PROYECTO NO. 2

LENGUAJES FORMALES Y DE PROGRAMACIÓN

VACACIONES SEGUNDO SEMESTRE 2022

POR WENDI PAULINA VICENTE PÉREZ CARNET

202106484

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Paradigmas Aplicadas Paradigmas Orientada a Objetos: **Se basa en los conceptos de objetos y clases de objetos. Un objeto es una variable equipada con un conjunto de operaciones que le pertenecen o están definidas para ellos. El paradigma orientado a objetos actualmente es el paradigma más popular y día a día los programadores, estudiantes y profesionales tratan de tomar algún curso que tenga que ver con este paradigma, podría decirse, que programar orientado a objetos está de moda. Paradigmas Imperativa: Se caracteriza por el uso de variables, comandos y procedimientos, la programación funcional se caracteriza por el uso de expresiones y funciones. Un programa dentro del paradigma funcional, es una función o un grupo de funciones compuestas por funciones más simples estableciéndose que una función puede llamar a otra, o el resultado de una función puede ser usado como argumento de otra función. El lenguaje por excelencia ubicado dentro de este paradigma es el LISP.**

CLASES UTILIZADAS

- **Clase para automatas**
- **Clase de transiciones**
- **Clase Node**
- **Clase Listar**
- **Clase de Producciones**
- **Clase de Gramaticas**
- **Clase para Ventana de cargar archivo**
- **Clase para VentanaGLC_Arbol**
- **Clase para Ventana de Informacion GLC**
- **Clase para ventana de archivos autómatas**
- **Clase para ventana Pila reportes**
- **Clase para validar cadena.**

Funciones Creadas

- **Para cargar gramáticas**
- **Para mostrar gramaticas**
- **Para generar árbol gramáticas**
- **Para mostrar información**
- **Para cargar archivo para autómatas de pila**
- **Para generar Grafo**
- **Para validar cadena**
- **Para mostrar ruta de cadena**
- **Para mostrar una tabla de las transiciones**

Código para cargar gramática

```
def Cargar_Gramatica(nom):
    try:
        from itertools import groupby as eliminar_repetidos
        File = open(nom, 'r')
        x = 1
        lista_producciones = []
        lista_terminales = []
        lista_no_terminales = []
        lctxt = False
        for file in File:
            file = file.rstrip('\n')
            if x==1:
                nombre = file
                if (nombre in lista_nombres_Gramatica):
                    print(f'\nALERTA: La Gramatica *** {nombre} *** sera saltada debido a que ya existe ...\n')
                    x = '%'
                else:
                    lista_nombres_Gramatica.append(nombre)

            if x==2:
                file = file.replace(' ','')
                le = file.split(",")
                for l in le:
                    lista_no_terminales.append(l)

            if x==3:
                la = file.split(',')
                la.sort()
                l = list(lista for lista, _ in eliminar_repetidos(la))
                for listae in lista_no_terminales:
                    for listaa in lista_terminales:
                        if listaa == listae:
                            print('\nERROR: No fue posible crear el automata debido a que el alfabeto de entrada contiene simbolos no terminales\n')
                            lista_nombres_Gramatica.remove(nombre)
                            x = '%'

            if x==4:
                einic = file
                recorrer = False
                for ee in lista_no_terminales:
                    if einic == ee:
```

```
            lista_nombres_Gramatica.remove(nombre)
            x = '%'

        else:
            einicial = False

    if x != '%':
        if x>=5 and file != '%':
            files = file.split('>')
            files2 = files[1].split()
            if len(files2) >= 3:
                lctxt = True

            if ((files[0] in lista_no_terminales) and (files[1] in lista_no_terminales)):
                lctxt = True

            if (files[0] in lista_terminales) and (files[1] in lista_terminales):
                lctxt = True

            if len(files2) == 1 and files2[0] in lista_no_terminales:
                lctxt = True

            lista_producciones.append(Producciones(files[0], files2))

    if file == '%' and x!="%":
        if lctxt == True:
            lista_Gramaticas.append(Gramatica(nombre, lista_no_terminales, lista_terminales, einic, lista_producciones))
            lista_producciones = []
            lista_terminales = []
            lista_no_terminales = []
            x = 0
            lctxt = False
        else:
            print('\nERROR: No fue posible crear la gramatica debido a que la gramatica no es libre de contexto\n')
            lista_nombres_Gramatica.remove(nombre)
            lista_producciones = []
            lista_terminales = []
            lista_no_terminales = []
            x = 0

    if file == '%':
        lista_producciones = []
        lista_terminales = []
        lista_no_terminales = []
```

