Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, Escuela de Vacaciones 2023.

Proyecto-----IPCmusic-----TDA-----Grupo #8

202000549 – Luis Daniel Salán Letona
202000363 – Adriana Lucia Ojeda Rivas
202100081 – Javier Andrés Monjes Solórzano
202000416– Jonathan Eduardo Santos Letona
202004769 – Luis Pablo de Jesús López Carrera

Resumen

Se desarrollo de un reproductor de audio capaz de ordenar y clasificar los audios extraídos de un archivo XML según el álbum y al artista que pertenece dicha canción. Para ello, se implementaron estructuras de datos como "Listas doblemente enlazadas" y su variante "Lista circular doblemente enlazada". Las estructuras de datos se basaron en la teoría de grafos, basándose en las características de nodos y apuntadores que emulan un grafo dirigido.

Se implementó TDA para poder manejar de una manera eficiente y optimizar el rendimiento del software respecto a la utilización de la memoria de la máquina en la que se desplegara. La clave del manejo de las estructuras de datos en los objetos creados (POO) fue vital para poder optimizar la organización de los datos al abstraer los objetos de la vida real a un ambiente informático.

Palabras clave

- Python: Lenguaje de programación de alto nivel
- Objects: Colección de datos y comportamientos dentro del código
- Lista: Tipo de estructura de datos que permite almacenar información dentro de la misma
- XML: lenguaje de marcado extensible que permite almacenar información de manera compartible
- Graphviz: herramienta de software para el diseño de diagramas

Abstract

This report presents the development of an audio player capable of ordering and classifying audio extracted from an XML file according to the album and artist to which the song belongs. To do this, data structures such as "doubly linked lists" and its variant "doubly linked circular list" were implemented. The data structures were based on graph theory, based on the characteristics of nodes and pointers that emulate a directed graph.

TDA was implemented to be able to handle efficiently and optimize the software performance with respect to the use of memory of the machine on which it was deployed. The key to handling data structures in the created objects (OOP) was essential to be able to optimize the organization of data by abstracting real-world objects to a computer environment.

Keywords

- *Python: High level programming language*
- *Objects: Collection of data and behaviors within the code*
- *List: Type of data structure that allows information to be stored within it.*
- *XML: extensible markup language that allows information to be stored in a sharable manner*
- *Graphviz: software tool for the design of diagrams*

Introducción

Python es un lenguaje de programación de alto nivel interpretado que se ha vuelto popular en los últimos años. Se caracteriza por su legibilidad, facilidad de aprendizaje y versatilidad, lo que lo hace adecuado para una amplia gama de aplicaciones.

Python se utilizó para las siguientes tareas:

- Almacenamiento de datos: Python se utilizó para crear listas enlazadas y listas circulares, que permiten almacenar objetos y datos de forma eficiente.
- Interfaz gráfica de usuario: Python se utilizó para desarrollar interfaces gráficas de usuario (GUI) con el módulo tkinter.
- Lectura de archivos XML: Python se utilizó para leer archivos XML con el módulo ElementTree.
- Reproducción de audio: Python se utilizará para reproducir archivos de audio con el módulo Pygame.

Desarrollo del tema

Se desea crear un reproductor de música basado en el lenguaje de programación Python para el cual se realizó mediante la creación de listas enlazadas. Una lista enlazada es una de las estructuras de datos fundamentales, pueden ser utilizadas para implementar otras estructuras de datos. Su funcionamiento se basa en una secuencia de nodos en los que se podrán guardar objetos con atributos o elementos de otros tipos, además también pueden contar con punteros o enlaces a un nodo anterior o al nodo siguiente.

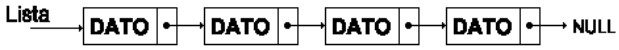

Figura 1. Lista Enlazada

Fuente: [Conclase.net. (2023). Listas abiertas]. https://conclase.net/c/edd/cap1

Además, también es posible que en el nodo final se pueda colocar un nodo que apunte al inicio de la lista teniendo así una lista circular.

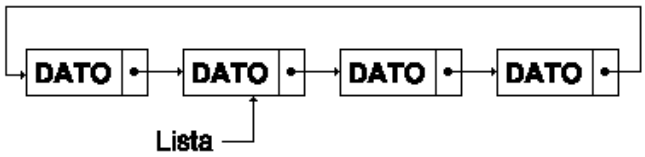

Figura 2. Lista Circular.

Fuente: [Conclase.net. (2023). Listas circulares]. https://conclase.net/c/edd/cap4

Para la elaboración del reproductor se ocupará el uso de listas doblemente enlazada lo que facilitaría el manejo de información, su concepto es igual a una lista enlazada sencilla, la diferencia radica en que ésta tendrá dos apuntadores una hacia el nodo siguiente y otro puntero al nodo anterior lo que permite desplazarse entre elementos de la lista.

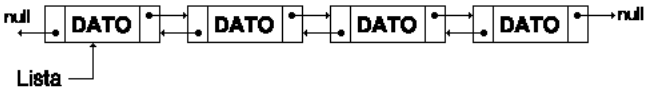

Figura 3. Lista Doblemente Enlazada.

Fuente: [Conclase.net. (2023). Listas doblemente enlazadas]. https://conclase.net/c/edd/cap5

El manejo de listas enlazadas en este proyecto es de suma importancia ya que el funcionamiento del reproductor dependerá de las mismas tanto para la carga masiva de información utilizando archivos xml y almacenar su contenido, como para crear listas nuevas de reproducción hechas por el usuario.

