Proyecto 1 IPC2 Primer semestre ciclo 2021

201701059 – Luis Amilcar Morales Xón

Resumen

Un proyecto desarrollado en Python que evalúa los conocimientos y la implementación de temas en específico, tales como: Manejo de archivos XML, tipos de datos abstractos, graficar con la ayuda de la herramienta graphviz.
El archivo XML es un estándar utilizado a nivel global para almacenar, publicar e intercambiar todo tipo de información, proporciona portabilidad y facilita la gestión de la información a través de distintas plataformas. Permite que diversas aplicaciones de datos puedan funcionar de forma independiente.
Los Tipos de Datos Abstractos (TDA) permiten utilizar nuevos tipos de datos que se definirán especificando sus posibles valores y las operaciones que los manipulan. Cada operación constituye una abstracción funcional. Un tipo de dato abstracto es un tipo definido por el usuario que tiene un conjunto de valores y un conjunto de operaciones.
Presentar la información de manera visual al usuario aumenta mejor la compresión de la información que se quiere transmitir, graphviz ofrece un motor potente para generar todo tipo de gráficas.

Palabras clave
Matriz
Grafo
Listas
TDA
XML

Abstract

A project developed in Python that assesses the knowledge and implementation of specific topics, such as: Handling XML files, abstract data types, graphing with the help of the graphviz tool.
The XML file is a standard used globally to store, publish and exchange all kinds of information, it provides portability and facilitates the management of information through different platforms. It allows various data applications to function independently.
Abstract Data Types (ADT) allow the use of new data types that will be defined by specifying their possible values and the operations that manipulate them. Each operation constitutes a functional abstraction. An abstract data type is a user-defined type that has a set of values and a set of operations.
Presenting the information visually to the user increases better the compression of the information to be transmitted, graphviz offers a powerful engine to generate all kinds of graphs

Keywords
Matrix
Graph
Lists
ADT
XML

Introducción

Un proyecto que busca mostrar la facilidad de manejo de información y la convención de los archivos XML, en estos archivos viene los datos de una o más matrices matemáticas las cuales se someterán a procesos y serán cargadas y almacenadas en memoria con la ayuda de los tipos de datos abstractos. Al final del procesamiento, el programa es capaz de generar una gráfica que representa la matriz seleccionada previamente por el usuario para tener una percepción visual. El proceso al que es sometido cada matriz es a la reducción de filas según un patrón de acceso binario, el resultado final se puede plasmar en un archivo de salida XML con el mismo formato que tiene el archivo de entrada.

Desarrollo del tema

EL proyecto se realizó en el lenguaje Python y consta de un menú que permite la manipulación de los objetos creados a partir de una fuente XML las opciones que se manejan en el programa son los siguientes:

1 carga de Archivos: En este apartado se permite el ingreso de una dirección hacía la dirección donde se encuentra ubicado el archivo XML

2 procesamiento: Esta opción será la encargada de procesar la información cargada en memoria, durante el proceso se mostrarán mensajes que le informaran al usuario lo que va sucediendo en cada paso.

3 escribir Archivo de Salida: En este apartado se procede a escribir en formato XML la información que se produce en la matriz salida.

4 mostrar Los datos del Estudiante: En esta opción se muestra el nombre y carnet del desarrollador

5 generar Grafica: En esta opción podemos generar la gráfica del archivo de entrada o la gráfica de una matriz en específico

6 salir: Con esta opción podemos finalizar la ejecución del programa

XML: ¿Qué es y para qué sirve este lenguaje de marcado?

Detrás del diseño y el texto de los sitios web, siempre hay un lenguaje propio y uno de ellos es el XML. Este acrónimo significa Extensible Markup Language, que 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 de prolog consiste en metadatos administrativos, como declaración XML, instrucción de procesamiento opcional, declaración de tipo de documento y comentarios. La parte de body se compone de dos partes: estructurales y de contenido (presente en los textos simples).

XML simplifica el intercambio de datos

Tanto los sistemas informáticos como las bases de datos contienen información en formatos incompatibles.

Los datos XML se almacenan en formato de texto simple, lo que nos posibilita una forma independiente de almacenar datos. Esto facilita mucho la creación de datos que pueden ser compartidos por diferentes aplicaciones.

Uno de los desafíos más difíciles para los desarrolladores es intercambiar datos entre sistemas incompatibles a través de Internet. El intercambio de datos como XML reduce en gran medida esta complejidad porque los datos pueden ser leídos por diferentes aplicaciones incompatibles.

XML simplifica el cambio de plataforma

La actualización a nuevos sistemas (plataformas de hardware o software) lleva mucho tiempo. Se deben convertir grandes cantidades de datos y los datos incompatibles a menudo se pierden.

Los datos XML se almacenan en formato de texto. Esto facilita la expansión o actualización a nuevos sistemas de información, nuevas aplicaciones o nuevos navegadores sin pérdida de datos.

