PROYECTO 1 – Reductor de Frecuencia de Señales

202100033 – Josue Daniel Chavez Portillo

Resumen

El proyecto consiste en la creación de un programa capaz de brindar una solución al inconveniente planteado por la facultad de ingeniería de la universidad de san Carlos de Guatemala.

El programa desarrollado como primer paso, contiene la lectura de un archivo xml para la importación de los datos de las señales y sus tiempos y amplitudes respectivas, con esta información el programa analiza en una serie de patrones, la señal adoptaun patron binario el cual se utiliza para agrupar las señales con el mismo patron binario y reduce las señales sumando los valores de misma fila y columna.

Para la elaboración del proyecto se hizo uso del lenguaje de programación Python, junto a las estructuras de programación secuenciales, cíclicas y condicionales. Para la parte de generar reporte, se hizo uso de la herramienta Grapvhiz, se implementó la manipulación de archivos XML utilizando Dom.Minidom y en la estructura de las listas haciendo uso de los conceptos de TDA aplicado a la memoria dinámica

Palabras clave

- TDA
- Memoria Dinamica

Abstract

The project consists of the creation of a program capable of providing a solution to the problem posed by the Faculty of Engineering of the University of San Carlos de Guatemala.

The program developed as a first step, contains the reading of an xml file for importing the data of the signals and their respective times and amplitudes, with this information the program analyzes in a series of patterns, the signal adopts a binary pattern which is used to group the signals with the same binary pattern and reduces the signals by adding the values of the same row and column.

The elaboration of the project, the Python programming language was used, together with the sequential, cyclical and conditional programming structures. For the part of generating the report, the Grapvhiz tool was used, the manipulation of XML files was implemented using Dom.Minidom and in the structure of the lists using the concepts of TDA applied to dynamic memory

Keywords

- *TDA*

- *Dynamic Memory*

Introducción

Para entender mejor la resolución del problema del programa debemos de entender diferentes conceptos para poder interpretar lo programado, como son las sintaxis básicas del lenguaje utilizado el cual fue Python, también las estructuras de programación que se utilizaron, lo que es la programación orientada a objetos y como utilizarla de la manera correcta y los errores frecuentes que se tienen a la hora de aplicarla a un programa, el como generar un reporte en este caso haciendo uso de graphviz el cual es un conjunto de herramientas de software para el diseño de diagramas, las lecturas de archivos usando minidom el cual es una implementación mínima de la interfaz Document Object Model con una API similar a la de otros lenguajes. Y el tema mas importante y talvez complicado de entender el cual es el TDA (Tipo de Dato Abstracto) el cual es un modelo matemático compuesto por una colección de operaciones definidas sobre un conjunto de datos para el modelo

Python

Python es un lenguaje de programación de alto nivel, es sencillo de leer y escribir debido a su similitud con el lenguaje humano. Además de esto se trata de un lenguaje multiplataforma de código abierto, y por lo tanto es gratuito lo cual permite el desarrollo de software sin ningún tipo de limite.

Como se menciona Python es un lenguaje de programación multiplataforma, lo cual permite desarrollar aplicaciones en cualquier sistema operativo con una facilidad asombrosa. Una gran cantidad de tecnología acepta como lenguaje Python debido a su sencillez y a su gran potencia para el tratamiento de datos, algo que sin duda ha hecho resurgir este lenguaje a nivel laboral, donde cada vez son más los usuarios que migran hacia este lenguaje.

Con esta información se concluye el por qué Python es uno de los lenguajes de programación más demandados en el mundo laboral. Debido a su relación con algunos de los campos con mayor relevancia de la actualidad, como la IA, el Machine Learning o el análisis de datos, se necesitan un gran número de programadores expertos en Python para desarrollar nuevas y emocionantes funciones.

Programación Orientada a Objetos

La Programación Orientada a Objetos (POO) es un paradigma de programación, es decir, un modelo o un estilo de programación que nos da unas guías sobre cómo trabajar con él. Se basa en el concepto de clases y objetos. Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.

“A lo largo de la historia, han ido apareciendo diferentes paradigmas de programación. Lenguajes secuenciales como COBOL o procedimentales como Basic o C, se centraban más en la lógica que en los datos. Otros más modernos como Java, C# y Python, utilizan paradigmas para definir los programas, siendo la Programación Orientada a Objetos la más popular.” (Canelo, 2020)

Con el paradigma de Programación Orientado a Objetos lo que buscamos es dejar de centrarnos en la

lógica pura de los programas, para empezar a pensar en objetos, lo que constituye la base de este paradigma. Esto nos ayuda muchísimo en sistemas grandes, ya que, en vez de pensar en funciones, pensamos en las relaciones o interacciones de los diferentes componentes del sistema.

