

---

## PROYECTO 1 - INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 2

---

202100081 – Javier Andrés Monjes Solórzano

### Resumen

Se debe desarrollar un programa que permita la compresión de señales de audio mediante un enfoque de agrupamiento de señales de audio mediante un enfoque de agrupamiento de patrones de frecuencia. Esto implica cargar datos desde archivos XML que contienen información sobre las señales, procesar estos datos y generar matrices reducidas que compartan los mismos patrones de frecuencia. Además, se busca crear visualizaciones gráficas de las señales y sus matrices reducidas utilizando la herramienta Graphviz. Para gestionar los nombres de las señales, se emplearán listas simplemente enlazadas, listas circulares, asegurando la consistencia de los datos y permitiendo generar informes detallados sobre las señales de audio.

### Palabras clave

1. Lista Simple
2. Lista Circular
3. XML
4. Grafo Dirigido
5. Matriz Binaria

### Abstract

*A program must be developed to enable audio signal compression using a frequency pattern clustering approach. This entails loading data from XML files containing signal information, processing this data, and generating reduced matrices sharing the same frequency patterns. Furthermore, the goal is to create graphical visualizations of the signals and their reduced matrices using the Graphviz tool. To manage signal names, singly linked lists and circular lists will be employed, ensuring data consistency and enabling the generation of detailed reports on audio signals.*

### Keywords

1. *Linked List*
2. *Circular Linked List*
3. *XML*
4. *Directed Graph*
5. *Binary Matrix*

## Introducción

Desarrollar este proyecto ofrece ventajas significativas tanto a empresas con recursos limitados como a aquellas que buscan optimizar gastos. Se enfoca en un Sistema de Control basado en Teoría de Colas, eficiente en la gestión de solicitudes y respuestas, sin una alta demanda de recursos de memoria. La implementación de matrices de base de datos usando Listas Enlazadas es esencial, aprovechando su naturaleza dinámica para evitar un consumo excesivo de memoria. Esto reduce los costos operativos y mejora la escalabilidad del sistema. En el ámbito empresarial actual, donde la agilidad y optimización son cruciales, esta solución ofrece eficacia para abordar los desafíos de la gestión de datos y solicitudes, impulsando la competitividad y eficiencia. Los detalles técnicos se presentan a continuación.

## Objetivo del Proyecto.

El proyecto tiene como objetivo simular y analizar un sistema de control utilizando un enfoque basado en teoría de colas. Este sistema de control procesa solicitudes y responde a ellas de acuerdo con ciertos parámetros configurables.

## Desarrollo del tema

Para poder desarrollar este programa el proceso fue dividido en varios pasos:

1. Realizar el Menú
2. Cargar el Archivo XML
3. Procesar el Archivo
4. Escribir Archivo Salida
5. Mostrar los Datos del Estudiante
6. Graficar las Matrices
7. Inicializar Sistema
8. Salida

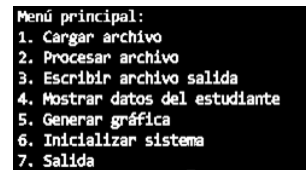
1. Realizar el Menú:

Este paso fue bastante sencillo, solamente consistió en desplegar un conjunto de opciones y pedir que ingrese una opción de

todas las opciones que ve en la pantalla siendo estas:

### Menú Principal:

1. Cargar archivo
2. Procesar archivo
3. Escribir archivo salida
4. Mostrar datos del estudiante
5. Generar gráfica
6. Inicializar sistema
7. Salida



```
Menú principal:
1. Cargar archivo
2. Procesar archivo
3. Escribir archivo salida
4. Mostrar datos del estudiante
5. Generar gráfica
6. Inicializar sistema
7. Salida
```

Figura 1. Menú principal en consola.

Fuente: elaboración propia

Para poder reconocer qué opción ingresó el usuario usaremos un input:



```
opcion = input("Ingrese una opción: ")
```

Figura 1.1 código de input.

