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 NO.2 PRIMER SEMESTRE 2021

201700747 – Dennis Alexander Gamboa Stokes

Resumen

La Programación Orientada (POO) a objetos permite que el código sea reutilizable, organizado y fácil de mantener. Sigue el principio de desarrollo de software utilizado por muchos programadores DRY (Don't Repeat Yourself), para evitar duplicar el código y crear de esta manera programas eficientes mientras Graphviz es una herramienta utilizada por muchas personas que permite la creación de graficas usando un lenguaje especifico.

Se investigo el uso de la Programación orientada a Objetos para la elaboración de una matriz ortogonal y también diferentes métodos para la facilitación del programa como también se investigó el uso del graphviz para poder graficar la matrices que entraron al sistema por medio de una estructura XML que también se investigo acerca de y por último se investigó TDA para ser utilizado en la programación.

Palabras clave

Matriz ortogonal , Graphviz, XML Estructura de programación

Abstract

Object Oriented Programming (OOP) allows code to be reusable, organized, and easy to maintain. It follows the software development principle used by many programmers DRY (Don't Repeat Yourself), to avoid duplicating the code and thus create efficient programs while Graphviz is a tool used by many people that allows the creation of graphs using a language specific.

The use of Object-Oriented Programming was investigated for the elaboration of a Orthogonal linked list and also different methods for the facilitation of the program, as well as the use of Graphviz to be able to graph the matrices that entered the system through an XML structure. that was also investigated about and finally ADD was investigated to be used in programming.

Keywords

Orthogonal linked list, Graphviz, XML, Programming Structures

Introducción

En el Presente trabajo o programa se mostrará la información principal de los métodos utilizados en la Programación orientada a objetos (POO) como también la visualización de TDA mediante el uso de Graphviz. Se hará una carga de datos provenientes de una estructura XML y se usara POO para facilitar la interpretación de datos.
Se mostrara la estructura de una matriz ortogonal como también su métodos para la facilitación de control de datos y por ultimo las graficas generada de los datos del XML se mostrara mediante el uso de Graphviz que es una herramienta muy útil para la creación de graficas.

Desarrollo del tema

XML:
Extensible Markup Language (XML) se utiliza para describir datos. El estándar XML es una forma flexible de crear formatos de información y compartir electrónicamente datos estructurados a través de la Internet pública, así como a través de redes corporativas.
Usos:
Intercambio de datos entre sistemas, uno de los objetivos fundamentales de XML es permitir la posibilidad de intercambiar datos de forma estructurada entre diferentes sistemas. Al tratarse de un formato de texto plano y ser un lenguaje estandarizado, hace que esta transferencia sea muy ágil e independiente de la plataforma utilizada.

Base de datos, XML permite guardar datos de forma estandarizada para luego poder ser tratados por multitud de lenguajes diferentes. Su manejo es mucho más sencillo que bases de datos como MySQL y mucho más rico que utilizar ficheros de texto planos.

Conversor, actualmente son muchos los formatos que ofrecen servicios de conversión a XML, como PDF, HTML, .text, .docx o XHTML.

```
<? xml version = "1.0" standalone = "yes"?>

<conversacion>

<saludo> ¡Hola, mundo! </saludo>

<respuesta> Cuiden al planeta </respuesta>

</conversacion>.
```

*Figura 1.*Estructura de XML.

Fuente: elaboración propia

Graphviz:

Graphviz (abreviatura de Graph Visualization Software) es un software abierto de libre distribución para graficar, que presenta información estructural en forma de diagramas y puede aplicarse en diversas áreas como el análisis de redes, bioinformática, ingeniería de software, bases de datos, diseño de sitios web, aprendizaje por computadora y tiene interfaces gráficas para otros dominios.

Su modo de utilización se basa en el diseño de pequeños programas que toman descripciones de los diagramas de un lenguaje de texto simple y los dibuja en diversos formatos tales como archivos de imágenes, SVG, PDF ó para desplegarse en exploradores.

En la práctica, generalmente las gráficas se generan de fuentes de datos externas, (típicamente mediante el recorrido de la estructura de datos correspondiente) para crear un programa que pueda ser interpretado y ejecutado por alguno de los componentes de Graphviz y generar la imagen correspondiente.

```
digraph G {Hello->World}
```

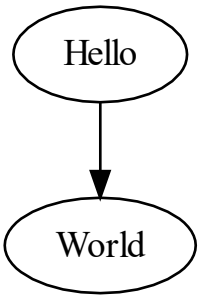

*Figura 2.*Estructura y diagrama de graphviz.

