PROYECTO ENSAMBLADORA DIGITAL INTELLIGENCE

202010223 – Luis Angel Barrera Velásquez

Resumen

Se realizo este proyecto con el objetivo de brindar una solución a la problemática de ensamblar de manera óptima y sencilla muchos productos a través de un software de simulación el cual se le debe cargar dos archivos XML y al ser procesado genera otro archivo con el mismo formato el cual se puede cargar directamente a alguna maquina real.
La solución para la problemática se realizó a través del lenguaje de programación Python y al procesar archivos en formato XML, procesar dichos archivo a través de librerías propias de Python, calcular el procedimiento de ensamblaje de productos de manera optima y automatizada con tan solo cargar dos archivos.
Es importante reconocer que para la solución fueron utilizados conceptos de Tipos de Datos Abstractos (TDA) para poder elaborar de manera dinámica una matrices de m columnas y n filas para una mejor interpretación y manejo de datos con el uso de listas enlazadas, nodos y la utilización de Programación orientada a objetos (POO).

Palabras clave
Formato
Lista
Enlazada
Nodo
Datos

Abstract

This project was carried out with the aim of providing a solution to the problem of optimally and easily assembling many products through simulation software which must be loaded with two XML files and when processed generates another file with the same format which can be loaded directly to a real machine.
The solution to the problem was carried out through the Python programming language and when processing files in XML format, processing said files through Python's own libraries, calculating the product assembly procedure in an optimal and automated way with just loading two records.
It is important to recognize that for the solution concepts of Abstract Data Types (ADT) were used to dynamically elaborate a matrix of m columns and n rows for a better interpretation and handling of data with the use of linked lists, nodes and the use Object Oriented Programming (OOP).

Keywords
Format
List
Linked
Node
Data

Introducción

La realización de este proyecto se tomó en cuenta diferentes conceptos de programación para poder dar una solución acertada y efectiva del problema inicial para esto fue muy importante la correcta interpretación de los datos de archivo de entrada y la correcta estructuración de los datos obtenidos.

Este ensayo tiene el propósito de proporcionar al lector un panorama claro de los conceptos utilizados para la elaboración de la solución esto para entender el funcionamiento del programa y si se necesita realizar alguna actualización saber que conceptos son necesarios emplear para modificar la lógica de programación.

Desarrollo del tema

a. Archivo XML

XML (Extensible Mark up Language) es un lenguaje de marcado que codifica los datos en texto plano. Permitiendo así que estos puedan ser legibles tanto por parte de máquinas como por personas, de manera análoga a los JSON. Actualmente es un lenguaje que es usado por múltiples programas para almacenar y transmitir datos estructurados. A diferencia de los archivos JSON o Excel no existe una forma fácil de importar los XML en Python, por lo que requiere algo más de trabajo.

Los archivos XML se componen de etiquetas que nos aportan datos e información que queremos procesar. Estas etiquetas pueden estar de forma individual o anidadas. Habitualmente un fichero XML incluye mucha información y debe de ser procesada correctamente por el usuario, en este caso el desarrollador. Cuanto más grande sea un fichero XML nos estará indicando que más información trae.

Es importante mencionar que en este proyecto fue implementado un case insensitve para la lectura del archivo XML esto quiere decir que no importa si las etiquetas están escritas con mayúsculas, minúsculas o mixtas estas igualmente van a ser reconocidas por el programa y guardado sus datos para un futuro procesamiento de estos según los gustos del usuario

b. Matriz mxn

Se denomina matriz a todo conjunto de números o expresiones dispuestos en forma rectangular, formando filas y columnas.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mj} & \cdots & a_{mn} \end{pmatrix}$$

Figura 1. Matriz mxn
Fuente: https://www.superprof.es/diccionario/matematicas/algebralineal/matriz.html

Cada uno de las posiciones de que consta la matriz se denomina elemento. Un elemento se distingue de otro por la posición que ocupa, es decir, la fila y la columna a la que pertenece.

