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.

PROYECTO 2 IPC 2

202000549 – Luis Daniel Salán Letona

Resumen

Programa realizado para fusionar elementos químicos de la tabla periódica y con ello generar compuestos nuevos. El programa cuenta con una base de elementos químicos, máquinas para la realización de cada compuesto nuevo, y los compuestos previamente cargados en el programa.

El programa cuenta con n cantidad de máquinas que contienen un cierto número de pines y número de elementos para poder trabajarlos. Las restricciones del programa son: No se pueden repetir los elementos en una misma máquina. Se podrá generar un archivo tipo xml con los tiempos en que se realizó cada compuesto nuevo, así como las instrucciones para que la máquina pueda fusionar y alinear cada elemento utilizado.

El programa fue realizado con el lenguaje Python y librerías como Tkinter como herramienta para realizar el entorno gráfico del mismo.

Palabras clave

Máquinas, Elementos, Listas enlazadas, POO, Compuestos químicos.

Abstract

A program was carried out to fuse chemical elements from the periodic table and create new compounds. The program includes a base of chemical elements, machines for creating each new compound, and previously loaded compounds.

The program includes a number of machines that each have a specific number of pins and elements that can be worked on. The program's limitations are as follows: The elements are not repeatable on the same machine. It will be possible to generate an xml file containing the times when each new compound was created, as well as the instructions that will allow the machine to merge and align each element used.

Keywords

. Machines, Elements, Linked lists, OOP, Chemical compounds.

Introducción

El programa puede iniciar su funcionamiento con un archivo tipo “xml” previamente cargado, de lo contrario los botones que contiene el menú principal o pantalla inicial que cumplen una función quedarán inactivos o bloqueados, y los únicos que quedan habilitados son 3: Cargar archivo, Salir y Ayuda.

El usuario podrá ver varias listas en las diferentes ventanas de acuerdo con lo que indique, como, por ejemplo, la lista de elementos cargados, lista de maquinas que contiene el archivo, compuestos si los tuviera, etc.

Asi mismo podrá visualizar gráficamente la información de las máquinas y la lista de instrucciones que utilizó la máquina para poder fusionar los elementos y generar nuevos compuestos.

Desarrollo del tema

Archivos xml:

XML consiste en un lenguaje de marcado creado por el W3C (World Wide Web Consortium), con la finalidad de definir una sintaxis para la codificación de documentos, que tanto los usuarios como las propias máquinas en sí puedan ser capaces de leer.

Para ello, lo hace mediante la utilización de una serie de etiquetas que definen la estructura que posee el documento en cuestión, además de cómo debe ser transportado y almacenado.

Para poder leer y recorrer la información del archivo de entrada se utilizó la librería elementTree, con la cual se parseo todo el archivo por tags o secciones según las etiquetas que contenía y asi mismo cada una fue agregada a una lista con sus atributos. Para la realización de las listas se utilizaron de tipo TDA.

La lista enlazada es un TDA que nos permite almacenar datos de una forma organizada, al igual que los vectores, pero, a diferencia de estos, esta estructura es dinámica, por lo que no tenemos que saber "a priori" los elementos que puede contener.

En una lista enlazada, cada elemento apunta al siguiente excepto el último que no tiene sucesor y el valor del enlace es NULL. Por ello los elementos son registros que contienen el dato a almacenar y un enlace al siguiente elemento. Los elementos de una lista suelen recibir también el nombre de nodos de la lista.

Para que esta estructura sea un TDA lista enlazada, debe tener unos operadores asociados que permitan la manipulación de los datos que contiene. Los operadores básicos de una lista enlazada son:

- Insertar: inserta un nodo con dato x en la lista, pudiendo realizarse esta inserción al principio o final de la lista o bien en orden.
- Eliminar: elimina un nodo de la lista, puede ser según la posición o por el dato.
- Buscar: busca un elemento en la lista.
- Localizar: obtiene la posición del nodo en la lista.
- Vaciar: borra todos los elementos de la lista

ESTRUCTURA DINÁMICA LISTA

Una lista está formada por una serie de elementos llamados nodos los cuales son objetos que contiene como variable miembro un puntero asignado y variables de cualquier tipo para manejar datos. El puntero sirve para enlazar cada nodo con el resto de nodos que conforman la lista.