Fuente: elaboración propia

Matriz Ortogonal:

Las Matrices Ortogonales son estructuras de datos que contienen dos lista doblemente enlazadas además que contiene 4 apuntadores uno que apunta hacia la derecha otro que apunta hacia la izquierda uno que apunta hacia abajo y el ultimo hacia arriba lo cual la convierte en una lista sin fin, cada nodo siempre tiene uno anterior y uno siguiente, su estructura es muy similar a las listas doblemente enlazadas por lo cual comparten características tanto en su implementación como en su manejo aunque requiere un mayor entendimiento del manejo de los punteros.

Características:

No existe ningún nodo que apunte a null.
La lista no tiene fin ya que al llegar al último nodo apunta hacia el nodo anterior

Se accede a la lista mediante el primer nodo o también llamado inicio de la lista.

Si no se tiene cuidado al manejar se pueden crear bucles infinitos.

No tiene acceso aleatorio es decir para acceder a un valor se debe recorrer toda la lista.

Operaciones:

agregar (valor): agrega el valor al final de la lista.

insertar (referencia, valor): inserta el valor después del valor de referencia en la lista.

remover (referencia): elimina el nodo con el valor que coincida con la referencia.

editar (referencia): actualiza el valor de nodo con el valor que coincida con la referencia.

esVacia (): retorna true si la lista está vacía, false en caso contrario.

buscar (valor): retorna la true si el elemento existe en la lista, false caso contrario.

eliminar(): elimina la lista
listar (): imprime en pantalla los elementos de la lista.

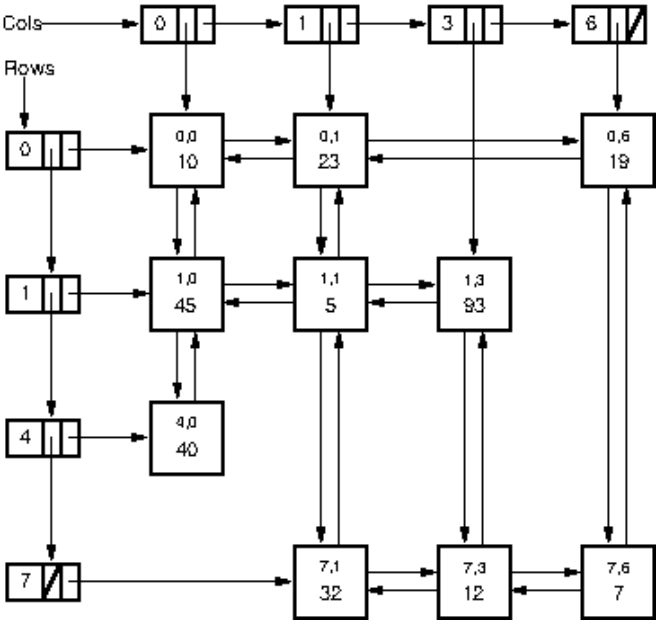

*Figura 3.*Estructura Matriz Ortogonal

Fuente: Google Image

Estructura de Programación:

Estructura Condicional:

Permite alterar la secuencia normal de pasos en un paso específico del Algoritmo, para crear 2 alternativas de bloques de ejecución, de manera excluyente entre ambos. En otras palabras: Solo uno de los 2 bloques se ejecutará, nunca ambos bloques y permite decidir por cuál alternativa seguirá el flujo del programa dependiendo del resultado de la evaluación de una condición. Para establecer condiciones complejas se utilizan los operadores relacionales y lógicos

Estructura cíclica:

