Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación
Área de Ingeniería en Computadores
Lenguajes, Compiladores e intérpretes (CE3104)
Segundo Semestre 2017
Tarea# 3

1. **Objetivo General**
   - ➔ Desarrollar una aplicación que permita reafirmar el conocimiento del **paradigma de programación imperativo**.

2. **Objetivos Específicos**
   - ➔ Desarrollar una aplicación en el lenguaje de programación C.
   - ➔ Aplicar los conceptos de programación imperativa.
   - ➔ Crear y manipular listas como estructuras de datos.

3. **Datos Generales**
   - ➔ El valor del proyecto: 5%
   - ➔ Nombre código: StructBenchmark
   - ➔ La tarea debe ser implementada en grupos de no más de 2 personas.
   - ➔ La **fecha de entrega** es 14/11/2017.
   - ➔ Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo al reglamento.

4. **Descripción del caso.**

   La tarea consiste en realizar un benchmark de los algoritmos de ordenamiento básicos como el BubbleSort, SelectionSort, InsertionSort, MergeSort, QuickSort y medir su rendimiento para poder compararlo con lo que la teoría dice, como estructuras de datos deben de desarrollarse listas enlazadas dobles con sus operaciones básicas.

   Deben de ejecutarse pruebas utilizando diferentes cantidades de elementos (para mantenerlo simple los datos serán enteros) en las estructuras (largo de la lista), listas ordenadas (mejor caso), listas ordenadas aleatoriamente (caso promedio) y listas ordenadas aleves (peor caso) medir la duración de la ejecución del algoritmo con estos datos de prueba y graficar este conforme crece N, en está gráfica debería aparecer también la gráfica del valor teórico para poder comparar la tasa de crecimiento obtenida vs la tasa de crecimiento esperada[1].

   Por simplicidad, utilizaremos únicamente enteros para realizar las pruebas, pero las estructuras/algoritmos deben de estar programadas de manera genérica.

[1] Cada uno de los algoritmos tendrá 3 gráficas peor caso, mejor caso y caso promedio, en total serian 15 gráficos.

Algoritmo	Peor Caso	Mejor Caso	Caso promedio
BubbleSort	O(n^2)	O(n)	O(n^2)
SelectionSort	O(n^2)	O(n^2)	O(n^2)
InsertionSort	O(n^2)	O(1)	O(n^2)
MergeSort	O(n log n)	O(n log n)	O(n log n)
QuickSort	O(n^2)	O(n log n)	O(n log n)

5. **Entregables**
   5.1. Código fuente comentado.
   5.2. Manual de usuario.
6. **Documentación**

1. Artículo científico (paper) donde se presenten los 15 gráficos obtenidos y su respectivo análisis (se debe respetar la estructura de un paper).
2. Se deberá entregar un documento que contenga:
   2.1. Manual de usuario: cómo ejecutar el programa.
   2.2. Descripción de las estructuras de datos desarrolladas.
   2.3. Descripción detallada de los algoritmos desarrollados.
   2.4. Problemas conocidos: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.
   2.5. Actividades realizadas por estudiante: Este es un resumen de las bitácoras de cada estudiante (estilo timesheet) en términos del tiempo invertido para una actividad específica que impactó directamente el desarrollo del trabajo, de manera breve (no más de una línea) se describe lo que se realizó, la cantidad de horas invertidas y la fecha en la que se realizó. Se deben sumar las horas invertidas por cada estudiante, sean conscientes a la hora de realizar esto el profesor determinará si los reportes están acordes al producto entregado.
   2.6. Problemas encontrados: descripción detallada, intentos de solución sin éxito, soluciones encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.
   2.7. Conclusiones y Recomendaciones del proyecto.
   2.8. Bibliografía consultada en todo el proyecto
3. Bitácora en digital, donde se describen las actividades realizadas, desde reuniones con el compañero de trabajo, investigaciones, consultas, etc. Se debe describir todo por más insignificante que sea, esto demostrará si ustedes están trabajando en realidad. Este es su diario de trabajo, llevan seguimiento de todo en el tiempo, imaginen que si un compañero los releva en su trabajo, le bastaría con leer sus bitácoras para seguir el trabajo.

7. **Evaluación**
    1. El proyecto tendrá un valor de un 80% de la nota final, debe estar funcional.
    2. La documentación tendrá un valor de un 20% de la nota final, cumplir con los requerimientos especificados en la documentación no significa que se tienen todos los puntos, se evaluará que la documentación sea coherente, acorde al tamaño del proyecto y el trabajo realizado, no escatimen en documentación.
    3. Cada grupo recibirá una nota en cada uno de los siguientes apartados Código y Documentación.
    4. El profesor no sólo evaluará la funcionalidad del proyecto, esto quiere decir que aunque el proyecto este 100% funcional esto no implica una nota de un 100, ya que se evaluarán aspectos de calidad de código, aplicación del **paradigma imperativo**, calidad de documentación interna y externa y trabajo en equipo.
    5. No se revisarán funcionalidades parciales, ni funcionalidades no integradas.
    6. Es responsabilidad de cada miembro del grupo conocer su código, el profesor puede preguntar a cualquier miembro del grupo que le explique alguna funcionalidad/porción de código.
    7. De las notas mencionadas en los puntos 1 y 2 se calculará la Nota Final del Proyecto.
    8. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
    9. Aun cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
        9.1. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
        9.2. Si no se entrega el punto 4 de la documentación se obtiene una nota de 0.
        9.3. Si el código y la documentación no se entregan en la fecha indicada se obtiene una nota de 0.
        9.4. Si el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
        9.5. Si el grupo no cuenta con los equipos necesarios para realizar la revisión y no avisó al profesor de esta situación obtendrá una nota de 0.
        9.6. El código debe ser desarrollado en **el lenguaje de programación C** utilizando el **paradigma de programación imperativo**, en caso contrario se obtendrá una nota de 0.
    10. Cada grupo tendrá como máximo 30 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
    11. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
    12. Cada grupo es responsable de llevar los equipos requeridos para la revisión.
    13. Durante la revisión únicamente podrán participar los miembros del grupo, asistentes, otros profesores y el coordinador del área.
    14. Las revisiones se realizan con los estudiantes matriculados en el curso, cualquier persona fuera de estos y los mencionados en el punto 13, no pueden participar en la revisión.
    15. Después de enviada la nota final del proyecto el estudiante tendrá un máximo de 3 días hábiles para presentar un reclamo siempre y cuando la funcionalidad esté completa.