Fuente: elaboración propia

Fuente: elaboración propia, o citar al autor, año y página. y al leer su respuesta es comparada con un if, else de varios casos para saber cuál de los métodos ejecutar, y este a su vez está encerrado por un try catch en caso de que el usuario decida ingresar un carácter inválido o lo haga por error.

```

if i <= opcion <= 7: # Verifica si la opción está en el rango válido
    if opcion == 1:
        ruta = input("Ingrese la ruta del archivo: ")
        confirmacion = input("¿Está seguro de que 'ruta' es la ruta correcta? (S/N): ").strip().upper()
        if confirmacion == 'S':
            archivo_xml_cargado = cargar_archivo(ruta)
            if archivo_xml_cargado:
                print("Archivo cargado con éxito.")
            else:
                print("No se ha cargado un archivo. Por favor, cargue un archivo primero.")
        elif opcion == 2:
            procesar_archivo()
        elif opcion == 3:
            if archivo_cargado:
                formato_salida = input("Elija el formato del archivo de salida (PNG o SVG): ").strip().lower()
                escribir_archivo_salida(formato_salida)
            else:
                formato_salida = input("Elija el formato del archivo de salida (PNG o SVG)(escriba en minúsculas): ").strip().lower()
                if formato_salida in ["png", "svg"]:
                    escribir_archivo_salida(formato_salida)
                else:
                    print(f"El formato de salida '{formato_salida}' no es válido. Por favor, elija un formato válido.")
            if not archivo_xml_cargado:
                print("No se ha cargado un archivo. Por favor, cargue un archivo primero.")
        elif opcion == 4:
            mostrar_datos_estudiante()
        elif opcion == 5:
            if archivo_xml_cargado:
                formato_salida = input("Elija el formato del archivo de salida (PNG o SVG): ").strip().lower()
                nombre_archivo = input("Ingrese el nombre del archivo de salida: ").strip()
                generar_grafica(nombre_archivo, formato_salida)
            else:
                print("No se ha cargado un archivo. Por favor, cargue un archivo primero.")
        elif opcion == 6:
            inicializar_sistema() # Esta opción permite inicializar el sistema
        elif opcion == 7:
            break
    else:
        print("Opción no válida. Ingrese una opción entre 1 - 7.")
else:
    print("Entrada no válida. Ingrese un número del menú (1-7).")

```

Figura 1.2 Código de Menú Principal

Fuente: elaboración propia.

## 2. Carga de Archivo XML:

**Carga de Datos:** El proyecto incluye una funcionalidad para cargar datos desde archivos XML, lo que sugiere que los datos de entrada se almacenan en estos archivos.

Para esto se le pide al usuario que ingrese la ruta absoluta del archivo incluyendo su extensión, ejemplo:

C: Proyecto\IPC2\_Proyecto1\_#202100081\entrada.xml

```

Ingrese una opción: 1
Ingrese la ruta del archivo: C:\Users\jared\Documents\Users\jared\Documents\USAC\Semestres\2023\Semestre 2do. Semestre\Curso\IPC_2\Lab\Proyectos\IPC2_Proyecto1_#202100081\entrada.xml

```

Figura 2.0 Ejemplo de Ruta

Fuente: elaboración propia

Si el archivo elegido no es un XML, entonces mostrara un error y regresara al menú, pero sí es un XML entonces la librería Element Tree para leer todas las etiquetas del archivo XML y las va identificando, luego de esto al identificar una Matriz almacena su nombre así como sus dimensiones y un apuntador al primer nodo de su lista de filas para poder acceder a la matriz

más adelante, luego, al llegar a las etiquetas de datos, obtiene el valor de X y de Y que contiene este dato para poder ubicarlos dentro de nuestras listas, y comparamos el valor de X con el id de la Fila de la matriz ya que X representa el valor de la Fila donde va ubicado el dato entonces decimos que si el id de la fila no es igual a la fila que dice el dato entonces avance al siguiente nodo hasta que ambos sean iguales, una vez son iguales le decimos a la lista que guarda este nodo que agregue a el nuevo nodo en este caso, el dato que nos indica el archivo XML, después de repetir este procedimiento para todas las Matrices encontradas en el archivo XML muestra un mensaje al usuario donde se indica que todo el archivo ya se ha cargado, así como viene mostrando todo el procedimiento de almacenarlo con mensajes en la consola.

```

[Está seguro de que 'C:\Users\jared\Documents\Users\jared\Documents\USAC\Semestres\2023\Semestre 2do. Semestre\Curso\IPC_2\Lab\Proyectos\IPC2_Proyecto1_#202100081\entrada.xml' es la ruta correcta? (S/N): S
Archivo XML cargado desde: C:\Users\jared\Documents\Users\jared\Documents\USAC\Semestres\2023\Semestre 2do. Semestre\Curso\IPC_2\Lab\Proyectos\IPC2_Proyecto1_#202100081\entrada.xml
Archivo cargado con éxito.

