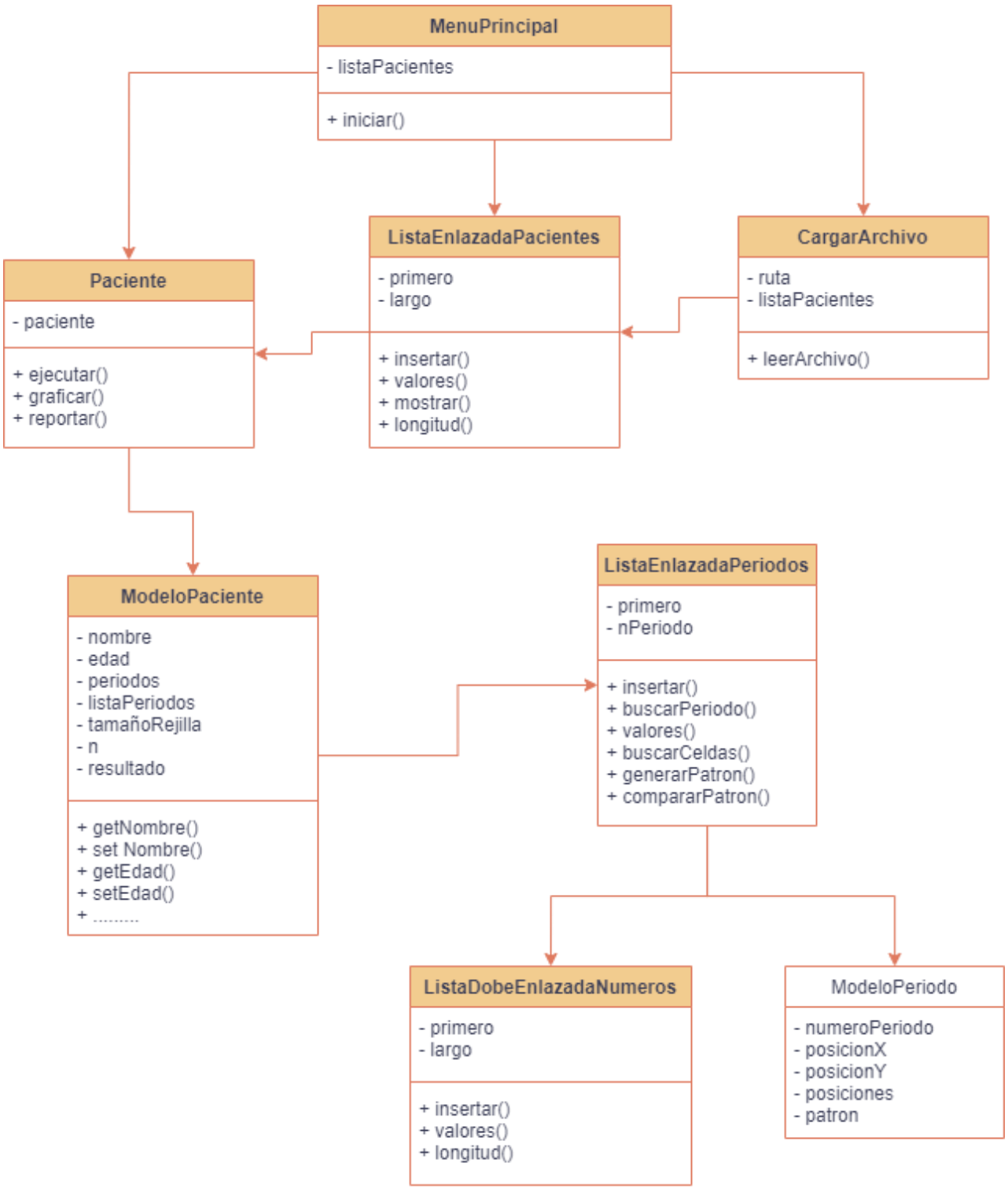
Conclusiones

- Para este proyecto se hicieron uso de estructuras de datos abstractas (TDA) utilizando el paradigma de la programación orientada a objetos, que permite hacer estructuras personalizadas que facilitan la resolución de este problema, estructurando las distintas partes del programa como módulos que se unen entre sí y son fáciles de modificar.
- Con el uso de los distintos algoritmos elaborados se puede concluir que son óptimos para la resolución de este problema ya que se puede llegar a determinar el resultado de una enfermedad por medio de las celdas infectadas.
- La utilización de TDA's permite hacer un uso óptimo de la memoria ya que son estructuras personalizadas.

Referencias bibliográficas

- Problemas resueltos de listas / Mónica Adriana Carreño León 2010.
- CAIRÓ O., GUARDATI S., Estructura de Datos, México, Mc Graw-Hill, 2001.
- JOYANES A. LUIS, Algoritmos y Estructuras de Datos: Una Perspectiva en C., España, Mc Graw-Hill, 2005.

Anexos


MenuPrincipal
- listaPacientes
+ iniciar()
ListaEnlazadaPacientes
- primero
- largo
+ insertar()
+ valores()
+ mostrar()
+ longitud()
CargarArchivo
- ruta
- listaPacientes
+ leerArchivo()
Paciente
- paciente
+ ejecutar()
+ graficar()
+ reportar()
ModeloPaciente
- nombre
- edad
- periodos
- listaPeriodos
- tamañoRejilla
- n
- resultado
+ getNombre()
+ set Nombre()
+ getEdad()
+ setEdad()
+ .....
ListaEnlazadaPeriodos
- primero
- nPeriodo
+ insertar()
+ buscarPeriodo()
+ valores()
+ buscarCeldas()
+ generarPatron()
+ compararPatron()
ListaDobeEnlazadaNumeros
- primero
- largo
+ insertar()
+ valores()
+ longitud()
ModeloPeriodo
- numeroPeriodo
- posicionX
- posicionY
- posiciones
- patron