El número de filas y columnas de una matriz se denomina dimensión de una matriz. Así, una matriz será de dimensión: 2x4, 3x2, 2x5,... Si la matriz tiene el mismo número de filas que de columna, se dice que es de orden: 2, 3, ... El conjunto de matrices de n filas y m columnas se denota por Amxn o (aij), y un elemento cualquiera de la misma, que se encuentra en la fila i y en la columna j, por aij.

Cabe mencionar que en este proyecto fue utilizado el concepto de matriz armando una lista enlazada (explicada mas adelante) con varias listas enlazadas en cada uno de los nodos esto para simular la condicional de la matriz y sea mas organizado el procesamiento de la misma para las funcionalidades del algoritmo requerido para la solución.

c. Estructura dinámica de datos

son estructuras que en este caso para Python pueden ser de distinto, organizadas de alguna manera que se necesite y estas a su vez a medida que se van introduciendo más datos se van agregando de manera automática y sin ocupar mucha memoria ya que a medida de que vayamos utilizando más elementos en la estructura así aumentara la memoria utilizada sin reservar espacios en memoria que no se estén utilizado, esto ayuda a que el programa funcione de una manera optimizada y fluida ya que no consumirá recursos en exceso del equipo donde se esté ejecutando el software.

d. Listas simplemente enlazadas

Durante la realización de la solución de este proyecto fue implementado la utilización de listas enlazadas para una estructuración de los datos de manera dinámica y acorde a lo que se requiera que realice en software sin consumir memoria de más. Para esto se requiere la utilización de diferentes elementos más como los nodos y apuntadores los cuales posteriormente serán explicados, pero se iniciara explicando el concepto de lista enlazada:

En ciencias de la computación, 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.

Es una lista enlazada de nodos, donde cada nodo tiene un único campo de enlace. Una variable de referencia contiene una referencia al primer nodo, cada nodo (excepto el último) enlaza con el nodo siguiente, y el enlace del último nodo contiene NULL para indicar el final de la lista. Aunque normalmente a la variable de referencia se la suele llamar top, se le podría llamar como se desee.

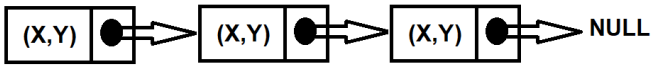

Figura 2. Ilustración de nodos de área
Fuente: Elaboración propia

Para este proyecto fue utilizado una lista simplemente enlazada para guardar cada uno de las posiciones de las líneas de producción indicadas en el archivo XML de entrada esto para que sin ningún problema se puede acceder a algún nodo en específico a través de un método de búsqueda y cuidando que siempre que se inserte un nuevo nodo este apunte a NULL indicando como se indicó en la teoría la finalización de la lista.

e. Listas doblemente enlazadas

Una lista de doble enlace contiene dos punteros conectados a la nodo anterior y siguiente nodo en la lista vinculada. Tenemos que almacenar los datos y dos punteros para cada nodo en la lista vinculada.

El puntero anterior del primer nodo es nulo y siguiente puntero del último nodo es nulo para representar el final de la lista enlazada en ambos lados.

f. Nodos

Fue creada una clase nodo con el propósito de pertenecer a una lista (lista de nodos) conteniendo cada uno de estos un espacio reservado de tipo objeto donde almacenara cada una de las posiciones y otro campo denominado apuntador que es necesario para apuntar al siguiente nodo y sea de utilidad para estructurar de una manera correcta la lista anteriormente explicada.

g. Apuntadores

Un apuntador es una variable que contiene la dirección de memoria de otra variable, es decir, hace referencia a la localidad de memoria de otra variable. Debido a que los apuntadores trabajan directamente con la memoria, a través de ellos se accede con rapidez a la información almacenada.

h. Liberia Minidom

xml.dom.minidom es una implementación mínima de la interfaz Document Object Model (Modelo de objetos del documento), con una API similar a la de otros lenguajes. Está destinada a ser más simple que una implementación completa del DOM y también significativamente más pequeña. Aquellos usuarios que aún no dominen el DOM deberían considerar usar el módulo xml.etree.ElementTree en su lugar para su procesamiento XML.

i. Graphviz

Graphviz es un software de visualización de gráficos de código abierto. La visualización gráfica es una forma de representar información estructural como gráficos abstractos y diagramas de red. Tiene importantes aplicaciones en la interfaz visual de redes, bioinformática, ingeniería de software, diseño de bases de datos y web, aprendizaje automático y otros campos técnicos.

