CICLOFORMATIVODEGRADOSUPERIOR-TÉCNICOEN ADMINISTRACIÓNDESISTEMASINFORMÁTICOSENRE DES

ADMINISTRACIÓNDESISTEMASOPERATIVOS

Nombre y apellidos:

- **Wuke Zhang , Álvaro Ginés Gómez**

Tarea2.1.Estudio de Caso: Problemas de Concurrencia y Sincronización de Procesos

Una empresa de servicios financieros ha implementado un sistema de procesamiento de transacciones en tiempo real que maneja datos financieros críticos, como cuentas bancarias, pagos y transferencias entre clientes. En este sistema, existen tres tipos de procesos:

1. Procesos de Validación de Transacciones
    - Estos procesos verifican la validez de cada transacción financiera antes de su aprobación.
    - Requieren acceso de lectura/escritura a un registro compartido donde se almacenan las transacciones pendientes y sus detalles.
    - Si dos procesos de validación acceden simultáneamente a este registro, existe un riesgo de que se produzcan condiciones de carrera, lo que podría provocar que las transacciones se aprueben o rechacen incorrectamente.
2. Procesos de Cálculo de Saldos
    - Estos procesos calculan y actualizan el saldo de las cuentas de los clientes en función de las transacciones aprobadas.
    - Solo pueden acceder al registro de transacciones aprobadas después de que los procesos de validación terminen su ejecución.
    - Debido a la naturaleza secuencial de este acceso, es crítico implementar un mecanismo de sincronización para que los procesos de cálculo de saldos no intenten acceder a los datos hasta que los procesos de validación hayan terminado.
3. Procesos de Generación de Reportes
    - Estos procesos generan reportes periódicos y estadísticas financieras a partir de las transacciones registradas y los saldos de los clientes.
    - Realizan operaciones intensivas de lectura, por lo que es fundamental que no bloqueen los procesos de validación o cálculo de saldos.
    - Requieren acceso en paralelo a los datos sin interferir con otros procesos que necesiten modificar estos datos.

Debido al crecimiento del volumen de transacciones, el sistema ha comenzado a experimentar problemas de rendimiento y fallas esporádicas, en las cuales el acceso simultáneo de varios procesos está provocando inconsistencias y errores en los datos financieros. Como administrador de sistemas, se te ha asignado la tarea de proponer un mecanismo de sincronización adecuado para resolver estos problemas y asegurar un flujo ordenado y seguro en el procesamiento de transacciones.

Actividad:

1. Identificar los Problemas de Concurrencia
    - Describe al menos dos problemas de concurrencia específicos que podrían surgir en este sistema debido al acceso simultáneo de los procesos a los recursos compartidos. En particular, menciona las condiciones de exclusión mutua y los problemas de inanición e interbloqueo.
    - Explica cómo estos problemas podrían afectar la estabilidad y el rendimiento del sistema.

Condición de carrera (Race Condition):

- **Descripción:** Los procesos de validación de transacciones acceden simultáneamente al registro compartido para aprobar o rechazar transacciones. Si no se gestiona adecuadamente, esto puede causar resultados inconsistentes, como transacciones duplicadas o rechazos incorrectos.
- **Impacto:** Las transacciones podrían perderse o duplicarse, afectando la integridad de los datos financieros y generando desconfianza entre los clientes.

Interbloqueo (Deadlock):

- **Descripción:** Los procesos de cálculo de saldos y generación de reportes podrían quedar en un estado en el que esperan indefinidamente por recursos bloqueados por otros procesos. Por ejemplo, un proceso de cálculo espera que se libere el registro compartido mientras el proceso de validación está bloqueado esperando el acceso al mismo recurso.
- **Impacto:** El sistema se paraliza, lo que retrasa el procesamiento de transacciones y afecta el rendimiento general del sistema.

Inanición (Starvation):

- **Descripción:** Los procesos de generación de reportes, al ser secundarios en prioridad, pueden no recibir acceso al registro compartido debido a la alta actividad de los procesos de validación y cálculo de saldos.
- **Impacto:** Los reportes no se generan a tiempo, afectando la toma de decisiones financieras y el cumplimiento normativo.

