POLITÉCNICO COLOMBIANO JAIME ISAZA CADAVID	FORMATO PARA REALIZAR GUÍA DE LABORATORIO – GUIA TRABAJO INDEPENDIENTE	Código: Versión: 01

PROGRAMA ACADÉMICO: Ingeniería informática	
Módulo: Sistemas operativos	
Área: Ingeniería Informática	**Docente:** Jhon Jairo Arango Tobón
Fecha de entrega	**21 de enero 2022**
Sustentación	**22 de enero 2022**

Miguel Ángel Bedoya Bonilla CC. 1001418288

Santiago Restrepo Idárraga CC. 1001370117

1. **Algoritmos de programación de la CPU Laboratorio N° 4,**
2. **Objetivo de Aprendizaje**
3. **Guía de desarrollo**

   Implementar un programa en un lenguaje de programación(C, C++, C#, Java, otro), que realice la simulación del algoritmo Round Robin.

   La Aplicación debe solicitar total de procesos.

   Tiempo de llegada de cada proceso.

   Cuántas entradas/salidas va a realizar cada proceso.

   Cuánto requiere cada proceso.

   Se debe mostrar la cola de los procesos en estado listo y el diagrama de Gantt. Debe

   calcular los tiempos de vuelta y los tiempos de espera y sus promedios.

   Nota: el tamaño del Quantum es de 50 milisegundos y el intercambio de 10 milisegundos.

4. **Entregables**

Se debe entregar el programa fuente y el ejecutable el cual se debe subir a la plataforma del poli virtual, además un informe con la explicación del laboratorio como lo desarrollo, además de los inconvenientes encontrados, forma como lo solucionó y que aprendió.

informe:

Este laboratorio fue desarrollado en lenguajes web como HTML5 y CSS3 y el lenguaje de programación JavaScript debido a su fácil interacción con la parte visual de una página y su versátil sintaxis que nos permite simular a la perfección el algoritmo de planificación de Round Robin. Utilizamos métodos nativos del lenguaje para realizar la simulación y renderizamos toda la información en elementos HTML los cuales dan al usuario final los resultados definitivos.

Primeramente comenzamos con unos inputs encargados de recibir las entradas por parte del usuario las cuales harán del programa uno realmente dinámico. Contamos con las siguientes entradas:

- Input “Tiempo de llegada en milisegundos”
- Input “Necesita de CPU en quantum”
- Input “Gasta entrada/salida en Quantums “
- Input “Necesita de CPU en quantum (Entrada / Salida)”

También contamos con 3 botones los cuales serán los encargados de las tres funciones principales del programa las cuales son:

- Guardar/Añadir procesos
- Añadir entradas/salidas a los procesos
- Mostrar resultados

Una vez presionado uno de estos 3 botones se realizarán las comprobaciones correspondientes y se validará si puede o no ejecutarse la simulación.

Luego de haberse validado los datos y de haber comprobado que no haya inconsistencias, pasaremos a almacenar los datos en estructuras de datos de JavaScript llamadas Arrays, más específicamente, Arrays de objetos JSON los cuales son otra estructura de datos muy útil para la manutención de su información interna.

Internamente, en una función, ordenamos los datos suministrados por el usuario de manera ascendente por el orden de llegada de los procesos. Inmediatamente después se comienza con la lógica para el envío de los procesos a la cola de procesos en estado listo y al diagrama de Gantt.

Contamos con un objeto que guarda los datos de la tabla que usábamos físicamente en Excel (**ver figura 1.1**)

Proceso	Tiempo de llegada en ms	Necesita de CPU en quantum	Gasta entrada/salida en Quantums	Necesita de CPU en quantum	Gasta entrada/salida en Quantums	Necesita de CPU en quantum	proceso	Regresa a cola de listo en tiempo
P0	0	1	2	2	2	1	p0	550 + 100 = 65
P1	30	2	2	1	2	2	p1	495 + 100 = 59
P2	70	3	-	-	-	-		
P3	140	3	-	-	-	-		

Figura 1.1 Ejemplo de tabla de Excel haciendo el algoritmo manualmente

Esta tabla es la que lleva el control de los procesos con sus tiempos y necesidades de CPU, esos mismos datos estarán en el objeto interno del código. Adicionalmente tenemos otros atributos que nos darán información como saber si un proceso fue ejecutado anteriormente.

Luego encontramos el ciclo principal que correrá hasta que esta tabla se quede sin procesos, los procesos saldrán de la tabla si y solo sí ya no necesitan CPU ni siquiera en las entradas y salidas. En cada iteración se escogerá el proceso que será enviado a la cola de procesos en estado listo; Solo podrá ser enviado un proceso por iteración y dependerá de ciertos criterios establecidos en un condicional (El condicional más importante del programa). Primero se escogerá un proceso provisional que a continuación será comparado con el resto de procesos en un ciclo for en el cual estará internamente el condicional del que se habló anteriormente. Si un proceso cumple con los criterios establecidos entonces él se convertirá en el nuevo proceso escogido provisional y se seguirá comparando con los procesos que queden de la tabla.

Una vez terminado el ciclo for, el proceso provisional que haya quedado escogido anteriormente será el proceso definitivo escogido en esta iteración.

Lo siguiente será añadir ese proceso a la cola de procesos en estado listo y modificar algunos de sus parámetros como restarle -1 a los quantums que necesita de CPU y cambiar a falso el atributo de si fue escogido anteriormente (wasChosenBefore), al proceso que estaba en la cabeza de la cola de procesos en estado listo; Inmediatamente quitamos ese proceso que se encuentra en la cabeza de la lista.

Seguido de esto, añadimos el proceso escogido al diagrama de Gantt y enviamos por parámetro el id del proceso, su tiempo de comienzo de ejecución y el tiempo en que termina su ejecución.

En el código usamos una variable (currentTime) que controla el tiempo en el que nos encontraremos en cualquier momento en el código. Esta importante variable será actualizada justo después de que el proceso escogido sea enviado al diagrama de Gantt; su tiempo aumentará en razón del tamaño del quantum en ms (QUANTUMSIZE) y el tiempo de intercambio en ms (INTERCHANGE).

Luego de actualizar el tiempo, verificamos si el proceso escogido todavía necesita CPU, en caso de que sí actualizaremos el parámetro interno que tiene este llamado **comingTime** el cual controla el momento en el que el proceso volverá a la cola de procesos en estado listo, ya sea por entradas y salidas o porque se esté ejecutando en el momento. En caso de no necesitar CPU, buscaremos en sus entradas/salidas y actualizaremos su **comingTime** de acuerdo a cuándo vuelve según la entrada/salida; cada entrada/salida estará guardada en un arreglo de objetos del proceso llamado **InOut** y cada vez que se llegue a este punto del código, eliminaremos la entrada y salida que esté al frente de **InOut**, así hasta que se acaben las entradas y salidas. En caso de no entrar en ninguna de las anteriores condiciones, entonces significa que el proceso escogido debe ser eliminado de la tabla inmediatamente.

Luego, volveremos a repetir el ciclo while hasta que no queden procesos en la tabla.

Una vez hayamos salido del ciclo while procederemos a ejecutar la función **calculateFinalResults()** para calcular los datos finales requeridos por el usuario como lo son:

- Tiempos de vuelta de cada proceso
- Tiempos de espera
- Los promedios de los 2 tiempos anteriores

Después de ejecutada esta función, todos los datos habrán sido pintados en pantalla y la simulación habrá terminado.

Observaciones:
Realizar en grupos de trabajo
El laboratorio debe sustentarse