Importancia del POO

La Programación Orientada a objetos permite que el código sea reutilizable, organizado y fácil de mantener. Sigue el principio de desarrollo de software utilizado por muchos programadores DRY (Don't Repeat Yourself), para evitar duplicar el código y crear de esta manera programas eficientes. Además, evita el acceso no deseado a los datos o la exposición de código propietario mediante la encapsulación y la abstracción, de la que hablaremos en detalle más adelante.

Clases, Objetos e Instancias

Una clase es una plantilla. Define de manera genérica cómo van a ser los objetos de un determinado tipo. Por ejemplo, una clase para representar a animales puede llamarse "animal" y tener una serie de atributos, como "nombre" o "edad" y una serie con los comportamientos que estos pueden tener, como caminar o comer, y que a su vez se implementan como métodos de la clase (funciones).

Un ejemplo sencillo de un objeto, como decíamos antes, podría ser un animal. Un animal tiene una edad, por lo que creamos un nuevo atributo de "edad" y, además, puede envejecer, por lo que definimos un nuevo método. Datos y lógica. Esto es lo que se define en muchos programas como la definición de una clase, que es la definición global y genérica de muchos objetos.

Principios del POO

Encapsulación: contiene toda la información importante de un objeto dentro del mismo y solo expone la información seleccionada al mundo exterior.

Abstracción: contiene toda la información importante de un objeto dentro del mismo y solo expone la información seleccionada al mundo exterior.

Herencia: define relaciones jerárquicas entre clases, de forma que atributos y métodos comunes puedan ser reutilizados. Las clases principales extienden atributos y comportamientos a las clases secundarias.

Polimorfismo: consiste en diseñar objetos para compartir comportamientos, lo que nos permite procesar objetos de diferentes maneras. Es la capacidad de presentar la misma interfaz para diferentes formas subyacentes o tipos de datos.

Alrededor de estos principios de la programación orientada a objetos se construyen muchas cosas. Por ejemplo, los Principios SOLID, o los Patrones de diseño.

TDA

Un Tipo de dato abstracto (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 cómo 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.

Ejemplos del uso de TDA:

- Listas
- Pilas
- Colas
- Conjuntos
- Diccionarios
- Grafos

TDA Lista

Una lista se define como una serie de N elementos E1, E2, ..., EN, ordenados de manera consecutiva, es decir, el elemento Ek (que se denomina elemento k-decimo) es previo al elemento Ek+1. Si la lista contiene 0 elementos se denomina como lista vacía.

Las operaciones que se pueden realizar en la lista son:

- Insertar un elemento
- Borrar un elemento
- Buscar un elemento
- Verificar lista vacía

Una manera simple de implementar una lista es utilizando un arreglo. Sin embargo, las operaciones de inserción y borrado de elementos en arreglos son ineficientes, puesto que para insertar un elemento en la parte media del arreglo es necesario mover todos los elementos que se encuentren delante de él, para hacer espacio, y al borrar un elemento es necesario mover todos los elementos para ocupar el espacio desocupado. Una implementación más eficiente del TDA se logra utilizando listas enlazadas.

A continuación, se presenta una implementación del TDA haciendo uso de listas enlazadas y sus operaciones.

- **EstáVacía ():** Devuelve un valor verdadero si esta vacía
- **Insertar ():** Inserta un elemento al final o al inicio
- **Buscar ():** Devuelve la posición del elemento
- **Eliminar ():** Elimina el elemento

Tuples

"Es una colección de datos cuyo orden es inalterable, o sea, son elementos ordenados en una secuencia específica y que posee importancia. En Python, los tuples se escriben entre paréntesis." (HETPRO, 2020)

Grapvhiz

Graphviz es un programa de visualización gráfica de fuente abierta. La visualización de gráficos es una forma de representar información estructural como diagramas de redes y gráficos abstractos. Tiene aplicaciones importantes en redes, bioinformática, ingeniería de software, diseño web y de bases de datos, aprendizaje automático y en interfaces visuales para otros dominios técnicos.

Figura III. Ejemplo de Grapvhiz
Fuente: (Grapvhiz, 2021)

Los programas de diseño Graphviz toman descripciones de gráficos en un lenguaje de texto simple y hacen diagramas en formatos útiles, como imágenes y SVG para páginas web; PDF o Postscript

para incluir en otros documentos; o mostrar en un navegador gráfico interactivo. Graphviz tiene muchas funciones útiles para diagramas concretos, como opciones de colores, fuentes, diseños de nodos tabulares, estilos de línea, hipervínculos y formas personalizadas.