Se llaman problemas repetitivos o cíclicos a aquellos en cuya solución es necesario utilizar un mismo conjunto de acciones que se puedan ejecutar una cantidad específica de veces. Esta cantidad puede ser

```python
class Nodo1():
    def __init__(self,fila,columna,valor,nombre):
        self.nombre=nombre
        self.valor=valor
        self.fila=fila
        self.columna=columna
        self.derecha=None
        self.izquierda=None
        self.abajo=None
        self.arriba=None
class nodoencabezado:
    def __init__(self,id):
        self.id=id
        self.siguiente=None
        self.anterior=None
        self.acceso=None

class listaencabezado:
    def __init__(self,primero=None):
        self.primero=primero
    def eliminarp(self):
        if(self.primero==None):
            print("vacio")

        if self.primero.siguiente==None:
                self.primero=None
                return

        self.primero=self.primero.siguiente
        self.primero.anterior=None
    def eliminar(self,id):
        if(self.primero==None):
            print("vacio")
        else:
            if(self.primero==id):
                print(self.primero.id)
                self.eliminarp()
```

```python
                if(actual.abajo==None):
                    actual.abajo=nuevo
                    nuevo.arriba=actual
        #insercion encabezado por filas
        efila= self.efilas.getencabezado(fila)
        if efila==None:
            efila=nodoencabezado(fila)
            efila.acceso=nuevo
            self.efilas.setencabezadp(efila)
        else:
            if (nuevo.columna < efila.acceso.columna):
                nuevo.derecha=efila.acceso
                efila.acceso.izquierda=nuevo
                efila.acceso=nuevo
            else:
                actual=efila.acceso
                while actual.derecha !=None:
                    if nuevo.columna< actual.derecha.columna:
                        nuevo.derecha=actual.derecha
                        actual.derecha.izquierda=nuevo
                        nuevo.izquierda=actual
                        actual.derecha=nuevo
                        break
                    actual=actual.derecha
                if(actual.derecha==None):
                    actual.derecha=nuevo
                    nuevo.izquierda=actual
    def buscar1(self,fila,columna):
        ecolumna=self.ecolumnas.primero
        while ecolumna !=None:
            actual=ecolumna.acceso
            while actual !=None:
                if actual.fila==fila and actual.columna==columna:
                    return actual.valor
                actual=actual.abajo
            ecolumna=ecolumna.siguiente
```

En esta imagen se puede apreciar la creación del nodo y la matriz ortogonal con algunos de sus métodos que se usó para aguardar los datos, buscar y eliminar

```python
    def setencabezadp(self,nuevo):
        if (self.primero==None):
            self.primero=nuevo
        elif(nuevo.id<self.primero.id):
            nuevo.siguiente=self.primero
            self.primero.anterior=nuevo
            self.primero=nuevo
        else:
            actual=self.primero
            while actual.siguiente !=None:
                if(nuevo.id<self.primero.id):
                    nuevo.siguiente=actual.siguiente
                    actual.siguiente.anterior=nuevo
                    nuevo.anterior=actual
                    actual.siguiente=nuevo
                    break
                actual=actual.siguiente
            if(actual.siguiente==None):
                actual.siguiente=nuevo
                nuevo.anterior=actual
    def getencabezado(self,id):
        actual=self.primero
        while actual !=None:
            if(actual.id==id):
                return actual
            actual=actual.siguiente
        return None
class matrizx:
    def __init__(self):
        self.efilas=listaencabezado()
        self.ecolumnas=listaencabezado()


    def insertar(self,fila,columna,valor,nombre):

        nuevo=Nodo1(fila,columna,valor,nombre)
        ecolumna=self.ecolumnas.getencabezado(columna)
```

```python
class matrizx:
    def __init__(self):
        self.efilas=listaencabezado()
        self.ecolumnas=listaencabezado()


    def insertar(self,fila,columna,valor,nombre):

        nuevo=Nodo1(fila,columna,valor,nombre)
        ecolumna=self.ecolumnas.getencabezado(columna)
        print(ecolumna)
        if ecolumna==None:
            ecolumna=nodoencabezado(columna)
            ecolumna.acceso=nuevo
            self.ecolumnas.setencabezadp(ecolumna)
        else:
            if (nuevo.fila < ecolumna.acceso.fila):
                nuevo.abajo=ecolumna.acceso
                ecolumna.acceso.arriba=nuevo
                ecolumna.acceso=nuevo
            else:
                actual=ecolumna.acceso
                while actual.abajo !=None:
                    if nuevo.fila< actual.abajo.fila:
                        nuevo.abajo=actual.abajo
                        actual.abajo.arriba=nuevo
                        nuevo.arriba=actual
                        actual.abajo=nuevo
                        break
                    actual=actual.abajo
                if(actual.abajo==None):
                    actual.abajo=nuevo
                    nuevo.arriba=actual
        #insercion encabezado por filas
        efila= self.efilas.getencabezado(fila)
        if efila==None:
            efila=nodoencabezado(fila)
```