De esto podemos deducir que una lista (lista) es una secuencia de nodos en el que cada nodo esta enlazado o conectado con el siguiente (por medio del puntero mencionado anteriormente). El primer nodo de la lista se denomina cabeza de la lista y el último nodo cola de la lista. Este último nodo suele tener su puntero igualado a NULL Para indicar que es el fin de la lista.

En las listas de acuerdo a quién apunte su cabeza y cola y al tipo de nodo que implementa (cuántos punteros tiene) se dividen en cuatro grandes tipos:

- Listas enlazadas (simplemente enlazadas)
- Listas doblemente enlazadas
- Listas circulares (simplemente circulares)
- Listas doblemente circulares

Las listas simplemente enlazadas, permiten recorrer la lista en un solo sentido y desde la cabeza hasta la cola.

Las listas doblemente enlazadas, permiten el recorrido en dos direcciones, de la cabeza a la cola y de la cola hacia la cabeza.

Las listas simplemente circulares, permiten el recorrido en una dirección, pero al llegar al último nodo (cola) este se encuentra comunicado o enlazado a la cabeza, haciendo un anillo o circulo si se representa gráficamente.

Las listas doblemente circulares, permiten el recorrido en ambas direcciones y la cabeza y cola se encuentran conectadas en ambas direcciones.

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.

Podemos entender la programación Orientada a objetos (POO) como una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación, que permite diseñar mejor las aplicaciones, llegando a mayores cotas de complejidad, sin que el código se vuelva inmanejable.

Al programar orientado a objetos tenemos que aprender a pensar cómo resolver los problemas de una manera distinta a como se realizaba anteriormente, en la programación estructurada. Ahora tendremos que escribir nuestros programas en términos de clases, objetos, propiedades, métodos y otras cosas que veremos rápidamente para aclarar conceptos y dar una pequeña base que permita soltarnos un poco con los conceptos de este tipo de programación.

Propiedades en clases

Las propiedades o atributos son las características de los objetos o ejemplares de una clase. Cuando definimos una propiedad normalmente especificamos su nombre y su tipo.

Aunque no es una manera muy correcta de hablar, nos podemos hacer a la idea de que las propiedades son algo así como variables donde almacenaremos los datos relacionados con los objeto

.

Métodos en las clases

Son las funcionalidades asociadas a los objetos, que son implantadas o programadas dentro de las clases.

Es decir, cuando estamos programando las clases, las funciones que creamos dentro asociadas a esas clases las llamamos métodos.

Aunque los métodos son como funciones, es importante que los llames métodos para que quede claro que son funciones que existen dentro del contexto de una clase, funciones que podremos invocar sobre todos los objetos creados a partir de una clase.

Todas las figuras deben ir enumeradas al pie de la imagen, como se muestra en el ejemplo.

Conclusiones

El uso de listas de tipo TDA, son creadas desde cero, tienen una mayor funcionalidad y eficacia que una lista nativa o por default del lenguaje utilizado, este caso, Python.

Para manejar mucha cantidad de datos es mucho mejor y viable usar este tipo de listas y agregar la misma mediante un método utilizando nodos.

Con la herramienta graphviz es mucho más visible e intuitivo para el usuario ver la información que contiene el archivo de entrada, en este caso para ver las máquinas y las instrucciones de la misma para trabajar.

