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 2023.

PROYECTO #1

201504499 – Elvin Leonel Mayen Carrillo

Resumen

Implementar conocimientos fundamentales en el desarrollo de aplicaciones y proveer una solución codificada en el lenguaje de programación Python, para lograrlo, se hace uso de diferentes técnicas y herramientas, como la manipulación eficiente de datos con estructuras de programación secuenciales, cíclicas y condicionales, con el propósito de entregar resultados completos, concisos y claros al usuario.

Además, se utiliza memoria dinámica mediante la implementación de datos de tipo abstracto (TDA's), lo que permite mantener un óptimo funcionamiento de las aplicaciones y una manipulación más eficiente de los datos. También se utiliza XML como estándar de archivo para la manipulación de datos, lo que permite una fácil interoperabilidad con otras aplicaciones. Finalmente, se brinda una visualización de los resultados mediante la herramienta Graphviz, que permite generar reportes en diferentes formatos y facilita la comprensión de los resultados por parte del usuario. Todo lo anterior es posible gracias al uso de librerías específicas para el lenguaje de programación Python.

Palabras clave

Memoria Dinámica, Listas Enlazadas, Abstracción, Método, TDA.

Abstract

Implement fundamental knowledge for the development of applications, provide a coded solution in the Python programming language to achieve this, different techniques and tools are used, such as efficient data manipulation with sequential, cyclic and conditional programming structures, with the purpose of delivering complete, concise and clear results to the user. In addition, dynamic memory is used through the implementation of abstract data types (ADT's), which allows maintaining an optimal performance of the applications and a more efficient data manipulation. XML is also used as a file standard for data manipulation, allowing easy interoperability with other applications. Finally, a visualization of the results is provided through the Graphviz tool, which allows the generation of reports in different formats and facilitates the understanding of results by the user. All of the above is possible thanks to the use of specific libraries for the Python programming language.

Keywords

Dynamic Memory, Linked Lists, Abstraction, Method, ADT.

Introducción

El lenguaje de programación Python se ha vuelto cada vez más popular debido a su facilidad de uso, su gran cantidad de librerías y herramientas disponibles, así como por su versatilidad y eficiencia en la manipulación de datos.

En este proyecto se busca implementar diferentes conceptos y técnicas fundamentales para el desarrollo de aplicaciones. Entre ellas, se incluye la manipulación eficiente de datos con estructuras de programación secuenciales, cíclicas y condicionales, la utilización de memoria dinámica mediante la implementación de datos de tipo abstracto, el uso de XML como estándar de archivo para la manipulación de datos, así como la generación de reportes visuales mediante la herramienta Graphviz.

A través de este proyecto, se espera proporcionar una idea clara de cómo aplicar estos conceptos y técnicas en el desarrollo de aplicaciones en Python, con el objetivo de mejorar las habilidades y conocimientos de programación.

Desarrollo del tema

a. Abstracción:

Para lograr una solución eficiente en el proyecto, se utilizó el concepto de abstracción, el cual permitió identificar las entidades fundamentales del sistema a partir de la lectura del archivo de ejemplo. En esta sección se explicará el proceso seguido para llevar a cabo esta tarea. Para ello, identifiqué las entidades principales que se requerían para representar la información del archivo: 'Organismo', 'Muestra' y 'Celda Viva'. Cada una de estas entidades posee atributos y métodos específicos que permiten modelar su comportamiento.

En la clase 'Organismo', definí los atributos correspondientes que la identifican, código, nombre y color. Además, implementé un método para asignar un color especifico que la representaría gráficamente en las muestras.

En la clase 'Muestra', incluí atributos como el identificador, descripción, cantidad de filas, cantidad de columnas y un atributo especial el cual hace referencia a una lista de ‘Celda Viva’.

Finalmente, en la clase 'Celda Viva', definí atributos como la posición en la muestra (fila y columna) y el código del ‘organismo’ en esa posición.

Además de las propiedades específicas de cada entidad, se añadió un atributo especial denominado 'siguiente' que permitiría a todas estas clases formar parte de una lista enlazada (TDA). Esta estructura de datos permite un fácil acceso y recorrido de las instancias de cada entidad, lo que sería útil en las futuras implementaciones del proyecto.

b. Implementación de TDA’s:

Memoria Dinámica: La memoria dinámica es una técnica de gestión de memoria utilizada en programación que permite asignar y liberar memoria en tiempo de ejecución. A diferencia de la memoria estática, que se asigna antes de la ejecución del programa y se mantiene constante

durante toda la ejecución, la memoria dinámica permite una asignación de memoria flexible según las necesidades del programa. Esto es importante para la programación ya que permite una gestión más eficiente y optimizada de la memoria, evitando desperdiciar recursos y permitiendo una mayor flexibilidad en la implementación de estructuras de datos y algoritmos.

Las listas enlazadas permitieron implementar las entidades 'Organismo', 'Muestra' y 'Celda Viva' debido a que estas entidades se pueden modelar como nodos, donde cada nodo tiene un conjunto de atributos que lo caracterizan y una referencia al siguiente nodo.

Cada entidad representa un nodo de una lista propia para cada una: 'Lista Organismo', 'Lista Celda Viva' y 'Lista Muestra', en la última cada nodo a su vez tiene un atributo 'lista_celdas_vivas' que se representa como una lista enlazada de nodos 'Celda Viva'.

Se creó una clase para cada lista, que incluye un atributo que hace referencia al primer nodo de la lista y un conjunto de métodos que permiten agregar, modificar y buscar elementos en la lista.

c. Persistencia de datos:

El patrón Singleton es un patrón de diseño de software que garantiza que una clase solo tenga una instancia y que provee un punto de acceso global a esa instancia. Esto significa que la instancia única de una clase Singleton es compartida por todos los objetos que la necesiten, lo que asegura que el estado de la clase se mantenga consistente en todo momento.