El XML aumenta la disponibilidad de datos
Diferentes aplicaciones pueden acceder a tus datos, no solo en páginas HTML, sino también en fuentes de datos XML.
Con el XML, tus datos pueden estar disponibles para todos los tipos de «máquinas de lectura» (computadoras de mano, máquinas de voz, feeds de noticias, etc.) y, además, facilita la accesibilidad para personas con capacidades diferentes, por ejemplo, no videntes.

Tipos de datos abstractos.

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 cómo estén 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.

En este capítulo se estudiarán TDA básicos como lo son las *listas*, *pilas* y *colas*, y se mostrarán algunos usos prácticos de estos TDA.

TDA lista

Una *lista* se define como una serie de *N* elementos E_1, E_2, ..., E_N, ordenados de manera consecutiva, es decir, el elemento E_k (que se denomina *elemento k-ésimo*) es previo al elemento E_{k+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 en la posición *k*, borrar el k-ésimo elemento, buscar un elemento dentro de la lista y preguntar si la lista está 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.

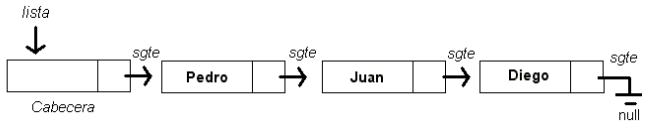
A continuación, se presenta una implementación en Java del TDA utilizando listas enlazadas y sus operaciones asociadas:

- **estaVacia()**: devuelve *verdadero* si la lista esta vacía, falso en caso contrario.
- **insertar(x, k)**: inserta el elemento *x* en la k-ésima posición de la lista.
- **buscar(x)**: devuelve la posición en la lista del elemento *x*.
- **buscarK(k)**: devuelve el k-ésimo elemento de la lista.
- **eliminar(x)**: elimina de la lista el elemento *x*.

En la implementación con listas enlazadas es necesario tener en cuenta algunos detalles importantes: si solamente se dispone de la referencia al primer elemento, el añadir o remover en la primera posición es un caso especial, puesto que la referencia a la lista enlazada debe modificarse según la operación realizada. Además, para eliminar un elemento en particular es necesario conocer el elemento que lo antecede, y en este caso, ¿qué pasa con el primer elemento, que no tiene un predecesor?

Para solucionar estos inconvenientes se utiliza la implementación de lista enlazada con nodo cabecera. Con esto, todos los elementos de la lista tendrán un elemento previo, puesto que el previo del primer

elemento es la cabecera. Una lista vacía corresponde, en este caso, a una cabecera cuya referencia siguiente es *null*.



¿Qué es la programación orientada a objetos?

La programación Orientada a objetos se define como un paradigma de la programación, una manera de programar específica, donde se organiza el código en unidades denominadas clases, de las cuales se crean objetos que se relacionan entre sí para conseguir los objetivos de las aplicaciones.

La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.

Clases en POO: las clases son declaraciones de objetos, también se podrían definir como abstracciones de objetos. Esto quiere decir que la definición de un objeto es la clase. Cuando programamos un objeto y definimos sus características y funcionalidades en realidad lo que estamos haciendo es programar una clase.

Propiedades en las clases: Las propiedades o atributos son las características de los objetos. Cuando definimos una propiedad normalmente especificamos su nombre y su tipo. Nos podemos hacer a la idea de que las propiedades son algo así como variables donde almacenamos datos relacionados con los objetos.

Métodos en las clases: Son las funcionalidades asociadas a los objetos. Cuando estamos programando las clases las llamamos métodos. Los métodos son como funciones que están asociadas a un objeto.

Objetos en POO: Los objetos son ejemplares de una clase cualquiera. Cuando creamos un ejemplar tenemos que especificar la clase a partir de la cual se creará. Esta acción de crear un objeto a partir de una clase se llama instanciar (que viene de una mala traducción de la palabra instace que en inglés significa ejemplar).

Conclusiones

Es imprescindible entender el comportamiento de una estructura abstracta para poder manipularlo y principalmente poder representarlo dentro de un lenguaje de programación ya que esto permitirá poder modelar cualquier información que se presente en el mundo real y convertirlo en datos que la computadora sea capaz de entender, así como el programador también es capaz de hacerlo

Referencias bibliográficas

Ignacio Onetto, A. ¿Por qué el archivo XML es importante? Nubox: Software para empresas y contadores. Recuperado de: https://blog.nubox.com/software/por-que-el-formato-xml-es-importante#:~:text=Es%20un%20estándar%20abierto%2C%20flexible,puedan%20funcionar%20de%20forma%20independiente.

De Souza, I. (Jul 12, 2019). XML ¿que es y para qué sirve este lenguaje de marcado? Rockcontent: Blog. Recuperado de: https://rockcontent.com/es/blog/que-es-xml/

Alvarez, M. Angel. (Dic 11, 2019). ¿Qué es la programación orientada a objetos?. Desarrolloweb.com: Teoría de la programación orientada a objetos. Recuperado de: https://desarrolloweb.com/articulos/499.php

Anexo