Para la ejecución de esta se debe implementar la extensión .dot que fue utilizada en este proyecto para la realización de la gráfica de la secuencia de cola que debe ser el ensamblaje de cierto producto es decir, el proceso que debe llevar en el orden indicado esto implementado de cierta manera que sea fácil entender en que orden debió ensamblarse las piezas y así poder ser verificado en algún archivo de salida.

Para la realización de la gráfica fue un grafo en formato .dot que se utilizó graph para hacer la cola de secuencia que debe seguir cierto producto en específico. Cabe mencionar que para ejecutar este reporte no se puede hacer de manera masiva, debe de ejecutarse de manera individual para cada uno de los productos que se desea ensamblar. A continuación, un ejemplo del modelo de graphviz que se utilizó para este proyecto:

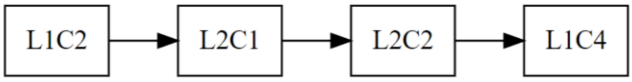

Figura 3. Ilustración de cola de secuencia del ensamblaje de cada producto.
Fuente: Enunciado proyecto 2 IPC2 USAC

Como se puede observar de esta manera se graficará la cola de secuencia del ensamblaje para cada producto, notándose también de un color diferente la parte de la cola que ya fue ensamblada y debido a esta implementación mediante la interfaz de usuario se tendrá que indicar en que tiempo se desea ver la representación gráfica del estado en ese momento de la cola de secuencia de ensamblaje.

j. Interfaz Grafica con Tkinter Python

La mayoría de las aplicaciones de escritorio de computadora más populares en la actualidad son la interfaz gráfica de usuario (GUI), es decir, a través del mouse para activar instrucciones sobre elementos gráficos como menús y botones, y obtenerlas de contenedores de visualización gráfica como etiquetas y cuadros de diálogo. información de diálogo de la máquina.

Python viene con el módulo tkinter, que es esencialmente la interfaz de programación Python de un popular kit de herramientas de GUI orientado a objetos, TK, que proporciona una forma rápida y conveniente de crear aplicaciones de GUI.

Tkinter es un módulo que usa Python para diseñar ventanas. El módulo Tkinter ("interfaz Tk") es una interfaz para el kit de herramientas GUI Tk estándar de Python. Como interfaz GUI específica para Python, es una ventana de imagen. Tkinter es una interfaz GUI que viene con Python y se puede editar. Podemos usar GUI para lograr muchas funciones intuitivas. Por ejemplo, si queremos desarrollar una calculadora, si es solo una entrada y salida de programa Con Windows, no hay experiencia de usuario. Es necesario todo el desarrollo de una pequeña ventana gráfica.

Conclusiones

- La implementación de listas enlazadas es muy importante ya que el manejo de datos lo realiza de manera dinámica. Sin embargo, no es la más optima debido a que el apuntador también ocupa un espacio en memoria.
- El manejo de archivos XML es una forma muy eficiente de estructurar datos de una manera clara y precisa sin necesidad de pasar horas en estructuras complejas y la lectura a través de Python es bastante sencilla.
- Graphviz es una herramienta muy útil si lo que queremos es representar una idea y plasmarla en una gráfica fácil de entender.

Referencias bibliográficas

- Jones, C. A., & Drake Jr, F. L. (2001). Python & XML: XML Processing with Python. " O'Reilly Media, Inc.".
- Joyanes Aguilar, L. (2003). Fundamentos de programación: algoritmos y estructura de datos y objetos.
- Ojeda, L. R. Tda Programacion Orientado a Objetos en Turbo. Univ. Nacional de Colombia.
- Roldán Blay, C. (2021). Crear e interrumpir bucles en Python.
- Roldán Blay, C. (2021). Definición y operaciones con listas en Python.

Anexos

Durante el desarrollo del proyecto fue necesario realizar distintas clases las cuales se realizo un diagrama de clases: