ANÁLISIS Y DISEÑO DE ALGORITMOS
Ing. Román Martínez M.
Semestre Enero-Mayo 2019

TAREA #2

El objetivo de esta tarea es que repases algunos de los conceptos básicos de programación y estructuras de datos a través de un ejercicio de programación, y que comiences a relacionar esto con la técnica de análisis de algoritmos que estamos aprendiendo en el curso. Específicamente, repasarás y pondrás en práctica los conceptos de archivos texto en C++, búsqueda y Hashing. Por otro lado, esta tarea es también una oportunidad para que aprendas y uses las herramientas más básicas de STL en caso de que no las conozcas.

Como apoyo a la solución de esta tarea, está a tu disposición el siguiente material de apoyo:

- En caso de que necesites repasar el tema de archivos texto, en la página del curso en Bb, en la sección de Material de apoyo, encontrarás un documento con un resumen de lo esencial que requeriremos en el curso, y podrás complementar con tutoriales que hay en Internet.
- En caso de que necesites repasar el tema de Hashing, en la página del curso en Bb, en la sección de Material de apoyo, encontrarás la presentación de este tema que se usa en el curso de Estructuras de datos.
- Para el tema de STL hay muchos tutoriales en Internet que puedes consultar. Más que un estudio a profundidad, en esta tarea utilizarás a los contenedores *vector* y *list*, y al algoritmo *sort*, por lo tanto, puedes concentrar tu estudio específicamente en estos elementos. En la página del curso en Bb, en la sección de Material de apoyo, encontrarás una presentación con un resumen de STL que te podrá ser útil.
- Para complementar tu estudio del tema de análisis de algoritmos puedes leer las siguientes referencias si lo deseas o tienes tiempo:
    - Libro de texto de Neapolitan y Naimipour: Capítulo 1, en particular las secciones 1.3 y 1.4. El libro es accesible en su primer capítulo desde books.google.com. Aunque la 5ª edición es la más reciente, puedes utilizar cualquier edición anterior, pues no hay muchos cambios en el primer capítulo.
    - Capítulo 1 y 2 de los apuntes de Elizabeth Pérez y René MacKinney que se encuentran en la página del curso en Bb en la sección de Material de apoyo.
    - Capítulo 3 y 4 del manual de Víctor Valenzuela que se encuentra en la página del curso en Bb en la sección de Material de apoyo.

Para la codificación de tus programas deberás utilizar el estándar establecido por el departamento de Computación del campus Monterrey. Estos estándares están en un documento que puedes encontrar en la sección de Material de apoyo de la página del curso. La falta de uso de estos estándares puede penalizar hasta el 10% de la calificación de esta tarea.

CASO A RESOLVER
Se requiere un programa que sirva para BUSCAR UN DATO en un archivo de texto. El contexto de aplicación será considerando que se requiere saber si una matrícula de un alumno del Tec, pertenece o no a un egresado EXATEC. El programa pedirá al usuario la matrícula a buscar e indicará si ésta es de un egresado EXATEC o no. El programa permitirá buscar tantas matrículas como el usuario desee. Las matrículas serán manejadas como números enteros (sin anteponer A0). Para realizar este proceso, el programa utilizará un archivo que contiene las matrículas de los egresados a considerar (en la realidad, en el Tec hay más de 200 mil EXATECs, pero el archivo de prueba que se te proporciona será una simulación con miles de datos). Este archivo contiene una matrícula por renglón, y las matriculas aparecen en forma aleatoria (no ordenadas). El archivo deberá cargarse a memoria una sola vez al iniciar la ejecución del programa, y las búsquedas se realizarán sobre la estructura de datos en que se guarden.
Como puedes observar, el caso es un simple problema de búsqueda, y puede ser resuelto de múltiples formas. Para esta tarea, tomaremos 3 propuestas de solución, las implementaremos y las evaluaremos con las herramientas de análisis de algoritmos que estás aprendiendo.

<u>Propuesta de solución #1</u>
Cargar los datos del archivo en un arreglo de enteros, y realizar la búsqueda secuencial de la matrícula **directamente sobre el arreglo**, **comparando dato por dato** hasta encontrarlo, o bien, hasta llegar al final del arreglo sin encontrarlo. Este algoritmo simplemente va accesando en el arreglo dato por dato, haciendo la comparación y desechándolos mientras no se encuentre la matrícula que se busca. Define el tamaño del arreglo con un valor constante que puedas adaptar en el código dependiendo de la cantidad de datos que tenga el archivo de datos con el que se hará la prueba.