*Figura 5.*Estructura Matriz Ortogonal Programación

Fuente: Propia

Figura 6. Uso de Xml minidom para leer Archivo

Fuente: Propia

```
lista=lista_circular()
def leerarchivo(cadena):
    global lista_matriz
    ver=False
    doc = xml.dom.minidom.parseString(cadena)

    nombre = doc.getElementsByTagName("matrices")

    matris = doc.getElementsByTagName("matriz")
    for ma in matris:
        if lista.vacia():

            nombre = ma.getElementsByTagName("nombre")
            name = Nodo(nombre[0].firstChild.nodeValue)
            lista.Agregarinicio(name)
            fila = ma.getElementsByTagName("filas")
            columna = ma.getElementsByTagName("columnas")
            im = ma.getElementsByTagName("imagen")
            print(nombre[0].firstChild.nodeValue,fila[0].firstChild.nodeValue,columna[0].firstChild.nodeValue)
            lista_matriz.append(matriz(nombre[0].firstChild.nodeValue,fila[0].firstChild.nodeValue,columna[0].firstChild.nodeValue,im[0

            ver=True
        else:
            nombre = ma.getElementsByTagName("nombre")
            name=Nodo(nombre[0].firstChild.nodeValue)
            if(lista.buscar(nombre[0].firstChild.nodeValue)==True):
                print("ya esta: "+nombre[0].firstChild.nodeValue)
            else:
                lista.Agregarinicio(name)
                fila = ma.getElementsByTagName("filas")
                columna = ma.getElementsByTagName("columnas")
                im = ma.getElementsByTagName("imagen")
                print(nombre[0].firstChild.nodeValue,fila[0].firstChild.nodeValue,columna[0].firstChild.nodeValue)
                lista_matriz.append(matriz(nombre[0].firstChild.nodeValue,fila[0].firstChild.nodeValue,columna[0].firstChild.nodeValue,
```

Se usa XML minidom para hacer la carga masiva con las validaciones pedidas y poder aguardar la información.

```
B = ttk.Button(raiz, text ="Seleccionar",command=verlo)
B2 = ttk.Button(raiz, text ="Seleccionar", command=verlo2)
B3 = ttk.Button(raiz, text ="Seleccionar",command=operar)
B4 = ttk.Button(raiz, text ="Seleccionar",command=rotacion)
B5 = ttk.Button(raiz, text ="Seleccionar",command=extra)
B.grid(column = 2, row = 15)
B2.grid(column = 2, row = 20)
B4.grid(column = 2, row = 30)
B3.grid(column = 2, row = 25)
B5.grid(column = 7, row = 40)
triangulo.grid(column=8,row=35)
horizontal.grid(column=8,row=20)
vertical.grid(column=8,row=25)
rectangulo.grid(column=8,row=30)
limpiar.grid(column=8,row=15)

raiz.mainloop()
```

Figura 7. Creación de la ventana

Fuente: Propia

```
def main():
    raiz=Tk()
    global original
    global segunda
    global resultado
    if(original !=""):

        raiz.title("Proyecto2")
        menubar=Menu(raiz)
        panel_1=PanedWindow(bd=4, relief='raised', bg='white')
        panel_1.pack(fill=BOTH, expand=1)


        ayuda=Menu(menubar,tearoff=0)
        menubar.add_command(label='Cargar Archivo', command=buscar)
        menubar.add_command(label='Operaciones',command=Operaciones)
        menubar.add_command(label='Reportes')

        #_____________________
        ayuda.add_command(label='Informacion Estudiante')
        ayuda.add_command(label='Documentacion del programa')
        ayuda.add_separator()
        menubar.add_cascade(label='Ayuda', menu=ayuda)
        raiz.config(menu=menubar)
        raiz.geometry("520x480")
    else:

        raiz.title("Proyecto2")
        menubar=Menu(raiz)



        ayuda=Menu(menubar,tearoff=0)
        menubar.add_command(label='Cargar Archivo', command=buscar)
        menubar.add_command(label='Operaciones',command=Operaciones)
        menubar.add_command(label='Reportes',command=html)
```