2. Investigar Mecanismos de Sincronización
    - Investiga y elige al menos dos mecanismos de sincronización, como semáforos, mutexes, monitores, o barreras, y explica cómo cada uno ayudaría a prevenir problemas de contención en este contexto.
      Semáforos:

- **Descripción:** Los semáforos son contadores utilizados para controlar el acceso a recursos compartidos, permitiendo un número limitado de accesos concurrentes.
- **Aplicación:**
    - **Validación de Transacciones:** Un semáforo binario puede asegurar que solo un proceso acceda al registro compartido a la vez, previniendo condiciones de carrera.
    - **Cálculo de Saldos:** Un semáforo de conteo garantizaría que este proceso comience solo cuando los procesos de validación hayan liberado el recurso.

      Monitores:

3. **Descripción:** Los monitores encapsulan datos compartidos y las operaciones sobre ellos, garantizando que solo un proceso ejecute una operación sobre el recurso a la vez.
4. **Aplicación:**
    - **Generación de Reportes:** Un monitor puede permitir múltiples accesos simultáneos de lectura mientras bloquea modificaciones por parte de procesos de validación o cálculo. Esto asegura que los reportes no interfieran con operaciones críticas.

    - Justifica por qué cada mecanismo es adecuado para resolver los problemas identificados en cada tipo de proceso (validación, cálculo de saldos y generación de reportes) en el sistema financiero.

Justificación de la Elección:

- **Semáforos:** Su flexibilidad permite controlar el acceso de múltiples procesos con diferentes prioridades.
- **Monitores:** Facilitan la gestión concurrente de datos compartidos sin la necesidad de diseñar mecanismos complejos de control manual.

    - Infórmate de cómo se redacta un **pseudocódigo** para la representación de soluciones por software.

Bueno antes del pseudocódigo haría el diagrama de flujo que sería lo ideal, tras eso si es opcional pasarlo a pseudocódigo que es básicamente expresar redactado el diagrama ya que independientemente del lenguaje que sea si sabes hacer el diagrama de flujo, luego ya es traducir al lenguaje o si quieres previamente al pseudocódigo para tener claridad y precisión para describir los algoritmos, las distintas estructuras de datos y procesos que vamos a utilizar.

El pseudocódigo es una herramienta que describe soluciones de software de forma clara y comprensible, sin depender de un lenguaje de programación específico. Para redactarlo, se deben seguir estos principios:

1. **Propiedades:** Ser legible, estructurado, abstracto y consistente.
2. **Elementos básicos:**
    - Declaraciones de inicio y fin (`Inicio`, `Fin`).
    - Variables y asignaciones (`variable = valor`).
    - Estructuras condicionales (`Si`, `Entonces`, `Sino`).
    - Bucles (`Mientras`, `Para`).
    - Funciones y procedimientos (`Función`, `Procedimiento`).
    - Entrada y salida (`Leer`, `Imprimir`).
    - Comentarios (`//` para explicar la lógica).
3. **Buenas prácticas:**
    - Usa nombres de variables descriptivos.
    - Emplea sangrías para mostrar jerarquías.
    - Divide procesos complejos en pasos más pequeños.
    - Sé explícito para evitar ambigüedades.