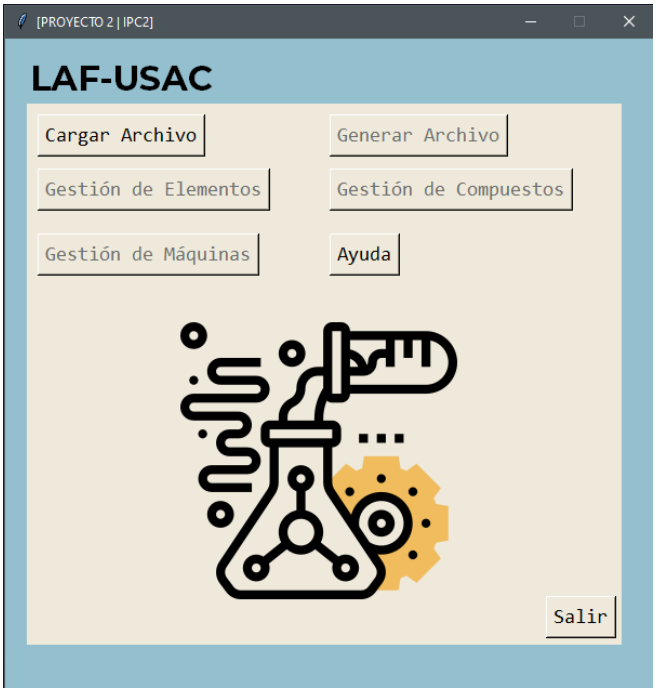

Figura 4. Ventana inicial sin archivo cargado.

Fuente: Elaboración propia, VS Code.

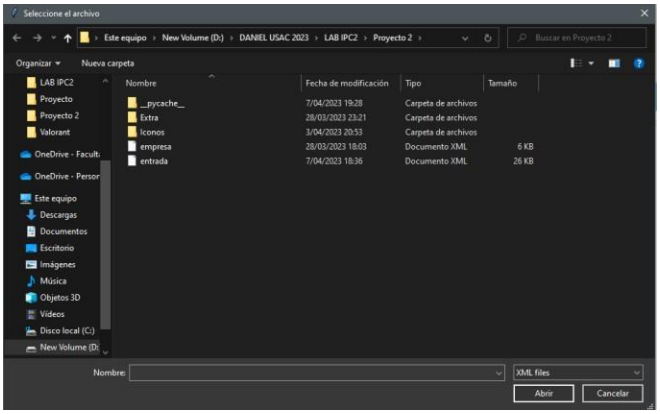

Figura 5. Ventana para seleccionar archivo de carga.

Fuente: Elaboración propia, VS Code.

Referencias bibliográficas

MÉNDEZ, Mariano, 75.41. *Algoritmos y Programación II TDA Lista y sus derivados.*

OJEDA, Luis Roberto. *TDA Programación Orientada a Objetos en Turbo. Universidad Nacional de Colombia.*

Anexos

```
def __init__(self):

    self.Ventana = tk.Tk()
    self.Ventana.title("[PROYECTO 2 | IPC2]")
    self.Ventana.geometry("600x600")
    self.Ventana.configure(bg="#93BFCF")
    self.Ventana.resizable(0,0)
    self.gadget()

    #------------Etiquetas-------------------
    titulo = tk.Label(self.Ventana, text="LAF-USAC",bg="#93BFCF", font=("Montserrat", 24)).place(x=20, y=10)
    img = tk.PhotoImage(file=".\Iconos\compuestos.png")
    logo = tk.Label(self.Ventana,bg="#EEE9DA", image=img).place(x=160, y=260)
```

Figura 3. Estructura Ventana inicial.

Fuente: Elaboración propia, VS Code.