Para el funcionamiento óptimo de las listas es esencial contar con una clase de tipo nodo que será la que contendrá los atributos de la lista contenidos en un constructor, luego se requerirá de una clase lista que en su constructor almacenará el nodo inicio y final de la cola, por último, se requerirá del método para insertar información y así llenar las listas, pero ¿de dónde se obtendrá dicha información?, como se mencionó anteriormente para la carga masiva de información se utilizará un archivo con extensión xml.

XML es el acrónimo de Extensible Markup Language, es un lenguaje marcado es decir un conjunto de códigos que pueden ser utilizados en análisis de datos o lectura de textos elaborados por computadoras o usuarios.

Básicamente el XML se centra en la simplicidad y facilidad de uso para manejo de información por lo cual se utilizará para la carga masiva a la biblioteca del reproductor.

Cargada la biblioteca con la música del usuario, se podrá visualizar todo el contenido de ésta mediante un reporte generado por graphviz.

Graphviz es un programa de visualización gráfica de información estructural, consiste en un lenguaje de descripción de gráficos llamado DOT, un conjunto de herramientas y librerías que pueden generar o procesar archivos DOT.

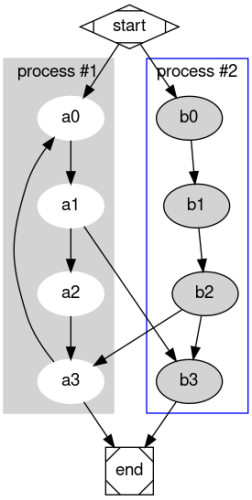

Figura 4. Ejemplo de Graphviz.

Fuente: graphviz.org, 2023.

Para comodidad del usuario se utilizó el módulo tkinter, es la interfaz gráfica (GUI) estándar de Python, tkinter cuenta con widgets que serán las piezas que forman el contenido visual para el usuario, se utilizaron ventanas, etiquetas, botones y se insertaron imagines en dichos botones para una experiencia más intuitiva y agradable para el usuario al ver imágenes con las que estará más familiarizado.

Figura 5. Interfaz Gráfica (GUI).

Fuente: Elaboración propia, 2023.

Conclusiones

Python es un lenguaje de programación versátil y potente que es adecuado para una amplia gama de aplicaciones. En este proyecto, Python se utilizó para tareas como el almacenamiento de datos, el desarrollo de GUI, la lectura de archivos XML y la reproducción de archivos de audio.

Python ofrece múltiples herramientas que le pueden facilitar la vida a las personas tanto para usuario como para desarrolladores.

Actualmente la innovación tecnológica constante obliga a todos por igual a estar actualizados lo que conlleva a manejos de información masiva mediante sitios web y aplicaciones digitales por lo que los archivos XML son de suma importancia para el manejo de tanto contenido.

Para facilitar la comprensión de contenido es necesario contar con recursos gráficos, para el cual el programa graphviz realiza un estupendo trabajo sobre todo tratándose de estructura de datos y redes.

Una interfaz gráfica fácil de comprender y amigable para el usuario puede ser la clave en cualquier proyecto sin importar su enfoque, ya que de esto puede depender el éxito de la aplicación desarrollada

Referencias bibliográficas

Bakliwal, S. (2023). Learn How to Create an MP3 Music Player in Python. Data Flair. https://data-flair.training/blogs/python-mp3-player/

Cepalia, A. (2023). Working with linked lists in Python. Real Python. Recuperado de https://realpython.com/linked-lists-python/.

Conclase.net. (2023). Estructuras de datos dinámicas. En: Conclase.net. [En línea]. Recuperado de https://conclase.net/c/edd/cap0

Graphviz (2023). https://graphviz.org/

Ivan de Souza, (2012). XML: ¿qué es y para qué sirve este lenguaje de marcado? https://rockcontent.com/es/blog/que-es-xml/

ANEXOS

Para poder comprender mejor el proyecto se puede dirigir a los siguiente:

REPOSITORIO

MANUAL USUARIO

MANUAL TECNICO

Requerimientos:

- Python 3.11.5 o Superior
- Graphviz 0.20 o superior
- PySide6 6.5.2 o superior
- Pillow 10.1.0
- Pygame

Estructura del proyecto:

```
IPC2_Proyecto1Diciembre_-Grupo8
├── Documentación
│   ├── Manual de Usuario
│   │   ├── README.md
│   └── Manual Tecnico
│   │   ├── README.md
│   └── Documentación
│   └── Proyecto1IPC2 - Diciembre.pdf
├── GUI
│   ├── __init__.py
│   └── app.py
├── iconos
│   ├── 15 imagenes png
├── IMG
│   ├── archivos jpg para documentación
├── method
│   ├── __init__.py
│   └── readxml.py
├── MisListas
│   ├── aquí se guardan las listas en xml
├── objects
│   ├── __init__.py
│   └── objects.py
├── Test
│   ├── Musica
│   │   ├──archivos mp3 y jpg
│   ├── biblioteca.xml
│   ├── MiLista.xml
├── thread
│   ├── __init__.py
│   └── thread.py
├── __init__.py
├── app.py
├── library.dot
├── main.py
├── objects.py
├── README.md
```

Diagrama De Clases
Grupo #8|December 18,2023