Propuesta de solución #2

Cargar todos los datos del archivo en un **vector de STL,** ordenar los datos por medio del **algoritmo de ordenamiento que provee STL**, y finalmente realizar la búsqueda de la matrícula, pero aplicando el proceso de la **búsqueda binaria** sobre el vector ya ordenado.

Propuesta de solución #3

Cargar todos los datos del archivo en una **tabla de hash** y aplicar la búsqueda sobre la tabla. La tabla debe ser implementada con las siguientes condiciones:

- La tabla será un arreglo estático de **1000 posiciones**.
- La **función de hashing** a utilizar es una función que selecciona a los **últimos 3 dígitos** de la matrícula como valor de posición en la tabla.
- Se utilizará **encadenamiento externo para administrar las colisiones**. Los datos de cada posición en la tabla serán almacenados en una **lista de STL**, considerando el **orden de llegada**, es decir, NO será una lista ordenada. La búsqueda implicará una **búsqueda secuencial aleatoria en la lista**.

¿Cuál de las 3 propuestas crees que es la mejor opción de solución?

Para responder esta pregunta, implementaremos cada propuesta y contrastaremos la ejecución empíricamente, pero también analizaremos los algoritmos para fundamentar la respuesta.

IMPLEMENTACIÓN Y ENTREGABLES

La atención y solución a esta tarea tendrá varios momentos en que deberás ir entregando resultados parciales. A continuación se indican las fechas y las entregas esperadas:

1. **Viernes 25 de enero.** Deberás ya tener entendida y razonada la tarea. Si tienes que estudiar algunas de las herramientas de trabajo, ya deberás haberte preparado. Si tienes dudas de la misma, ya deberás haber contactado al profesor para que te apoye. Además, las 3 propuestas de solución deberás analizarlas intuitivamente y con los conocimientos que tienes, y en base a eso, decidir cuál es la que recomendarías como la mejor para implementar como solución profesional al caso. TU respuesta deberás capturarla antes de la sesión de clase en la página de Facebook a través de un voto. En clase se revisarán los resultados de la votación y deberás ir preparado para defender tu voto con argumentos sólidos.
2. **Martes 29 de enero.** Deberás tener terminada y probada la implementación de la propuesta de solución #1. También, si tu matrícula es par, deberás tener terminada y probada la implementación de la propuesta de solución #2, y si tu matrícula es impar, la propuesta de solución #3. En la sesión de clase entregarás una impresión del código de las dos propuestas que implementaste, y deberás estar preparado para explicar el orden de complejidad de los algoritmos, así como para exponer en clase la ejecución de tus programas desde tu computadora.
3. **Viernes 1 de febrero.** Deberás tener terminada y probada la implementación de la propuesta de solución faltante (la #2 si tienes matrícula impar, y la #3 si tienes matrícula par) y entregar impreso el código correspondiente. En caso de que hayas hecho modificaciones o correcciones a los códigos entregados previamente, también deberás entregarlos impresos de nuevo, especificando el porqué de esos cambios. Durante la sesión se podrá pedir que muestres la ejecución de cualquiera de tus programas. Adicionalmente, durante ese mismo día, deberás subir a la página de Facebook, un video de no más de 5 minutos en el que muestres en pantalla la ejecución de tus 3 programas con el archivo de prueba que se te facilitó. En el video deberás comentar cuál es la solución más eficiente y cómo lo comprobaste o demostraste, aclarando cuáles son los órdenes de complejidad en tiempo y espacio de cada programa, y cómo esto soporta tu decisión de la mejor solución. Para la sesión de clase deberás estar preparado para llenar una forma de captura en la que indicarás el orden de complejidad de algunos de los procesos que programaste.

El archivo de prueba estará publicado en el mismo espacio en Bb de la tarea, sin embargo, es recomendable que tú generes tus propios archivos de prueba, inicialmente más pequeños.

Es opcional que uses la función de medición de tiempo para comprobar de forma precisa cuál es el programa que resuelve más rápido el caso. Si agregas esta funcionalidad en la entrega final y la muestras en tu video, podrás obtener algunos puntos extras.

Esta es una tarea en la que los aprendizajes que desarrolles son básicos y esenciales para el curso y la carrera, por lo tanto, es una tarea OBLIGATORIA en el curso. Esto significa que no se puede dejar de hacerla y entregarla, y quien por algún motivo no la realice en tiempo, tiene la obligación de realizarla para que el resto de los puntos por las tareas del semestre sean tomados en cuenta.
No dejes que las dudas o problemas al programar te agobien y ante cualquier duda, aclaración o asesoría, busca con confianza el apoyo del profesor.