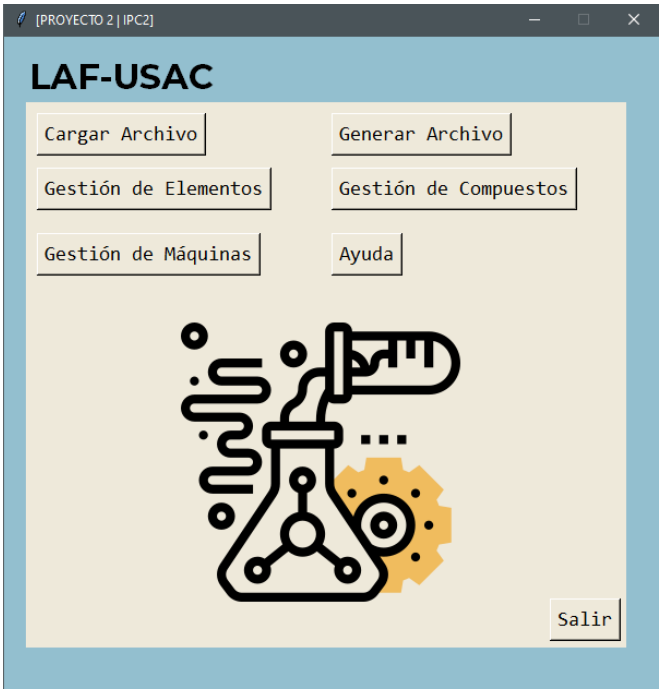

Figura 6. Ventana con botones activos.

Fuente: Elaboración propia, VS Code.

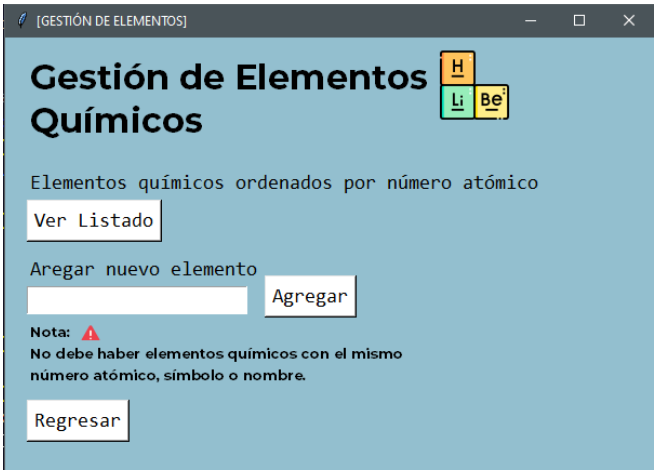

Figura 7. Ventana Gestión Elementos.

Fuente: Elaboración propia, VS Code.

Número atómico	Simbolo	Nombre
118	Og	Oganesón
117	Ts	Teneso
116	Lv	Livermorio
115	Mc	Moscovio
114	Fl	Flerovio
113	Nh	Nihonio
112	Cn	Copernicio
111	Rg	Roentgenio
110	Ds	Darmstatio
109	Mt	Meitnerio
108	Hs	Hassio
107	Bh	Bohrio
106	Sg	Seaborgio
105	Db	Dubnio
104	Rf	Rutherfordio
103	Lr	Laurencio
102	No	Nobelio
101	Md	Mendelevio

Figura 8. Lista de elementos cargados.

Fuente: Elaboración propia, VS Code.

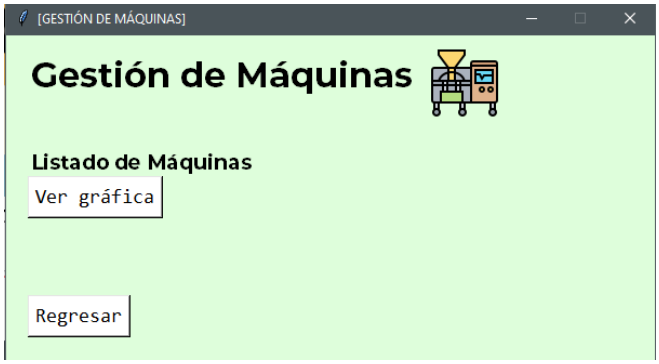

Figura 9. Ventana Gestión de Máquinas.

Fuente: Elaboración propia, VS Code.

Figura 10. Grafo de las máquinas cargadas.

Fuente: Elaboración propia, VS Code.