```

Figura 2.1 Mensaje Figura Cargada

Fuente: elaboración propia

## 3. Procesar de Archivo XML:

Aquí es en donde transformaremos la matriz en una Matriz de Patrones de Acceso, el primer paso para esto es hacer la matriz binaria a partir de la Matriz Ingresada, esto lo que hace es recorrer toda la matriz tanto en filas como en columnas e ir verificando por cada una “Si el valor de la matriz es diferente de 0 entonces el valor de la matriz = 1” y almacenar estos datos junto con su valor original en otras matrices para luego operarlas, ahora debemos obtener una fila de la matriz en una lista simple auxiliar y compararla con una fila de la matriz y ver si su valor binario es idéntico, si sí es idéntico entonces se sumará el valor original de cada dato de esa Fila que estamos comparando con el valor original de la Fila de la matriz con la cual acabamos de comparar la lista auxiliar, y debemos repetir el proceso hasta que

estén comparadas y sumadas todas las filas de la matriz binaria y la matriz resultado de este proceso es la Matriz de acceso, ahora bien, también debemos de crear una variable que lleve la cuenta de cuantas veces se repite cada Fila de la Matriz binaria para después de mostrarla al usuario y también para guardarla en el Archivo XML que se generará luego Ahora debemos almacenar esta matriz de patrones de acceso así como la frecuencia de cada fila en otra matriz e informarle al usuario de esto mediante un mensaje en la consola, justo como durante todo este proceso se le debe informar al usuario acerca de qué parte del procedimiento está realizando en determinado momento.

```
def procesar_archivo():
    # Aquí puedes implementar la lógica para procesar el archivo cargado previamente
    print("Calculando la matriz binaria...")
    # Aquí puedes implementar la lógica para calcular la matriz binaria
    for i in range(len(matriz)):
        for j in range(len(matriz[i])):
            if matriz[i][j] != 0:
                matriz_binaria[i][j] = 1

    print("Realizando suma de tuplas...")
    # Aquí puedes implementar la lógica para realizar la suma de tuplas
    for i in range(len(matriz_binaria)):
        for j in range(len(matriz_binaria[i])):
            if i != j and matriz_binaria[i] == matriz_binaria[j]:
                for k in range(len(matriz_binaria[i])):
                    matriz_acceso[i][k] += matriz_binaria[j][k]
    pass
```

Figura 3.0 Código de Matriz Binaria y Opcion2

Fuente: elaboración propia

#### 4. Escribir Archivo de Salida:

Se debe escribir un archivo XML utilizando los datos de las Matrices de Patrones de Acceso, así como escribir la frecuencia de sus filas y el grado de cada matriz junto con sus dimensiones, para esto se debe recorrer la Matriz de Patrones de Acceso creada en el inciso anterior, así como ir concatenando un String que vaya conteniendo las etiquetas del XML y e ir recorriendo la lista y obteniendo sus datos al mismo tiempo, para esto se debe de recorrer la matriz tanto en filas como en columnas, normalmente se utilizarían dos fors anidados para esto, pero debido a la naturaleza de una Lista Enlazada, y de los Tipos de Datos Abstractos esto no es posible porque los nodos

no son iterables, entonces debemos de utilizar un while dentro de otro while que vaya indicándole al nodo actual que después de concatenar un dato al string avance al siguiente nodo, esto debe de hacerse dentro de cada fila y al terminar de recorrer cada fila debe de repetir todo este proceso por cada una de las filas que tenga la matriz, para esto debemos de haber obtenido las dimensiones de la matriz antes de recorrerla ya que así tendremos cómo ir identificando cada posición de la matriz e indicarle al while cuando detenerse, después de recorrer toda la matriz ya solo debemos agregar las etiquetas de frecuencia para cada fila, y luego de esto decirle al programa que escriba este String en un archivo llamado "Archivo Salida.xml".