Figura 8. Creación de la ventana
Fuente Propia

```
    combo= ttk.Combobox(raiz,width = 27)
    combo2= ttk.Combobox(raiz,width = 27)
    limpiar = ttk.Entry(raiz)
    rectangulo = ttk.Entry(raiz)
    vertical = ttk.Entry(raiz)
    horizontal = ttk.Entry(raiz)
    triangulo = ttk.Entry(raiz)
    data = ("Unión", "Intersección", "Diferencia", "Diferenciasimétrica")
    data1=("horizontal","vertical","Transpuesta")
    cb = ttk.Combobox(raiz, values=data)
    cb2 = ttk.Combobox(raiz, values=data1)
    ttk.Label(raiz, text = "Selecione Matrix Original :",
    font = ("Times New Roman", 10)).grid(column = 0,
    row = 15, padx = 10, pady = 25)
    ttk.Label(raiz, text = "Limpiar :",
    font = ("Times New Roman", 10)).grid(column = 7,
    row = 15, padx = 10, pady = 25)
    ttk.Label(raiz, text = "Selecione Matrix a operar :",
    font = ("Times New Roman", 10)).grid(column = 0,
    row = 20, padx = 10, pady = 25)
    ttk.Label(raiz, text = "Linea Horizontal :",
    font = ("Times New Roman", 10)).grid(column = 7,
    row =20 , padx = 10, pady = 25)
    ttk.Label(raiz, text = "Tipo de Operacion :",
    font = ("Times New Roman", 10)).grid(column = 0,
    row = 25, padx = 10, pady = 25)
    ttk.Label(raiz, text = "Linea Vertical :",
    font = ("Times New Roman", 10)).grid(column = 7,
    row =25 , padx = 10, pady = 25)
    ttk.Label(raiz, text = "Rotacion :",
    font = ("Times New Roman", 10)).grid(column = 0,
    row = 30, padx = 10, pady = 25)
    ttk.Label(raiz, text = "Rectangulo :",
    font = ("Times New Roman", 10)).grid(column = 7,
```

Figura 9. Creación de la ventana
Fuente :Propia

```python
def matrizp(self,fila,columna):
    linea=''
    jump=''
    final=''
    jk=int(fila)
    jk=jk+1
    kj=int(columna)
    kj=kj+1
    for i in range(1,jk):
        for j in range(1, kj):
            val=self.buscar1(i,j)
            if val==None:
                pass
            else:
                linea=linea+val
        salto=linea+'\n'
        final=final+salto
        linea=''
    return(final)
```

Figura 10.Ingreso de datos a matriz ortogonal
Fuente :Propia

```python
    if(estado==0):
        if(char=='-'):
            li.insertar(ff,cc,char,valor)
            listaextra.append(char)
            cc=cc+1
            vacio=vacio+1
        elif(char=='*'):
            li.insertar(ff,cc,char,valor)
            listaextra.append(char)
            cc=cc+1
            lleno=lleno+1
        elif(char.isspace()):
            estado=1
    elif(estado==1):
        if(char=='-'):
            li.insertar(ff,cc,char,valor)
            listaextra.append(char)
            vacio=vacio+1
            cc=cc+1
        elif(char=='*'):
            li.insertar(ff,cc,char,valor)
            listaextra.append(char)
            lleno=lleno+1
            cc=cc+1
        elif(char.isspace()):
            estado=1
            if(char=='\n'):
                listaextra.append(char)
                ff=ff+1
                cc=1
                estado=0
reporte=reporte+", Espacios LLenos: "+str(lleno)+", Espacios Vacios: "+str(vacio)
listareporte.append(report("matriz",reporte))
mo1()
```

