|  |
| --- |
| Pruebas de Caja Blanca |
| ***“Sistema de pagos de alícuotas”*** |
|  |
|  |
| **Integrantes:**  **Lucas Góngora**  **Gabriel Manosalvas**  **Jairo Molina**  **Yandry Vélez**  **Fecha 2025-01-23** |

**Prueba caja blanca de** RF 1.Login

1. **Código FUENTE**

**import csv**

**import sys**

**def iniciar\_sesion():**

**clave\_incorrecta = True**

**intentos\_realizados = 0**

**intentos\_permitidos = 3**

**while clave\_incorrecta:**

**usuario,clave = obtener\_credenciales()**

**validacion\_exitosa = validar\_credenciales(usuario,clave)**

**if validacion\_exitosa:**

**clave\_incorrecta = False**

**print("\nCredenciales validas")**

**else:**

**print("\nCredenciales invalidas\n")**

**intentos\_realizados += 1**

**print ("Le quedan "+ str(intentos\_permitidos - intentos\_realizados) + " intentos")**

**if intentos\_realizados >= intentos\_permitidos:**

**print("Numero de intentos permitidos alcanzados")**

**sys.exit()**

**def obtener\_credenciales():**

**usuario=input("Ingresa tu usuario: ")**

**clave=input("Ingresa tu clave: ")**

**return [usuario,clave]**

**def validar\_credenciales(usuario,clave):**

**usuario\_autorizado,clave\_autorizada = credenciales\_autorizadas()**

**return usuario == usuario\_autorizado and clave == clave\_autorizada**

**def credenciales\_autorizadas():**

**usuario\_autorizado=""**

**clave\_autorizada=""**

**with open("./repositorios/usuario.csv","r") as credenciales\_csv:**

**lector\_csv = csv.DictReader(credenciales\_csv)**

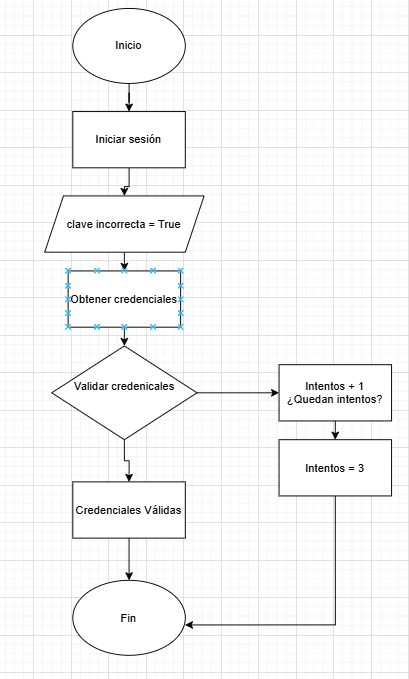
**for fila in lector\_csv:**

**usuario\_autorizado = fila["usuario"]**

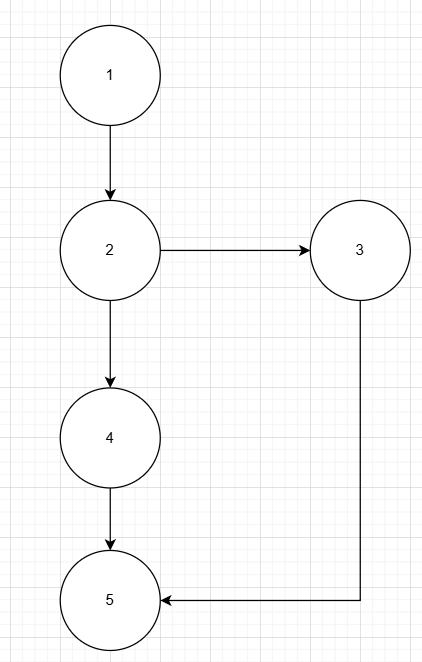
**clave\_autorizada = fila["clave"]**

**return [usuario\_autorizado,clave\_autorizada]**

1. **Diagrama de flujo**

****

1. **Grafo de Flujo (GF)**

****

1. **Identificación de rutas**

R1: 1,2,4,5

R2: 1,2,3,5

1. **Complejidad Ciclomática**

Se puede calcular de las siguientes formas:

