Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2021.

INFORME DEL PROYECTO 2
INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 2

201902209 – Katheryn Darleny Yuman Oscal

Resumen

Una estructura de datos, en general se puede definir como cualquier colección o grupo de datos organizados de tal forma que tengan asociados un conjunto de operaciones para poder manipularlos. Las estructuras de datos se implementan a través de los lenguajes de programación y son un modelo que se caracteriza por permitir el almacenamiento y utilizar una determinada organización de datos Una estructura de datos puede ser de dos tipos: Estructuras de datos estáticas y Estructuras de datos dinámicas.

Estructuras de datos estáticas: Son aquellas en las que se asigna una cantidad fija de memoria y no cambia durante la ejecución de un programa, es decir, las variables no pueden crearse ni destruirse durante la ejecución del programa.

Estructuras de datos dinámicas: Son aquellas en las que su ocupación en memoria puede aumentar o disminuir durante el tiempo de ejecución de un programa. A su vez las estructuras de datos dinámicas se pueden clasificar en lineales y no lineales.

Abstract

A data structure generally can be defined as any collection or group of data organized in such a way that they have associated a set of operations to be able to manipulate them. Data structures are implemented through programming languages and are a model that is characterized by allowing the storage and use of a certain organization of data A data structure can be of two types: Static data structures and Dynamic data structures.

Static data structures: They are those in which a fixed amount of memory is allocated and does not change during the execution of a program, that is, the variables cannot be created or destroyed during the execution of the program.

Dynamic data structures: They are those in which their memory occupation can increase or decrease during the execution time of a program. In turn, dynamic data structures can be classified as linear and non-linear.

+++

Palabras clave

Nodo: Apuntador al nodo cuya dirección de memoria está en nodo.

Nodo(valor): El campo de información para el nodo apuntado por nodo.

Nodo(siguiente): El campo almacena la dirección de memoria del nodo sucesor del nodo. Dependiendo del tipo de lista un nodo puede tener más de un enlace.

Nodo(siguiente) ← NULL: El campo almacena un valor nulo para indicar que no existe sucesor del nodo.

Cabeza: dirección en donde se encuentra el primer nodo de la lista

Nuevo (): Reserva un espacio en memoria y la asigna a una variable de tipo apuntador.

Leer (): Permite introducir un valor por el usuario a una variable

Mensaje (" "):Visualiza el mensaje que se encuentra entre comillas

Mostar (): Visualiza el contenido de una variable.

←: asignar un valor a una variable

Lista circular sencilla: es aquella en la que sus nodos se encuentran enlazados únicamente por una liga, y el último nodo de la lista apunta hacia el primer nodo de la lista.
Lista Ortogonal: es aquella en la que sus nodos se encuentran encadenados por cuatro ligas.

Keywords

Node: Pointer to the node whose memory address is at node

Node(value): The information field for the node pointed to by node.

Node(next): The field stores the memory address of the successor node to node. Depending on the type of list, a node can have more than one link.

Node(next) ← NULL: The field stores a null value to indicate that there is no successor to node.

Head: address where the first node in the list is located.
simple circular list: It is one in which its nodes are linked only by a link, and the last node in the list points to the first node in the list.

orthogonal list: It is the one in which its nodes are chained by four leagues.

New (): Reserves a space in memory and assigns it to a variable of type pointer.

Read (): Allows the user to introduce a value to a variable

Message (" "): Displays the message enclosed in quotation marks

Show (): Displays the content of a variable.

←: assign a value to a variable

Introducción

Un Tipo de Dato Abstracto (TDA) es un modelo que define datos u objetos y las operaciones que se pueden realizan sobre ellos. Y se denomina abstracto ya que la intención es que quien lo utiliza, no necesita conocer los detalles de la representación interna o bien el cómo están implementadas las operaciones, lo cual permite el encapsulamiento de los datos y las operaciones y gracias a esto se puede ocultar como ha sido implementado el TDA.