Código para mostrar árbol de gramática

```
def Arbol_Gramatica(resp):
    z = False
    while not z:

            if resp in lista_nombres_Gramatica:
                j = lista_nombres_Gramatica(resp)
                resp = j+1
            if resp !='-1' and int(resp)>=0:
                from graphviz import Graph

                dot = Graph(name='GramaticaLC', encoding='utf-8', format='png', filename='GramaticasLC')

                dot.attr(rankdir='TB', layout='dot', shape='none')

                numero = -1
                listaP = []
                indice = 0
                aux = 0
                lista_Nodos = []
                r_2 = int(resp)-1
                for nodo in lista_Gramaticas[r_2].producciones:
                    aux = 0
                    if lista_Nodos[:] != []:
                        for x in lista_Nodos:
                            if nodo.origen == x:
                                indice = aux
                            aux+=1
                    else:
                        numero+=1
                        dot.node(name='nodo'+str(numero), label=nodo.origen, shape='none')
                        lista_Nodos.append(nodo.origen)

                    for y in nodo.destinos:
                        numero +=1
                        dot.node(name='nodo'+str(numero), label=y, shape='none')
                        listaP.append(numero)
                        lista_Nodos.append(y)
                    for z in listaP:
                        dot.edge('nodo'+str(indice), 'nodo'+str(z))
                    listaP = []
                    aux = 0
                dot.render('GramaticasLC/'+lista_Gramaticas[r_2].nombre, format='png' ,view=True)

            else:
```