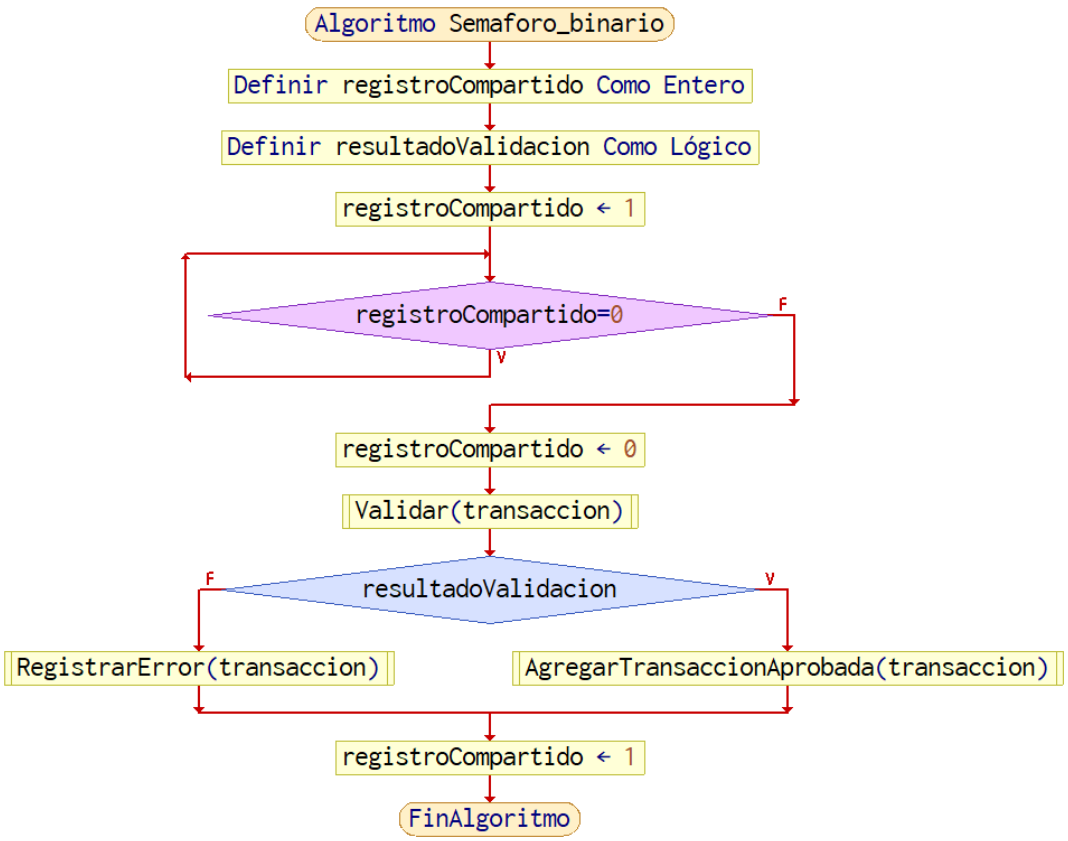
El pseudocódigo debe enfocarse en la lógica del algoritmo, priorizando claridad y simplicidad.

5. Implementación Práctica
    - Desarrolla los algoritmos de pseudocódigo de cómo se ejecutarían los diferentes procesos, los mecanismos de sincronización que propones para gestionar el acceso a un recurso compartido.

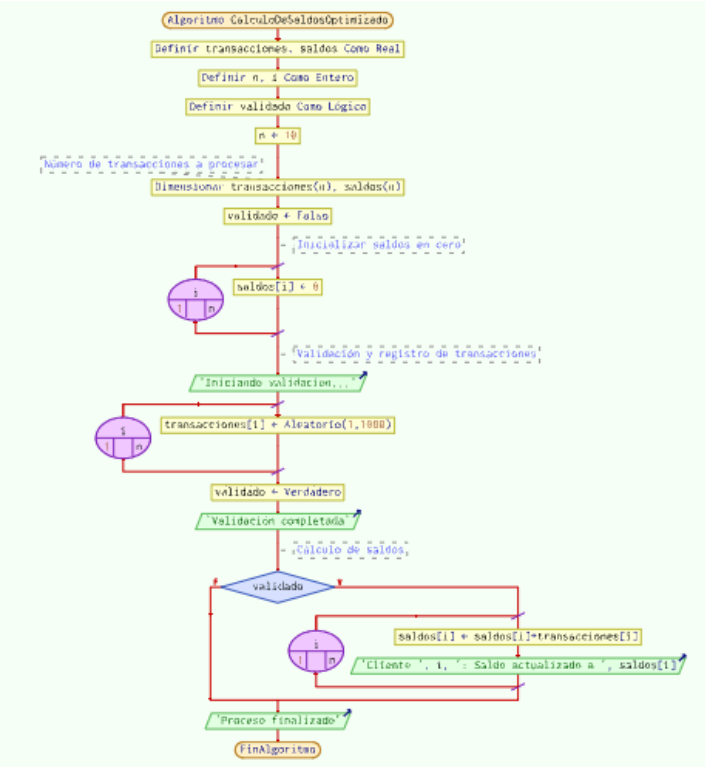
      Validación de Transacciones (Usando Semáforos Binarios)