Desarrollo del tema

El programa se desarrolló mediante una solución integral que implementa tipos de datos abstractos (TDA) y visualización de datos (Graphviz) bajo el concepto de programación orientada a objetos (POO).

Un Tipo de Dato Abstracto es un modelo que define datos u objetos y las operaciones que se pueden realizan sobre ellos. Y se denomina abstracto ya que la intención es que quien lo utiliza, no necesita conocer los detalles de la representación interna o bien el cómo están implementadas las operaciones, lo cual permite el encapsulamiento de los datos y las operaciones y gracias a esto se puede ocultar como ha sido implementado el TDA.

Es por esto una práctica que nos provee un grado de abstracción que permite desacoplar al código que usa un TDA de aquel código que lo implementa.

Mediante los TDA se pueden crear listas que son estructuras de datos que son dinámicas, esto significa que adquieren espacio y liberan espacio a medida que se necesita. Son muy versátiles, pueden definirse estructuras más complejas a partir de las listas, como por ejemplo arreglos de listas. En algunas ocasiones los grafos se definen como listas de adyacencia. También se utilizan para las tablas de dispersión como arreglos de listas.

a. Estructuras lineales: Son aquellas en las que se definen secuencias como conjuntos de elementos entre los que se establece una relación de predecesor y sucesor. Las diferentes estructuras de datos basadas en este concepto se diferencian por las operaciones de acceso a los elementos y manipulación de la estructura. Existen tres estructuras lineales especialmente importantes: las pilas, las colas y las listas.
b. Estructuras no lineales. Son aquellas en las que no existe una relación de adyacencia entre sus elementos, es decir, un elemento puede estar relacionado con cero, uno o más elementos. Existen dos estructuras no lineales especialmente importantes: los árboles y los grafos.

Listas ortogonales:
Una lista ortogonal es aquella en la que sus nodos se encuentran encadenados por cuatro ligas, es decir, cada nodo se encuentra doblemente ligado en forma horizontal, y cada nodo se encuentra doblemente ligado en forma vertical. Una lista ortogonal puede ser implementada como lineal o circular. Este tipo de listas se puede utilizar para representar matrices.

Listas sencillas circular:
Una lista sencilla es aquella en la que sus nodos se encuentran enlazados únicamente por una liga, es decir, cada nodo apunta al siguiente nodo de la lista y cada nodo es apuntado por el nodo anterior de lista, a excepción del primer nodo y del último nodo de la lista. En una circular el último nodo de la lista apunta hacia el primer nodo de la lista.

Una lista se pueden efectuar operaciones por medio de algoritmos que se deben desarrollar de acuerdo con el tipo de lista. Algunas de las operaciones básicas que se pueden efectuar sobre una lista son:

a. Recorrido. Esta operación consiste en visitar todos los nodos que forman parte de una lista. Para recorrer todos los nodos de la lista es necesario posicionarse en el primer nodo de la lista y después avanzar hacia el nodo que apunte la liga siguiente y así sucesivamente hasta encontrar el fin de la lista.
b. Inserción. Esta operación consiste en agregar un nuevo nodo a una lista. La ubicación del nuevo nodo puede ser al inicio, al final o en cualquier posición dentro de la lista.
c. Borrado. Esta operación consiste en eliminar un nodo de la lista y redireccionar las ligas de los nodos antecesor y sucesor para el caso de un nodo que se encuentre en una posición intermedia. El borrado también se puede aplicar tanto al primer nodo de la lista como al último nodo de la lista.
d. Búsqueda. Esta operación consiste en recorrer todos los nodos de la lista desde el primer nodo para ir comparando el valor de cada nodo con el valor

El desarrollo del programa se costa de seis opciones que se muestran en el menú principal, las opciones son: Cargar archivos, procesar archivos, escribir archivo de salida, mostrar datos del estudiante, generar gráfica y salir.

a. Cargar archivos: en esta parte se lee un archivo con extensión y estructura xml en el cual se limitan las siguientes etiquetas:
   matrices: este será necesario para la lectura inicial del archivo, ya que será la etiqueta padre de todo.
   1. matriz: esta etiqueta será la que indica que una nueva matriz será creada para su respectivo análisis y únicamente puede estar dentro de la etiqueta matrices y puede tener los atributos:
      1.1. nombre: este contendrá el identificador de la matriz leída (se deberá validar la existencia de matrices con el mismo nombre, para mantener la consistencia de los datos).
      1.2. n: será la fila que tendrá la matriz, si los datos dentro de ella son mayor o menor a este atributo será error.
      1.3. m: será la columna que tendrá la matriz, si los datos dentro de ella son mayor o menor a este atributo será error.

