Proyecto 1

202007092 – Jonatan Josue Vasquez Pastor

Resumen

Se presentará a continuación una probable solución sobre la resolución para el proyecto del laboratorio de IPC2 utilizando los temas vistos en la clase y poniendo a prueba los conocimientos obtenidos en clase, usando diferentes métodos como listas doblemente enlazadas y herramientas como graphviz y librerías para leer xml's, también se subió a la plataforma github para la calificación del proyecto.

Palabras clave

Estructura de datos, XML, graphviz, listas enlazadas

Abstract

Next, a probable solution on the resolution for the IPC2 laboratory project will be presented using the topics seen in class and testing the knowledge obtained in class, using different methods such as doubly linked lists and tools such as graphviz and libraries to read xml's , it was also uploaded to the github platform for the qualification of the project.

Keywords

Data structure, XML, graphviz, linked lists

Introducción

La estructura de datos en XML es flexible y extensible, lo que permite a los desarrolladores definir sus propias etiquetas y atributos según sus necesidades específicas. Además, XML es compatible con muchos lenguajes de programación y plataformas, lo que lo hace ampliamente utilizado y versátil en diferentes entornos de programación.

En resumen, XML es un lenguaje de marcado que se utiliza para estructurar y almacenar datos en un formato legible por máquina y humano. La estructura de datos en XML se basa en etiquetas y atributos, lo que lo hace flexible y extensible para diferentes aplicaciones y entornos de programación.

XML

XML significa "Lenguaje de Marcado Extensible" (en inglés, "Extensible Markup Language"). Es un lenguaje de programación utilizado para almacenar y transportar datos estructurados en forma de texto.

El XML utiliza etiquetas para definir los elementos de un documento y para indicar cómo se relacionan entre sí. Por ejemplo, se podría utilizar XML para definir un documento que contenga información sobre libros, incluyendo título, autor, ISBN, fecha de publicación, etc. Cada elemento del documento tendría una etiqueta que lo identifica y describe su contenido.

Estructura de datos

forma de organizar y almacenar datos de manera eficiente, para que puedan ser accedidos y manipulados de manera efectiva. Las estructuras de datos son fundamentales en la programación, ya que permiten que los programas puedan trabajar con grandes cantidades de datos de manera eficiente y eficaz.

Cada estructura de datos tiene sus propias ventajas y desventajas, y se utiliza en diferentes situaciones dependiendo de las necesidades del programa. La elección de la estructura de datos adecuada es crucial para el rendimiento y la eficiencia del programa.

Ideas principales para resolver el problema del proyecto 1

Leer el documento xml para obtener los datos dentro de los tags, para ser guardados en una lista doblemente enlazada, se ordenan los datos según se vayan dando, muestra, organismo, etc.

Se ingresa un organismo vivo y se realiza la operación para verificar si el organismo le dará vida a otros organismos o no.

Conclusiones

En conclusión, XML es un lenguaje de marcado utilizado para estructurar y almacenar datos en un formato legible por máquina y humano. La estructuración de datos en XML se basa en etiquetas que definen elementos y atributos que describen los valores de los elementos.

Referencias bibliográficas

Apéndices


```
from os import remove
from clase import *
from le import *




def crearTabla(ruta, columas, filas, celdasVivas):
    """Crea una tabla usando Graphviz"""
    contenido = open(ruta, 'w')

    """serivira para poder darle inicio a nuestra tabla en graphviz"""
    contenido.write('digraph L{ a0[shape=none label=<<TABLE border="1"
    cellspacing="0" cellpadding="15">\n')


    for i in range(filas):
        contenido.write('<TR>')
        for s in range(columas):
            contenido.write('<TD></TD>')
        contenido.write("</TR>")


    """servira para ponerle fin a nuestra tabla en graphviz"""
    contenido.write('\n</TABLE>>];}')
    contenido.close()

def eliminarArchivo(ruta):
    remove(ruta)

root = 'prueba2.dot'
colum = 10
row = 10

def agregarOrganismo(columna, fila):
    print('se agrego un aroganismo a la posicion x: ', columna, ' Y: ',
    fila)


celdaV = CeldaViva(2,3,'5454', 'red')
celdV2 = CeldaViva(4,2,'3456','yellow')
celdV3 = CeldaViva(6,2,'5432','gray')
```