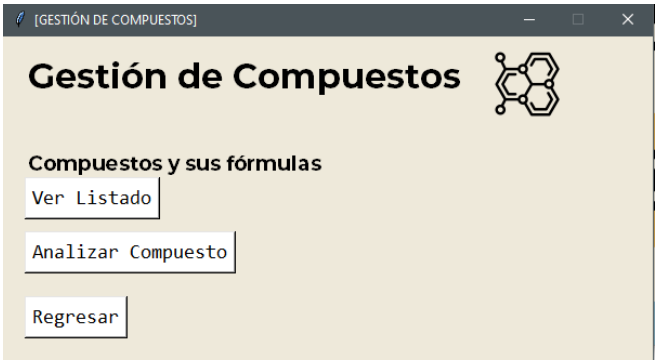

Figura 11. Ventana Gestión de Compuestos.

Fuente: Elaboración propia, VS Code.

Figura 12. Lista de Compuestos y sus fórmulas.

Fuente: Elaboración propia, VS Code.

Figura 13. Ventana Analizar compuesto.

Fuente: Elaboración propia, VS Code.

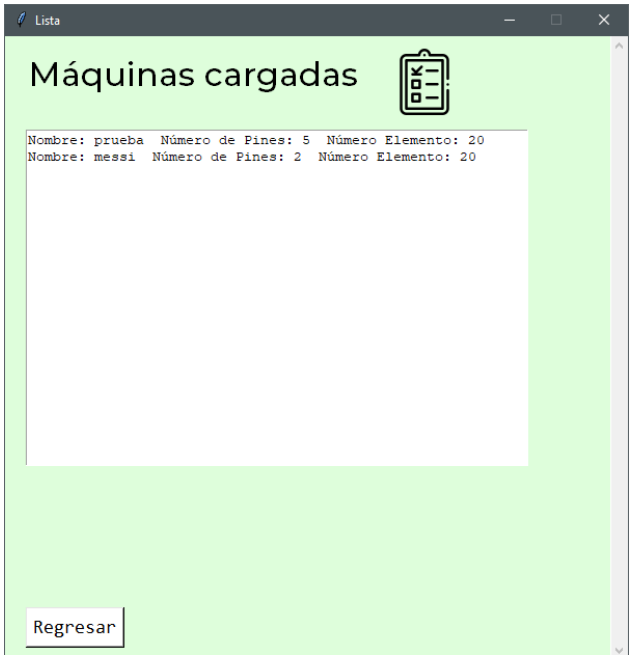

Figura 14. Lista de Máquinas cargadas.

Fuente: Elaboración propia, VS Code.

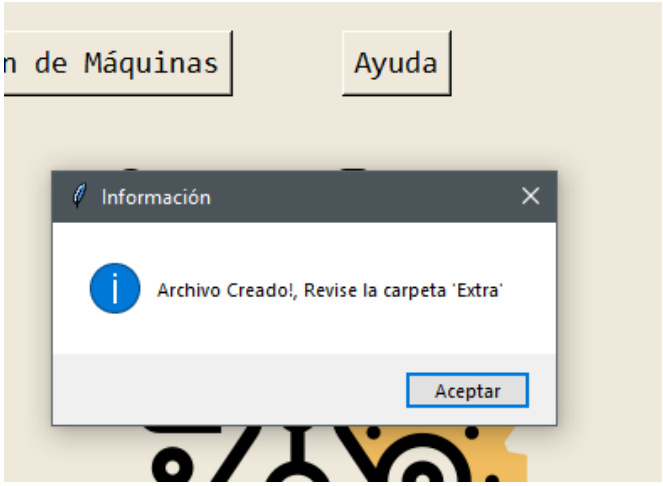

Figura 15. Archivo con link con información del creador.

Fuente: Elaboración propia, VS Code.

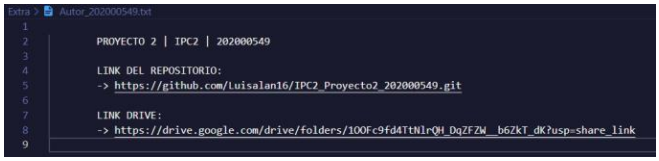

Figura 16. Archivo creado.

Fuente: Elaboración propia, VS Code