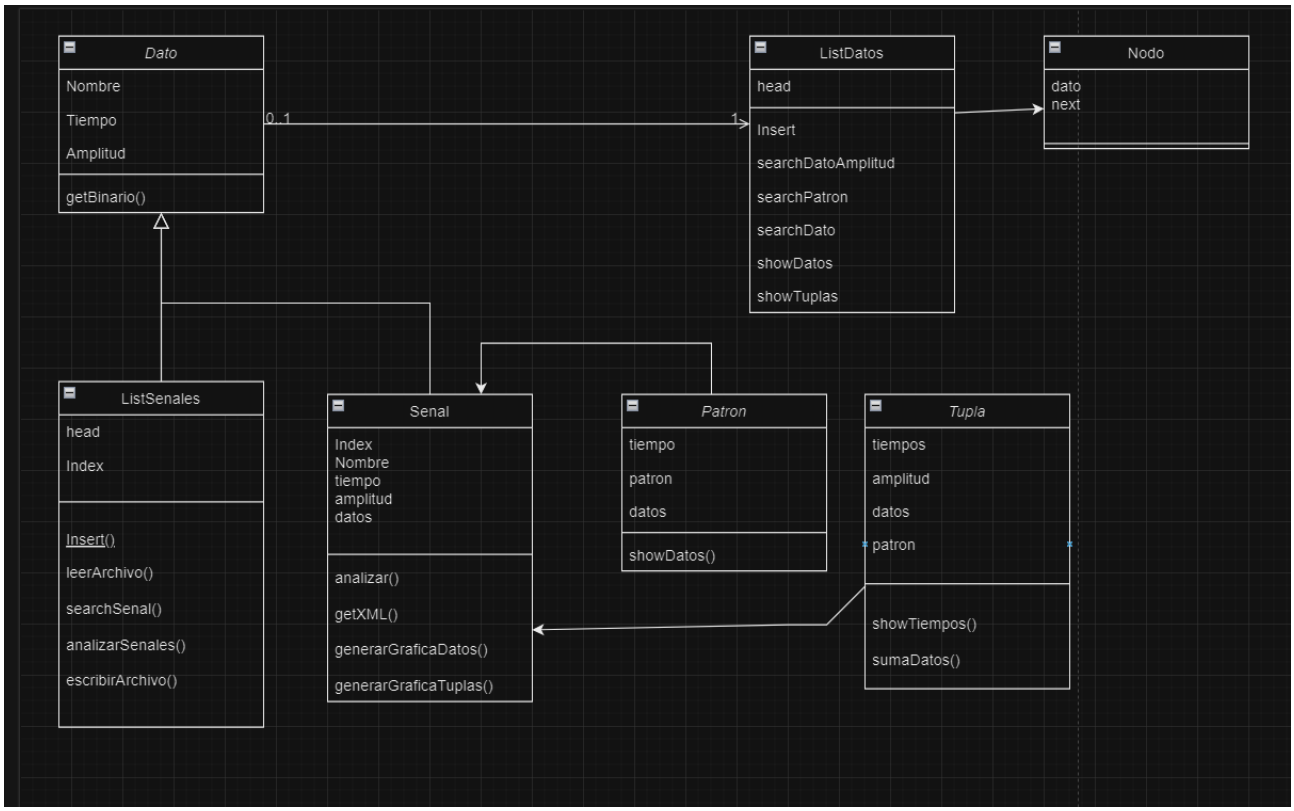
Conclusiones

• La Programación Orientada a Objetos es actualmente el paradigma que más se utiliza para diseñar aplicaciones y programas informáticos.
• Python es un lenguaje fácil de aprender e ideal para aquellos programadores que se están iniciando
• El TDA permite el desarrollo independiente de módulos, facilita la depuración y encapsulación

Referencias bibliográficas

- Canelo, M. M. (02 de Noviembre de 2020). Profile. Obtenido de Profile: https://profile.es/blog/que-es-laprogramacion-orientada-a-objetos/
- • Grapvhiz. (10 de Agosto de 2021). Grapvhiz. Obtenido de https://graphviz.org/
- • HETPRO. (16 de Junio de 2020). HETPRO. Obtenido de HETPRO: https://hetprostore.com/TUTORIALES/tuples-en-python6-colecciones/#:~:text=Un%20tuple%20es%20una%20colecci%C3%B3n,tuples%20se%20escriben%20entre%20par%C3%A9ntesis.

Anexos


Dato
Nombre
Tiempo
Amplitud
getBinario()
0..1
1
ListDatos
head
Insert
searchDatoAmplitud
searchPatron
searchDato
showDatos
showTuplas
Nodo
dato
next
ListSenales
head
Index
Insert()
leerArchivo()
searchSenal()
analizarSenales()
escribirArchivo()
Senal
Index
Nombre
tiempo
amplitud
datos
analizar()
getXML()
generarGraficaDatos()
generarGraficaTuplas()
Patron
tiempo
patron
datos
showDatos()
Tupla
tiempos
amplitud
datos
patron
showTiempos()
sumaDatos()