* V(G) = número de nodos predicados(decisiones)+1

V(G)= 2+1=3

* V(G) = A – N + 2

V(G)= 5−5+2=2

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

**N:** Número de nodos

**Prueba de caja blanca de RF 2.menú**

1. Código fuente
2. **def display\_menu(self):**
3. **"""Display the main menu"""**
4. **while True:**
5. **print("\n=== Condominio Management System ===")**
6. **print("1. Agregar Residente")**
7. **print("2. Registrar Pago de Cuota de Condominio")**
8. **print("3. Mirar Detalles de Residente")**
9. **print("4. Generar informe de pagos pendientes")**
10. **print("5. Generar informe de pago completo")**
11. **print("6. Salir")**
13. **choice = input("\nEnter your choice (1-6): ")**
15. **if choice == '1':**
16. **self.add\_resident()**
17. **elif choice == '2':**
18. **self.record\_payment()**
19. **elif choice == '3':**
20. **self.view\_resident()**
21. **elif choice == '4':**
22. **self.generate\_outstanding\_payments\_report()**
23. **elif choice == '5':**
24. **self.generate\_payment\_report()**
25. **elif choice == '6':**
26. **print("Salienddo del Sistema")**
27. **break**
28. **else:**
29. **print("Opcion invalida. Intente de nuevo.")**

2. Diagrama de flujo

Diagrama

Descripción generada automáticamente

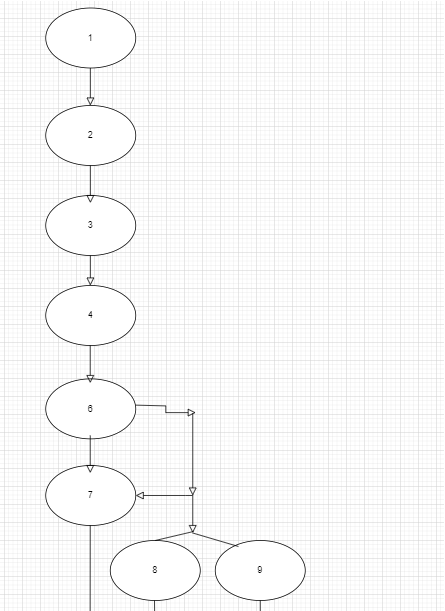
Diagrama

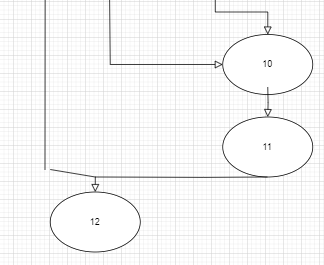
Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