   2. dato: esta etiqueta únicamente podrá estar entro de la etiqueta matriz y contendrán los valores respectivos a cada celda de la matriz, esta etiqueta puede tener los siguientes atributos.
      2.1. x: será la fila de la matriz y no puede ser mayor al atributo n de la matriz.
      2.2. y: será la columna de la matriz y no puede ser mayor al atributo m de la matriz

Figura I. Ordenamiento burbuja

Fuente: elaboración propia, 2021

Después de leer los datos se verifico que estuvieran en el orden correcto y de lo contrario se ordenaron por medio de un ordenamiento burbuja como se muestra en la figura No. 2.

Como se observa en la figura II esta función retorna una lista con los datos de la matriz, estos se envían a un procedimiento que crea

una matriz mediante una lista ortogonal para cada una de las matrices leídas y todas las listas ortogonales se guardaron en una lista circular junto con el nombre que le corresponde a cada una.

b. Procesar archivos: en esta opción se procesa la información cargada en la memoria y se muestra en consola el proceso que se va realizando como se muestra en la figura III. Para ostra el proceso paso a paso se implementó la librería time para realizar una pausa de un segundo entre cada proceso.

   Para procesar el archivo primero se genera una matriz binaria para cada una de los matices y luego se comparan las filas de la matriz binaria y si estas coincides se suman y se guardan en una tercera matriz llamada matriz reducida.

c. Escribir archivo de salida: En esta opción se solicita una runa especifica en la cual se guarda el archivo de salida, el cual al igual que el de entrada tiene extensión y formato xml, en este se imprimen los datos de la matriz reducida con las mismas etiquetas del archivo de entrada y agregando la etiqueta:

   1. frecuencia: esta etiqueta representa las veces que se encontró coincidencia en cada uno de los patrones de los grupos de registros utilizados para crear la matriz reducida y puede tener los atributos:
      1.1. g: será el número del grupo y contendrá el numero de filas que fueron sumadas en dicho grupo.

   El archivo de salida que se genera cuenta con el formato de la figura III. Para generar el archivo xlm de salida se debe importar la librería xml.etree.ElementTree y se generó por medio de una función que recibe una lista con los satos en la primera posición, los grupos en la segunda y en la tercera posición el número de columnas.

d. Mostrar datos del estudiante: en esta opción se muestran los datos del estudiante que desarrollo el proyecto.

e. Generar gráfica: al elegir esta opción se mostrará un menú con los nombres de las matrices que se encuentran guardas en la lista circular y se generar la grafica de la matriz que el usuario elija.

   Para generar la grafica se hizo usa de Graphviz el cual se instaló dentro de la IDE por medio del pip intall, la grafica que se genera es la de la figura VI.

Conclusiones

1. Para el desarrollo de este proyecto es necesario tener claro los conceptos de tipos de datos abstractos y visualización a través de graphiz.