Figura 11. Lectura de datos en etiqueta Imagen
Fuente :Propia

Figura 12. Creación de Html
Fuente :Propia

```python
def html():
    global listareporte
    qu=""
    sk=""
    nombre=""
    estado=0
    error=[]
    matriz=[]
    operacion=[]
    f = open(r"C:\Users\denni\OneDrive\Desktop\reporte.html",'w')
    f.write("<html> <head> <style> </style></head> <body>"+'\n')
    for i in listareporte:
        ver=i.id
        nombre=i.descripcion
        if(ver=="operacion"):
            operacion.append(nombre)
        elif(ver=="matriz"):
            matriz.append(nombre)
        elif(ver=="error"):
            error.append(nombre)
    f.write("<h1 align="+qu+"center"+qu+">Matrices Trabajadas</h1>"+'\n')
    for j in matriz:
        nombre=j
        f.write("<h2>"+nombre+"</h2>"+'\n')
    f.write("<h1 align="+qu+"center"+qu+">Operaciones</h1>"+'\n')
    for k in operacion:
        nombre=k
        f.write("<h2>"+nombre+"</h2>"+'\n')
    f.write("<h1 align="+qu+"center"+qu+">Errores</h1>"+'\n')
    for l in error:
        nombre=l
        f.write("<h2>"+nombre+"</h2>"+'\n')

    f.write("</body> </html>")


    f.close()
    os.startfile(r"C:\Users\denni\OneDrive\Desktop\reporte.html")
```

```python
    global ima
    global valor
    global fll
    global coll
    kf=int(fll)
    kc=int(coll)
    kf=kf+1
    kc=kc+1
    f=1
    c=1
    fila=int(fll)
    columna=int(coll)
    x=1
    y=1
    flag=False
    quotes='"'
    MapaRuta = open(r"C:\Users\denni\OneDrive\Desktop\ima.txt",'w')
    MapaRuta.write('digraph {' + "\n")
    MapaRuta.write('node [shape=plaintext]' + "\n")
    MapaRuta.write('some_node [' + "\n")
    MapaRuta.write('label=<' + "\n")
    MapaRuta.write('<table border="0" cellborder="1" cellspacing="0">' + "\n")
    MapaRuta.write('<tr>' + "\n")
    MapaRuta.write('<td>' + "\n")
    MapaRuta.write(valor + "\n")
    MapaRuta.write('</td>' + "\n")
    for i in range(1,kc):
        MapaRuta.write('<td>' + "\n")
        MapaRuta.write(str(i) + "\n")
        MapaRuta.write('</td>' + "\n")
    MapaRuta.write('</tr>' + "\n")
    for i in range(1,kf):
        MapaRuta.write('<tr>' + "\n")
        MapaRuta.write('<td>' + "\n")
        MapaRuta.write(str(i) + "\n")
        MapaRuta.write('</td>' + "\n")
```

Figura 13.Creacion de código Graphviz
Fuente :Propia

rr	1	2	3	4	5	6
1	-	*	*	*	*	-
2	-	-	*	-	-	-
3	-	-	*	-	*	-
4	-	*	*	*	*	-
5	-	-	-	-	-	-
6	-	-	-	-	-	-

Figura 14. Grafica Creada Con Graphviz
Fuente :Propia

Conclusiones

- Las matrices Ortogonales nos permiten un manejo de la información mediante POO que un vector además permite el manejo de datos, ya sea para insertar, eliminar, buscar, modificar datos dentro de los nodos que contiene la lista.

- Graphviz es una herramienta que permite la visualización de estructuras TDA mediante graficas.

- Archivos XML nos permiten hacer una carga masiva de datos en un programa con una estructura muy básica, y fácil de manejar, así como importar datos de un programa
- Estructuras Secuencial permite que una instrucción siga a otra en secuencia.

- Estructuras condicionales son instrucciones que se pueden ejecutar o no en función del valor de una condición.

- Estructuras cíclicas permite ejecutar fragmentos de Código un umero limitado de veces

Referencias bibliográficas

Máximo 5 referencias en orden alfabético.

- Estructuras de programación secuenciales, cíclicas y condicionales Disponible en https://prezi.com/fmivkws5biwl/estructuras-condicionales-secuenciales-y-ciclicas/
- Estructura de XML y usos en http://www.maestrosdelweb.com/xmlusos/
- Estructura de Graphviz, Disponible en https://graphviz.org/

- Programación orientada a objetos, Disponible en https://desarrolloweb.com/articulos/499.php#:~:text=La%20programaci%C3%B3n%20Orientada%20a%20objetos%20se%20define%20como%20un%20paradigma,los%20objetivos%20de%20las%20aplicaciones.
- Programación orientada a objetos, Disponible en https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/
- http://www.rdebug.com/2010/10/matriz-ortogonal-estructura-de-datos-en.html#:~:text=una%20matriz%20ortogonal%20es%20una,las%20columnas%20representan%20los%20modelos.
- https://drive.google.com/file/d/1D3qqnpO33Qc7VH_9yIGz0qDXHQSmXamU/view