Figura 1. Algoritmo planteado
Fuente: elaboración propia.

```python
class Organismo:
    def __init__(self, codigo, nombre, color):
        self.codigo = codigo
        self.nombre = nombre
        self.color = color

    def mostrar(self):
        if not None:
            print(f'nombre: {self.nombre}')
            print(f'codigo: {self.codigo}')
            print(f'color: {self.color}')


class CeldaViva:
    def __init__(self, fila, columna, codigo, color):
        self.fila = fila
        self.columna = columna
        self.codigo = codigo
        self.color = color

    def mostrar(self):
        print(f'columna: {self.columna}')
        print(f'fila: {self.fila}')
        print(f'codigo: {self.codigo}')
        print(f'color: {self.color}')

    def color(self):
        return print(f'color: {self.color}')


class Muestra:
    def __init__(self, descripcion, codigo, filas, columnas):
        self.descripcion = descripcion
        self.codigo = codigo
        self.filas = filas
        self.columnas = columnas

    def mostrarInfo(self):
        print(f'Descripcion: {self.descripcion}')
        print(f'Codigo: {self.codigo}')
        print(f'Filas: {self.filas}')
        print(f'Columnas: {self.columnas}')
```

Figura II. Algoritmo planteado
Fuente: elaboración propia.

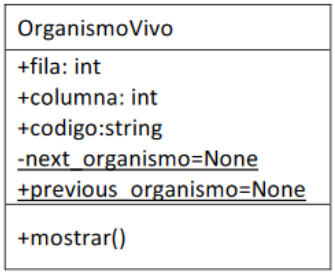

Figura III. Algoritmo planteado
Fuente: elaboración propia.

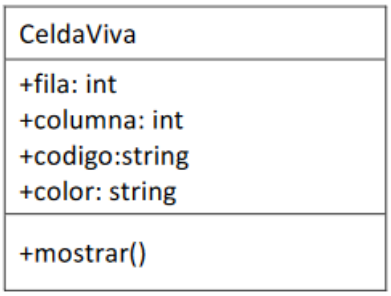

Figura IV. Algoritmo planteado
Fuente: elaboración propia

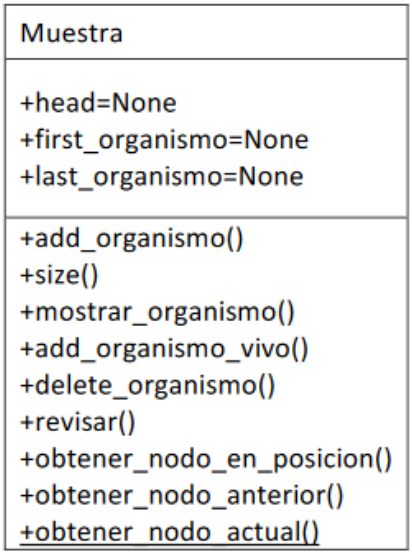

Figura V. Algoritmo planteado
Fuente: elaboración propia

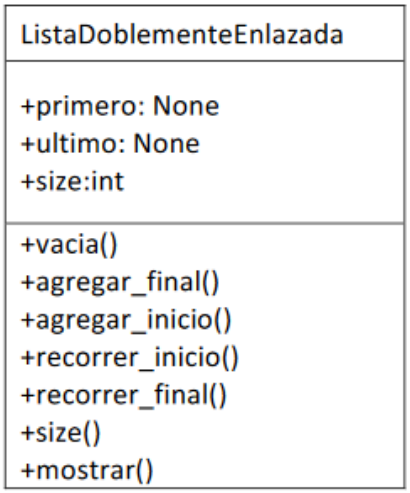

Figura VI. Algoritmo planteado
Fuente: elaboración propia

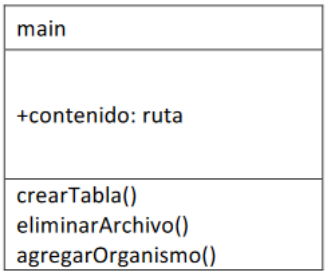

Figura V1I. Algoritmo planteado
Fuente: elaboración propia