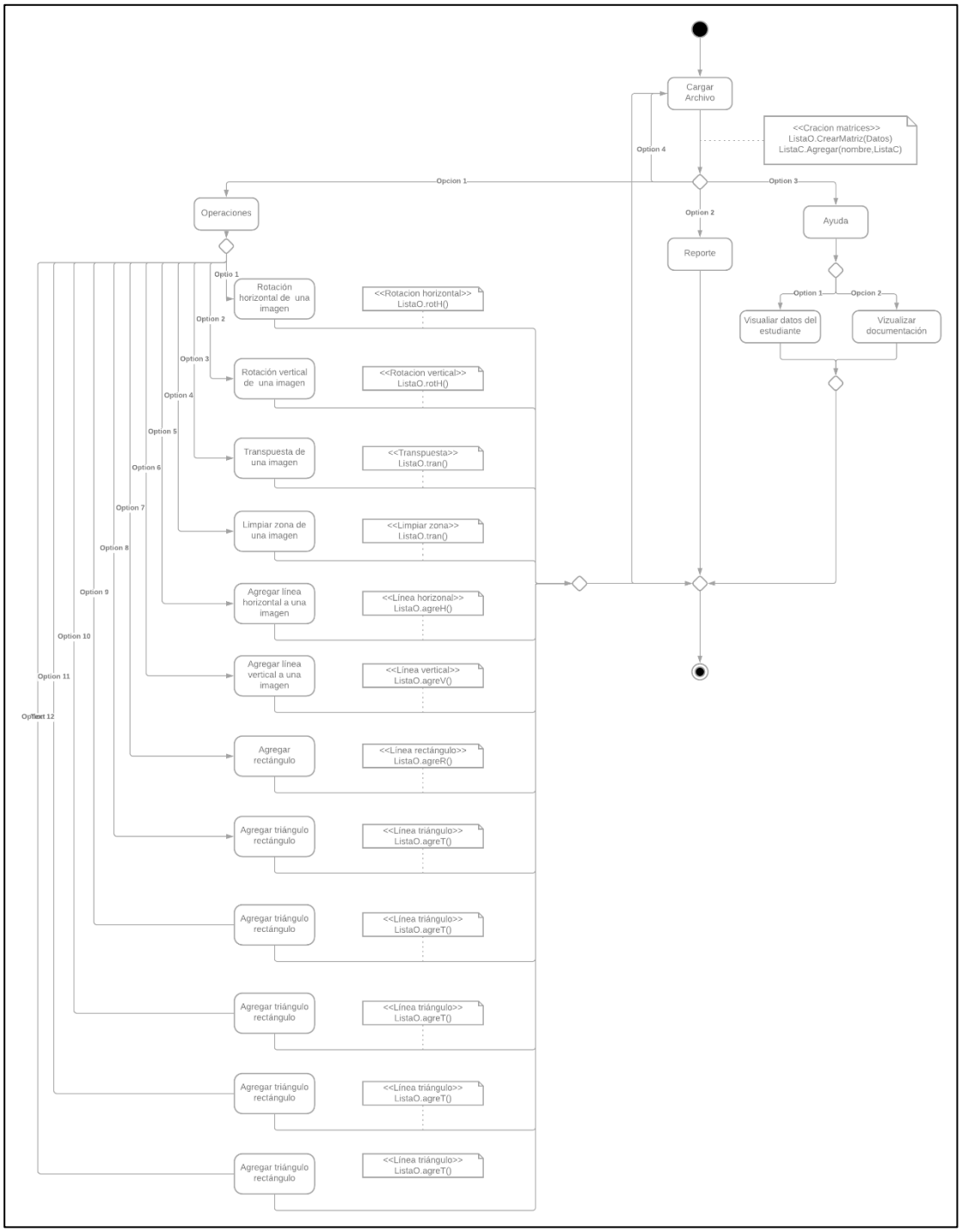
2. El uso de matrices ortogonales facilita la creación de una estructura de una matriz al trabajar con tipos de datos abstractos, además de que de esta manera es más fácil recorrerla y realizar operaciones con ella.

3. Es necesaria la implementación orientada a objetos en el desarrollo de este tipo de proyectos en donde se hace uso de TDA ya que hace más fácil la manipulación de los mismos.