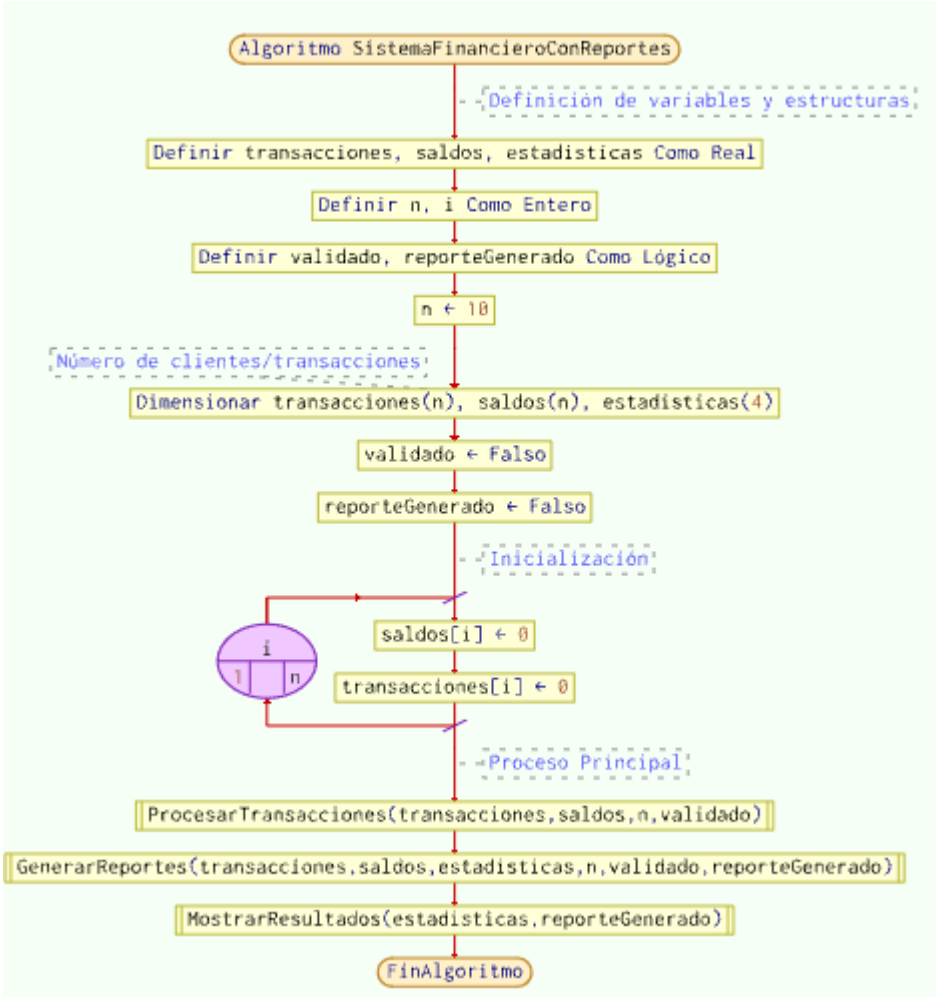
      Este algoritmo usa una combinación de estructura repetitiva para que while sea 0 el recurso está ocupado y no se pueda acceder y luego usamos la estructura alternativa para validar según que ejecutar.

```
Algoritmo Semaforo_binario
    Definir registroCompartido Como Entero;
    Definir resultadoValidacion Como Logico;
    registroCompartido ← 1;

    Mientras registroCompartido = 0 Hacer
        // Espera activa o uso de una función wait
    FinMientras

    registroCompartido ← 0  // Solicitar acceso exclusivo

    // Validar transacción
    Validar(transaccion)

    Si resultadoValidacion Entonces
        AgregarTransaccionAprobada(transaccion)
    Sino
        RegistrarError(transaccion)
    FinSi

    registroCompartido ← 1  // Liberar recurso
FinAlgoritmo

Subproceso Validar(transaccion)
    // Aquí va la lógica para validar la transacción
    resultadoValidacion ← Verdadero  // Cambiar según la lógica
FinSubproceso

Subproceso AgregarTransaccionAprobada(transaccion)
    // Lógica para agregar la transacción aprobada
FinSubproceso

Subproceso RegistrarError(transaccion)
    // Lógica para registrar el error
FinSubproceso
```



```
Algoritmo CalculoDeSaldosOptimizado
    Definir transacciones, saldos Como Real;
    Definir n, i Como Entero;
    Definir validado Como Logico;

    n ← 10; // Número de transacciones a procesar
    Dimension transacciones[n], saldos[n];
    validado ← Falso;

    // Inicializar saldos en cero
    Para i ← 1 Hasta n Hacer
        saldos[i] ← 0;
    FinPara

    // Validación y registro de transacciones
    Escribir "Iniciando validación...";
    Para i ← 1 Hasta n Hacer
        transacciones[i] ← Aleatorio(1, 1000);
    FinPara
    validado ← Verdadero;
    Escribir "Validación completada";

    // Cálculo de saldos
    Si validado Entonces
        Para i ← 1 Hasta n Hacer
            saldos[i] ← saldos[i] + transacciones[i];
            Escribir "Cliente ", i, ": Saldo actualizado a ", saldos[i];
        FinPara
    FinSi

    Escribir "Proceso finalizado";
FinAlgoritmo
```






```
Algoritmo SistemaFinancieroConReportes
    // Definición de variables y estructuras
    Definir transacciones, saldos, estadisticas Como Real;
    Definir n, i Como Entero;
    Definir validado, reporteGenerado Como Logico;

    n <- 10; // Número de clientes/transacciones
    Dimension transacciones[n], saldos[n], estadisticas[4];
    validado <- Falso;
    reporteGenerado <- Falso;

    // Inicialización
    Para i <- 1 Hasta n Hacer
        saldos[i] <- 0;
        transacciones[i] <- 0;
    FinPara

    // Proceso Principal
    ProcesarTransacciones(transacciones, saldos, n, validado);
    GenerarReportes(transacciones, saldos, estadisticas, n,
validado, reporteGenerado);
    MostrarResultados(estadisticas, reporteGenerado);
```

```
FinAlgoritmo

SubProceso ProcesarTransacciones(transacciones Por
Referencia, saldos Por Referencia, n, validado Por Referencia)
   Definir i Como Entero;

   Escribir "Procesando transacciones...";
   Para i <- 1 Hasta n Hacer
      transacciones[i] <- Aleatorio(100, 1000);
      saldos[i] <- saldos[i] + transacciones[i];
   FinPara
   validado <- Verdadero;
   Escribir "Transacciones procesadas.";
FinSubProceso

SubProceso GenerarReportes(transacciones, saldos, estadisticas
Por Referencia, n, validado, reporteGenerado Por Referencia)
   Definir total, promedio, maximo, minimo Como Real;
   Definir i Como Entero;

   Si validado Entonces
      // Cálculo de estadísticas
      total <- 0;
      maximo <- saldos[1];
      minimo <- saldos[1];

      Para i <- 1 Hasta n Hacer
         total <- total + saldos[i];
         Si saldos[i] > maximo Entonces
            maximo <- saldos[i];
         FinSi
         Si saldos[i] < minimo Entonces
            minimo <- saldos[i];
         FinSi
      FinPara

      promedio <- total / n;

      // Almacenar estadísticas
      estadisticas[1] <- total;
      estadisticas[2] <- promedio;
      estadisticas[3] <- maximo;
      estadisticas[4] <- minimo;

      reporteGenerado <- Verdadero;
   Sino
      Escribir "Error: Los datos no han sido validados.";
```

```
    FinSi
FinSubProceso

SubProceso MostrarResultados(estadisticas, reporteGenerado)
    Si reporteGenerado Entonces
        Escribir "=== REPORTE FINANCIERO ===";
        Escribir "Total de saldos: $", estadisticas[1];
        Escribir "Saldo promedio: $", estadisticas[2];
        Escribir "Saldo máximo: $", estadisticas[3];
        Escribir "Saldo mínimo: $", estadisticas[4];
        Escribir "========================";
    Sino
        Escribir "No hay reportes generados.";
    FinSi
FinSubProceso
```