Figura 4.0 Ejemplo Archivo

Fuente: elaboración propia

#### 5. Mostrar los Datos del Estudiante:

Bien, solo se usan varios String para mostrar los datos del desarrollador de la aplicación, en este caso se mostrarán:

- Nombre Completo
- Carné
- Nombre Completo del Curso
- Carrera del Estudiante
- A qué semestre de dicha carrera pertenece el curso.

```
Nombre del estudiante: Javier Andrés Monjes Solórzano
Carné del estudiante: 202100081
Introducción a la Programación y Computación 2 Sección "A"
Ingeniería en Ciencias y Sistemas
4to Semestre
```

Figura 5.0 Ejemplo de Muestra Datos

Fuente: elaboración propia

#### 6. Graficar Las Matrcies:

En esta parte debemos de generar una gráfica en GraphViz de la Matriz que nos solicite el usuario, para esto solo debemos de desplegar todos las matrices almacenadas en nuestra matriz circular y pedirle al usuario que ingrese el número de la matriz que

quiere ver graficada y con un if else comparar el valor que ingrese el usuario con el id de la matriz circular, si este id es igual al de cualquier matriz en la lista circular entonces ya procede a graficarla si no es igual al id de ninguna matriz en la lista circular entonces debe mostrarle al usuario que ingreso mal el dato o bien que no se encontró ninguna matriz con ese número, para esto debemos utilizar un try catch, que si en dado caso el valor no es ninguno de los valores que se muestran tire una excepción, y vuelva a mostrar el menú y a pedir una opción hasta que el usuario ingrese una opción válida, después de esto, se procederá a graficar la matriz seleccionada para esto se utilizará un String que vaya concatenando el código de GraphViz (El cuál no verá el usuario, esto ya va escrito en el código del programa) y al mismo tiempo que este string se va concatenando también se va recorriendo la Matriz seleccionada por el usuario para poder obtener los datos correctos en cada fila y columna, así como también ir haciendo las conexiones correctas entre cada uno de los nodos de la matriz, junto con esta gráfica también se incluirán las dimensiones de Filas y Columnas de la matriz (llamadas "Nombre" y "Datos" respectivamente) así como también mostrará el nombre de la matriz. Después de concatenar todo el String, así como en el inciso anterior debemos de escribirlo todo a un archivo llamado "Grafica Matriz.dot" y utilizaremos la librería Web Browser para usar su método Open para que automáticamente abra la gráfica de la matriz generada.

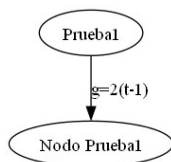


Figura 6.0 Ejemplo Gráfica

Fuente: elaboración propia

## 7. Salida:

Se cierra el programa usando una declaración break para salir del bucle while principal en el que se encuentra el menú.

```

elif opcion == 7:
    print("Saliendo del programa...")
    break
  
```

Figura 7.0 Código de Salida

Fuente: elaboración propia

```

Ingrese una opción: 7
Saliendo del programa...
  
```

Figura 7.1 Mensaje que muestra en consola

Fuente: elaboración propia

## Conclusiones

El programa fue dividido en varios "Mini Programas" o módulos para poder simplificarlos la tarea de programarlo y también para poder repartirnos el trabajo entre varias personas si fuera necesario hacerlo así, esta dedicación y atención al detalle se ve reflejada a lo largo de todo el programa, empezando por el método del menú donde fue necesario usar una condición que verificara si una opción era válida y después usar un while que repitiera el ciclo try catch ya que sin el while el bloque try solo se ejecuta una vez, entonces fue necesario encerrarlo todo en un while y escribirle dentro del if else a cada opción válida que cambiara el valor de la variable de False a True para que el código del while dejara de repetirse, este nivel de calidad lo hemos puesto en todas las partes del programa así que creemos que utilizarlo será una experiencia agradable para nuestros usuarios.

Módulos y Clases principales

### Archivo Amplitud.py

- *Clase Amplitud:* Esta clase representa el atributo "Amplitud" de las solicitudes en el sistema.

Contiene métodos para calcular y manipular la amplitud.

- **Archivo Tiempo.py**

*Clase Tiempo:* Esta clase gestiona el tiempo de procesamiento y respuesta de las solicitudes en el sistema.

- **Archivo Graficar.py**

*Función graficar:* Proporciona la funcionalidad para generar gráficos basados en los datos de la simulación.

- **Archivo Sistema.py**

*Clase simularSistema:* Esta clase representa el sistema de control y es responsable de la simulación de solicitudes y el análisis de su comportamiento.

- **Archivo Archivo.py**

*Clase Archivo:* Esta clase se utiliza para interactuar con archivos XML, lo que sugiere que se utilizan archivos XML para cargar datos en el sistema.