4. Se debe tener en cuenta que el proyecto se desarrollo con los siguientes datos técnicos:
   Lenguaje utilizado: Python
   IDE utilizado: Visual Studio Code
   Sistema operativo: Windows 10(64 bits)

Apéndice

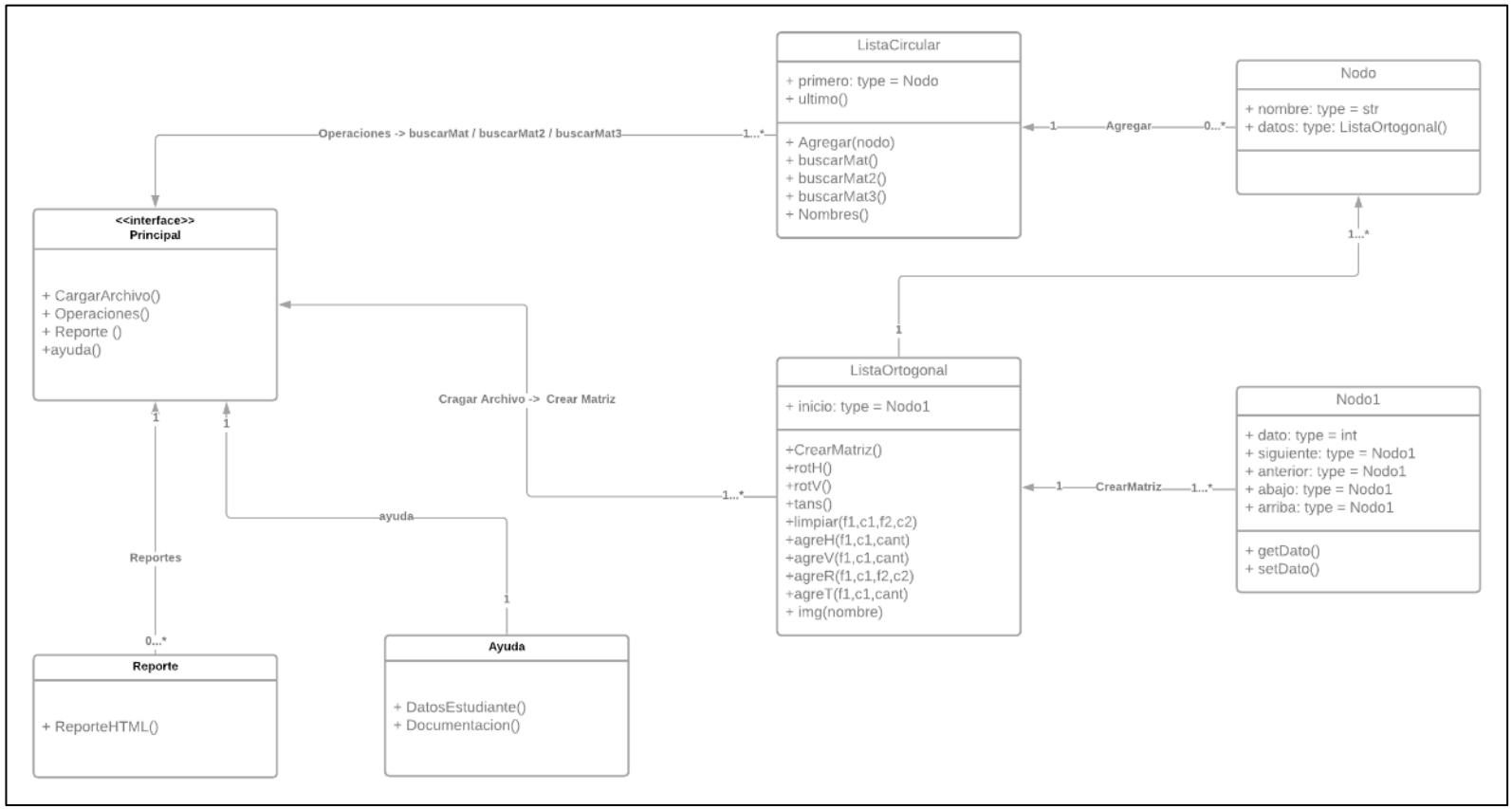
Diagrama de actividades:



Apéndice I. Diagrama de actividades

Fuente: elaboración propia, 2021

Diagrama de clases:



Apéndice II. Diagrama de clases

Fuente: elaboración propia, 2021