- Explica por qué elegiste este mecanismo para el caso específico y cómo tu solución permitirá que cada tipo de proceso acceda a los recursos sin causar inconsistencias o bloqueos. Describe los beneficios que aporta la solución en términos de seguridad y rendimiento del sistema.

La solución propuesta utiliza **mecanismos de sincronización** (bloqueos y semáforos) para coordinar el acceso a los recursos compartidos entre los distintos procesos:

1. **Procesos de Validación**: Se utilizan bloqueos exclusivos para evitar condiciones de carrera al acceder a registros de transacciones pendientes, asegurando que solo un proceso modifique los datos a la vez.
2. **Procesos de Cálculo de Saldos**: Se implementa semáforos para asegurar que los cálculos solo ocurran después de que las validaciones hayan terminado, garantizando que los saldos se actualicen con transacciones válidas.
3. **Procesos de Generación de Reportes**: Se permite el acceso concurrente para generar reportes sin bloquear los procesos de validación o cálculo, ya que solo necesitan realizar operaciones de lectura.

Beneficios: La solución asegura la **integridad de los datos**, **evita condiciones de carrera**, y permite **acceso eficiente y concurrente** a los datos, mejorando tanto la **seguridad** como el **rendimiento** del sistema.

Entregar un documento en formato pdf con la realización de los tres apartados, incluyendo los nombres de todos los miembros del grupo y la bibliografía consultada.