- **Archivo Cola.py**

*Clase Cola:* Este archivo contiene una implementación de una estructura de datos de cola, que podría ser utilizada para gestionar las solicitudes en el sistema.

### Uso del Proyecto

- **Carga de Datos:** El proyecto incluye una funcionalidad para cargar datos desde archivos XML, lo que sugiere que los datos de entrada se almacenan en estos archivos.
- **Simulación de Sistema:** La clase simularSistema se encarga de la simulación del sistema, procesando solicitudes y generando resultados.
- **Generación de Gráficos:** Utiliza la función graficar para visualizar los resultados de la simulación en forma de gráficos.

### Dependencias Externas

- No se mencionan dependencias externas en el código proporcionado, por lo que parece que el proyecto no utiliza bibliotecas externas específicas.

### Referencias bibliográficas

1. GeeksforGeeks. (2018, August 31). Linked List Data Structure.  
<https://www.geeksforgeeks.org/data-structures/linked-list/>
2. Pérez, G. M. C.-. (2014, July 15). Grafo Dirigido | Grafos.  
[Http://163.10.22.82/OAS/Estructuras\\_de\\_grafos](Http://163.10.22.82/OAS/Estructuras_de_grafos)  
[http://163.10.22.82/OAS/estructuras\\_de\\_grafos/grafito\\_dirigido.html](http://163.10.22.82/OAS/estructuras_de_grafos/grafito_dirigido.html)
3. Matriz Lineal y Matriz Binaria. (2014, June 22). WordPress.Com.  
<https://mejorfuturoya.wordpress.com/matriz-linea-matrizbinaria/>
4. GeeksforGeeks. (2018, September 7). Circular Linked List | Set 1 (Introduction and Applications).  
<https://www.geeksforgeeks.org/circular-linked-list/>
5. XML introduction - XML: Extensible Markup Language | MDN. (2021, February 19). MDN Web Docs.  
[https://developer.mozilla.org/en-US/docs/Web/XML/XML\\_introduction](https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction)

## Extensión: de cuatro a siete páginas como máximo

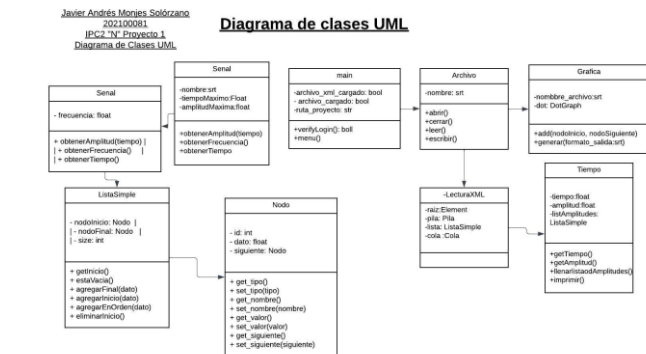


Figura 8.0 Diagrama UML

Fuente: elaboración propia

```

Menu principal:
1. Cargar archivo
2. Procesar archivo
3. Escribir archivo salida
4. Mostrar datos del estudiante
5. Generar grafica
6. Inicializar sistema
7. Salida
Ingrese una opción: 1
Ingrese la ruta del archivo: C:\Users\javier\Documents\Estados\IPC2\Semestre1\2023\Semestre1\Datos\Semestre1\Carpetas\IPC_2_LAB\Proyectos\IPC2_Project1_#00200801\entrada.txt
Cada registro de que C:\Users\javier\Documents\Estados\IPC2\Semestre1\2023\Semestre1\Datos\Semestre1\Carpetas\IPC_2_LAB\Proyectos\IPC2_Project1_#00200801\entrada.txt es la ruta de
archivo de salida.
Archivo de salida: C:\Users\javier\Documents\Estados\IPC2\Semestre1\2023\Semestre1\Datos\Semestre1\Carpetas\IPC_2_LAB\Proyectos\IPC2_Project1_#00200801\entrada.txt
Archivo cargado con éxito.
    
```

Figura 8.1 Consola, Programa Opcion1

Fuente: elaboración propia

```

Menú principal:
1. Cargar archivo
2. Procesar archivo
3. Escribir archivo salida
4. Mostrar datos del estudiante
5. Generar gráfica
6. Inicializar sistema
7. Salida
Ingrese una opción: 2
Calculando la matriz binaria...
Realizando suma de tuplas...
    
```

Figura 8.2 Consola, Programa Opcion2

Fuente: elaboración propia