Para implementar la persistencia de los datos en las listas enlazadas, utilicé el patrón Singleton para tener una sola instancia para cada lista. Esto significa que en lugar de crear una nueva instancia cada vez que se lea un archivo de datos, se utiliza la misma instancia existente de la lista enlazada. De esta manera, se garantiza que los datos leídos de diferentes archivos se almacenen en la misma lista enlazada correspondiente a la entidad específica (Organismo, Muestra o Celda Viva).

Para aplicar el patrón Singleton, se creó una clase Singleton para las listas de 'Muestra' y 'Organismo' que tenía una instancia protegida y un constructor para inicializar cada lista en la instancia del Singleton. También se creó un método estático en la clase Singleton que devolvía la única instancia de la clase Singleton correspondiente a la lista específica.

d. Lectura de archivos:

Para la lectura del archivo XML, se utilizó la librería ElementTree de Python. Esta librería permite la manipulación de datos en formato XML, facilitando la lectura y escritura de archivos en este formato. En el caso específico de este proyecto, se implementó un método para la lectura del archivo XML. Este método recibe como parámetro el nombre del archivo y utiliza la función 'parse' de ElementTree para abrir el archivo y obtener su contenido en formato de árbol.

Posteriormente, se recorrió el árbol obteniendo los elementos de cada entidad utilizando funciones cíclicas y pasando los datos correspondientes al método agregar de cada lista enlazada correspondiente el cual se encarga propiamente de crear sus nodos.

Es importante destacar que la utilización de la librería ElementTree permitió una lectura eficiente y efectiva de los archivos XML, evitando la necesidad de implementar complejos algoritmos de lectura y procesamiento de este tipo de archivos.

e. Reportes visuales:

Para desarrollar graficas para presentar como reportes visuales se utilizó la herramienta Graphviz, la cual se utilizó para generar la gráfica correspondiente, se generó un archivo DOT en Python. Los archivos DOT es un archivo de texto que contiene las instrucciones necesarias para que Graphviz pueda generar la gráfica correspondiente.

El proceso para generar las gráficas a partir de las listas enlazadas del proyecto se llevó a cabo en varias etapas. En primer lugar, se recibió el código de la muestra que el usuario deseaba graficar. A continuación, se procedió a iterar las listas de celdas vivas de la muestra dada utilizando estructuras de programación cíclicas y condicionales.

Con la información necesaria, se generó un tablero de un tamaño dado por la muestra. En este tablero se representaron las casillas en donde se encuentran los organismos, siendo cada tipo de organismo identificado por su color único asignado previamente.

f. Manipulación de datos:

Utilizando diversas estructuras de programación secuenciales, cíclicas y condicionales se desarrollaron diversos métodos que en conjunto permiten determinar las posiciones indicadas para agregar nuevos organismos a las diferentes muestras, cambiando de tal manera las muestras necesarias formando una estructura nueva.

Con nuestras muestras modificadas se decidió implementar la posibilidad de escribir nuevos archivos XML con los cambios realizados para su posterior procesamiento.

Conclusiones

El proyecto permitió la aplicación de diversos conceptos de programación en Python, así como la implementación de diferentes herramientas para la manipulación y visualización de datos. A través de la utilización de listas enlazadas, se logró almacenar y manipular información de manera eficiente, permitiendo la representación y evolución de diferentes muestras de organismos.

La utilización de estructuras de programación secuenciales, cíclicas y condicionales fue fundamental para la implementación de las diferentes funcionalidades del proyecto, permitiendo el manejo adecuado de la información y la toma de decisiones en función de las condiciones particulares de cada escenario.

La herramienta Graphviz resultó de gran utilidad para la visualización de los datos almacenados en las listas enlazadas, permitiendo la generación de gráficas claras y representativas de la evolución de los organismos en cada muestra. Asimismo, la

utilización de XML para la manipulación de los datos permitió la organización y almacenamiento adecuado de la información.

Finalmente, la implementación de conceptos de memoria dinámica a través de la creación de TDAs permitió la adecuada manipulación de la información en las listas enlazadas, permitiendo la adición y eliminación de elementos de manera eficiente y segura.

Referencias bibliográficas

Brandon Rhodes. *The Singleton Pattern*. https://python-patterns.guide/gang-of-four/singleton/.

Graphviz Documentation. https://graphviz.org/doc/info/lang.html.

Element Tree documentation, https://docs.python.org/3/library/xml.etree.elementtree.html.

Anexos

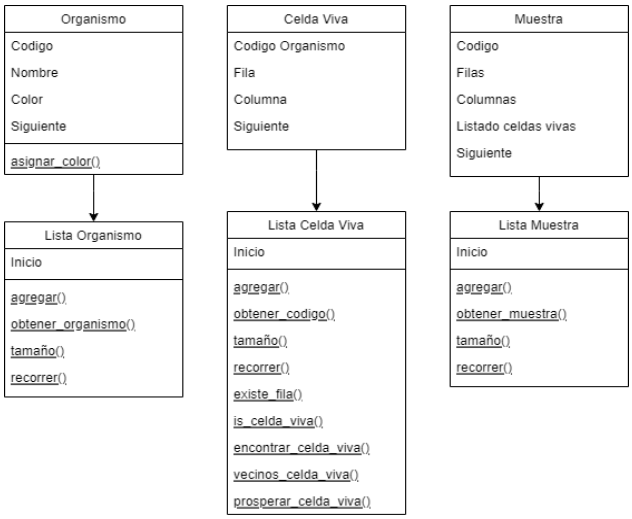

Figura 1. Diagrama de Clases TDA's. Fuente: elaboración propia.