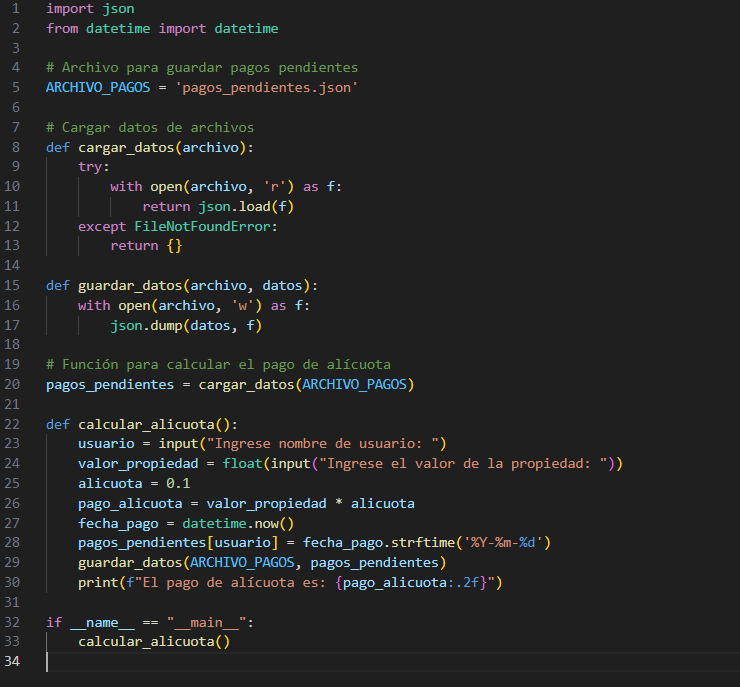
3. Grafo de flujo



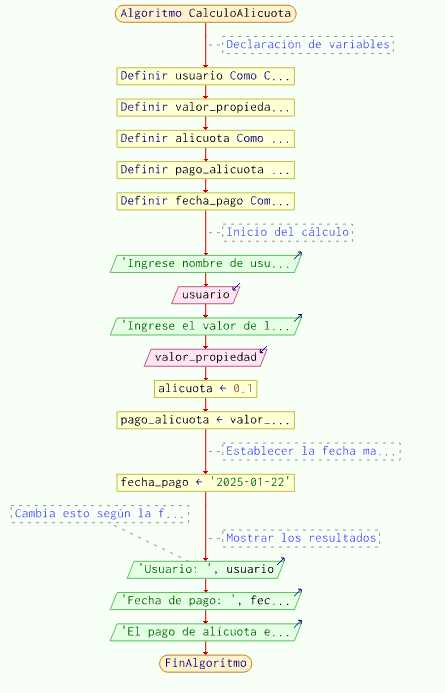


**Prueba caja blanca de RF 3. Notificación de pagos pendientes**

1. CÓDIGO FUENTE



2.DIAGRAMA DE FLUJO (DF)



1. **GRAFO DE FLUJO (GF)**

















1. **IDENTIFIACCIÒN DE LAS RUTAS (Camino basico)**

**Ruta 1:** Nodo 1 → Nodo 2 → Nodo 3 → Nodo 4 → Nodo 5 → Nodo 6 → Nodo7 → Nodo 8 → Nodo 9

1. **Complejidad ciclomática**

CC=E−N+2PCC = E - N + 2PCC=E−N+2P

* EEE = Número de aristas (líneas entre nodos en el grafo).
* NNN = Número de nodos.
* PPP = Número de componentes conexos (normalmente 1 para algoritmos simples).
* E=8E = 8E=8 (hay 8 líneas de conexión entre los nodos).
* N=9N = 9N=9 (hay 9 nodos en el grafo).
* P=1P = 1P=1 (todo está en un solo componente).

CC=8−9+2(1)=1CC = 8 - 9 + 2(1) = 1CC=8−9+2(1)=1

La **complejidad ciclomática** es:

CC=1CC = 1CC=1

Esto confirma que el algoritmo tiene solo un camino básico, sin bifurcaciones.

**Prueba caja blanca de** RF 5.Generación de reportes

1. **CÓDIGO FUENTE**

def generate\_payment\_report(self):

    """Generar el historial completo de pagos de la cuota del condominio"""

    print("\n=== Reporte del Historial de Pagos del Condominio ===")

    report\_data = []

    headers = ['Número de Unidad', 'Nombre del Propietario', 'Total Pagado', 'Saldo Pendiente']

    for unit\_number, data in self.residents.items():

        total\_paid = sum(payment['amount'] for payment in data['payment\_history'])

        report\_data.append({

            'Número de Unidad': unit\_number,

            'Nombre del Propietario': data['owner\_name'],

            'Total Pagado': f"${total\_paid:.2f}",

            'Saldo Pendiente': f"${data['outstanding\_balance']:.2f}"

        })

        print(f"\nNúmero de Unidad: {unit\_number}")

        print(f"Nombre del Propietario: {data['owner\_name']}")

        print(f"Total Pagado: ${total\_paid:.2f}")

        print(f"Saldo Pendiente: ${data['outstanding\_balance']:.2f}")

    if report\_data:

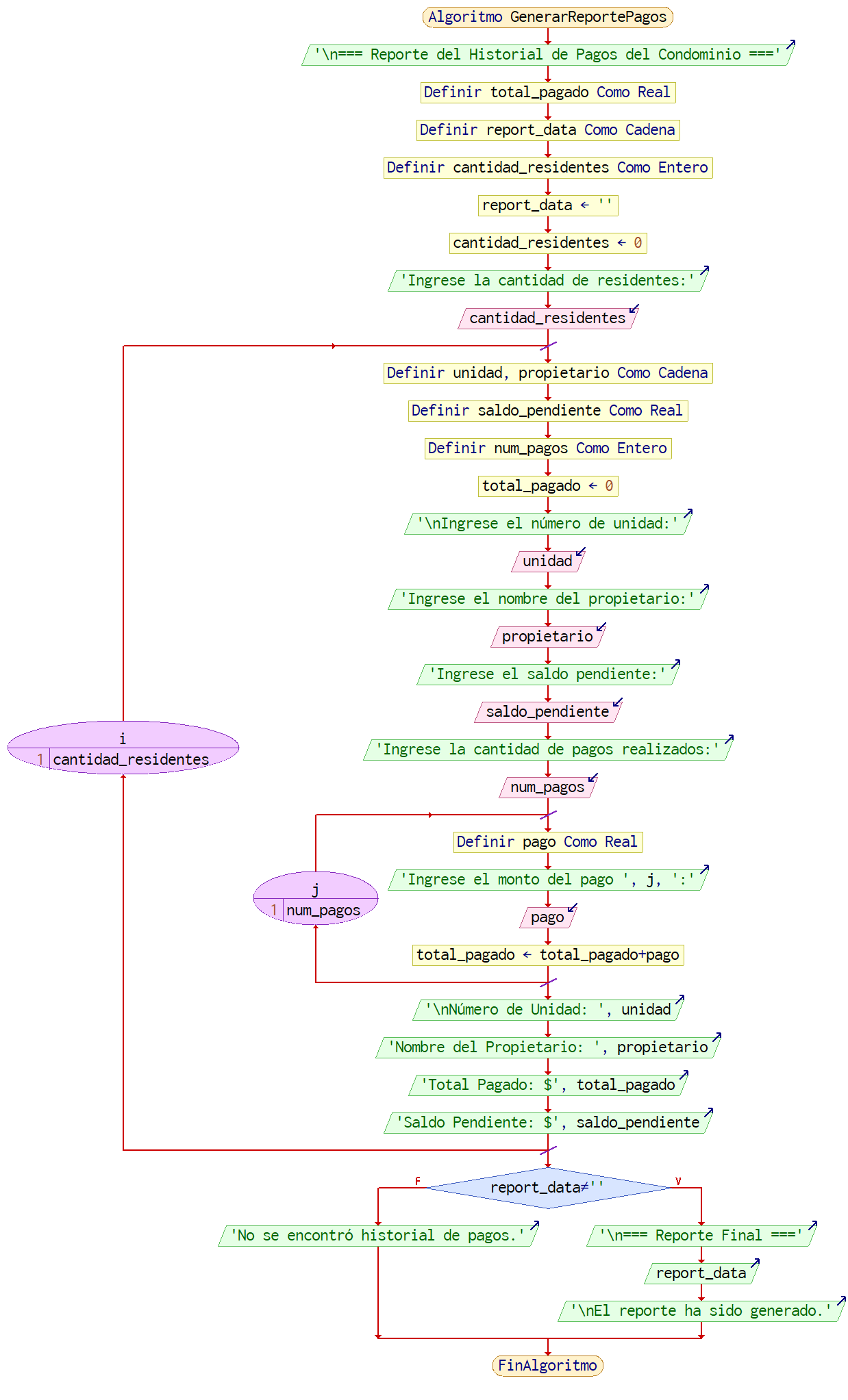
        self.generate\_report\_file(report\_data, 'payment\_history\_report.csv', headers)

        print("\nReporte guardado en 'payment\_history\_report.csv'")

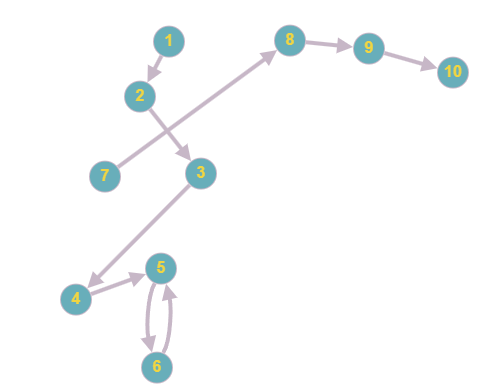
    else:

        print("No se encontró historial de pagos.")

1. **DIAGRAMA DE FLUJO (DF)**



1. **GRAFO DE FLUJO (GF)**



1. **IDENTIFIACCIÒN DE LAS RUTAS (Camino basico)**

Determinar en base al GF del numeral 4

**RUTAS**

**R1:** 1,2,7,3,8,9,10

**R2:** 1,2,3,4,5,6,5,3,8,9,10

1. **COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

* V(G) = número de nodos predicados(decisiones)+1

V(G)= 2+1=3

* V(G) = A – N + 2

V(G)= 12−10+2=3

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

**N:** Número de nodos