Código que se utilizó para validar cadenas

```
def Cadena(cadena,seleccion):
    Automata=Lista_Automata.Operar(seleccion)
    estado=0
    tran=0
    origenes=[]
    pila=[]
    cant_pila=0
    contar_cadena=0
    for caracter in cadena:
        contar_cadena+=1

        if(caracter in Automata[1]):
                if estado==0:
                    if caracter in Automata[6][tran]:
                        if caracter==Automata[6][tran][1]:#Lee entrada
                            if Automata[6][tran][3]!="$":#extraer
                                if len(pila)==0:
                                    print("No se puede extraer nada, por lo tanto no es valido")
                                else:
                                    if pila[cant_pila]==Automata[6][tran][3]:
                                        pila.remove(Automata[6][tran][3])
                                        print("Se extrajo"+Automata[6][tran][3])
                            elif Automata[6][tran][5]!="$":#inserta
                                pila.append(Automata[6][tran][5])
                                cant_pila+=1
                            origenes.insert(0,tran)
                            estado+=1
                            tran=0
                        else:
                            origenes.insert(0,tran)
                            estado+=1
                            tran=0
                        continue
                    elif caracter not in Automata[6][tran]:
                        if Automata[6][tran][1]=="$":
                            if caracter==Automata[6][tran][1]:#Lee entrada
                                if Automata[6][tran][3]!="$":#extraer
                                    if len(pila)==0:
                                        print("No se puede extraer nada, por lo tanto no es valido")
                                    else:
                                        if pila[0]==Automata[6][tran][3]:
```

```
            elif caracter not in Automata[6][tran]:
                if Automata[6][tran][1]=="$":
                    if caracter==Automata[6][tran][1]:#Lee entrada
                        if Automata[6][tran][3]!="$":#extraer
                            if len(pila)==0:
                                print("No se puede extraer nada, por lo tanto no es valido")
                            else:
                                if pila[0]==Automata[6][tran][3]:
                                    pila.remove(Automata[6][tran][3])
                                    print("Se extrajo"+Automata[6][tran][3])
                        elif Automata[6][tran][5]!="$":#inserta
                            if Automata[6][tran][5]==Automata[6][tran][5]:
                                pila.append(Automata[6][tran][5])
                                cant_pila+=1
                                origenes.insert(0,tran)
                                tran+=1
                                if caracter==Automata[6][tran][1]:#Lee entrada
                                    if Automata[6][tran][3]!="$":#extraer
                                        if len(pila)==0:
                                            print("No se puede extraer nada, por lo tanto no es valido")
                                        else:
                                            if pila[cant_pila]==Automata[6][tran][3]:
                                                pila.remove(Automata[6][tran][3])
                                                print("Se extrajo"+Automata[6][tran][3])
                                    elif Automata[6][tran][5]!="$":#inserta
                                        pila.append(Automata[6][tran][5])
                                        cant_pila+=1
                                    origenes.insert(0,tran)
                                    estado+=1
                                    tran=0
                                else:
                                    print("No es aceptado")
                                continue

                    else:#--------------
                        origenes.insert(0,tran)
                        estado+=1
                        tran=0

                else:
                    while caracter not in Automata[6][tran] or tran>=len(Automata[6]): #Buscar Transicion
                        tran+=1
                    if Automata[6][tran][0] in Automata[4]:#Estado inicial
                        if caracter==Automata[6][tran][1]:#Lee entrada
                            if Automata[6][tran][3]!="$":#extraer
```

```
        if Automata[6][tran][0] in Automata[4]:#Estado Inicial
            if caracter==Automata[6][tran][1]:#Lee entrada
                if Automata[6][tran][3]!="$":#extraer
                    if len(pila)==0:
                        print("No se puede extraer nada, por lo tanto no es valido")
                    else:
                        if pila[cant_pila]==Automata[6][tran][3]:
                            pila.remove(Automata[6][tran][3])
                            print("Se extrajo"+Automata[6][tran][3])
                elif Automata[6][tran][5]!="$":#inserta
                    pila.append(Automata[6][tran][5])
                    cant_pila+=1
                origenes.insert(0,tran)
                estado+=1
                tran=0
            else:
                origenes.insert(0,tran)
                estado+=1
                tran=0

    elif estado!=0:

        while Automata[6][origenes[0]][4]!=Automata[6][tran][0] or not caracter in Automata[6][tran]:#Buscar Transicion
            tran+=1
        if caracter==Automata[6][tran][1]:#Lee entrada
            if Automata[6][tran][3]!="$":#extraer
                if len(pila)==0:
                    print("No se puede extraer nada, por lo tanto no es valido")
                else:
                    if pila[len(pila)-1]==Automata[6][tran][3]:
                        pila.remove(Automata[6][tran][3])
                        print("Se extrajo"+Automata[6][tran][3])

            elif Automata[6][tran][5]!="$":#inserta
                pila.append(Automata[6][tran][5])
                cant_pila+=1
            origenes.insert(0,tran)
            estado+=1
            tran=0
            continue


    else:
        print("No puede ser aceptado")
        break
```

```
                else:
                    print("No puede ser aceptado")
                    break
    else:
        messagebox.showinfo("Información","Revisar entrada... un caracter no pertenece al alfabeto")
if pila[0] in Automata[2] and len(pila)==1:
    try:
        while Automata[6][origenes[0]][4]!=Automata[6][tran][0] or not pila[0] in Automata[6][tran]:#Buscar Transicion
            tran+=1
        if Automata[6][tran][3]!="$":#extraer
            if len(pila)==0:
                print("No se puede extraer nada, por lo tanto no es valido")
            else:
                if pila[len(pila)-1]==Automata[6][tran][3]:
                    pila.remove(Automata[6][tran][3])
                    print("Se extrajo"+Automata[6][tran][3])

        elif Automata[6][tran][5]!="$":#inserta
            pila.append(Automata[6][tran][5])
            cant_pila+=1
        origenes.insert(0,tran)
        estado+=1
        tran=0
        messagebox.showinfo("Informacion","Es aceptado")

    except:
        messagebox.showinfo("Informacion","No es aceptado")
else:
        messagebox.showinfo("Información","No es aceptado")
```