Tarea 2

Manejo de datos y punteros

Entrega: miércoles 24 de abril a las 21:00 hrs.

1. Introducción

En esta tarea, se le pide que escriba un programa en `C` que maneje una base de datos de ciudades y pueblos del mundo. Concretamente, el programa deberá:

1. Recibir un nombre de archivo como argumento al llamar el programa (`argv`)
2. Leer el archivo y organizar los datos
3. Indexar los datos según determinados criterios
4. Responder consultas que efectuará el usuario por comandos entregados durante la ejecución (`stdin`)
5. Finalizar la ejecución cuando el usuario lo pida

Los detalles de estas tareas se entregan a continuación.

2. Funcionamiento

Su programa deberá maximizar la velocidad de ejecución y la eficiencia en el uso de memoria, según las pautas que se dictan en este enunciado. La evaluación tendrá en cuenta estos dos criterios.

Lectura del archivo

Su programa deberá leer la base de datos de ciudades desde un archivo de texto. Para ello, primero deberá verificar que el archivo exista, y si no, lo informará al usuario y terminará su ejecución. En dicho archivo, cada fila representará una ciudad, de la que entregará los siguientes datos como columnas separadas por punto y coma:

```
ID;Geoname_ID;Name;Country_Code;Country_name;Population;Elevation;Timezone;Coordinates
```

Nótese que las *coordenadas* corresponden a la latitud y la longitud (en ese orden), separadas por coma.

Para maximizar la velocidad de ejecución, los datos deberán permanecer en memoria durante la ejecución del programa. Para maximizar el uso de memoria, evite dentro de lo posible manejar copias múltiples de un mismo dato.

Indexación

Luego de leído el archivo, su programa deberá crear *índices*, de manera de encontrar rápidamente los datos que el usuario solicite. Concretamente, se espera un orden de las ciudades por elevación, población, latitud y nombre. Para evitar la redundancia en el uso de memoria, se recomienda la construcción de los índices como arreglos de punteros.

Consultas en tiempo de ejecución

Una vez construídos los índices, se entrega el control al usuario para consultas. Las consultas que el usuario podrá efectuar harán que el programa despliegue los datos de la ciudad señalada, y son las siguientes:

- `POBL [K]`: la k^{a} mayor ciudad por población (o la –k^{a} menor si k es negativo). P.ej., `POBL 1`, `POBL -100`.
- `ELEV [K]`: la k^{a} ciudad de mayor elevación (o la –k^{a} de menor si k es negativo). P.ej., `ELEV 500`, `ELEV -1000`.
- `LAT [K]`: la k^{a} ciudad más al norte (o la –k^{a} más al sur si k es negativo). P.ej., `LAT 50`, `LAT -1000`. Aquí los `LAT -1` a `LAT -10` lógicamente corresponderán a lugares en Chile, Argentina o la Antártica.
- `[ciudad]`: despliega los datos de la ciudad mencionada. Por simplicidad, el programa buscará si existe una ciudad cuyo nombre coincida exactamente con el nombre dado por el usuario (aquí, una letra mayúscula no coincide exactamente con una minúscula). P.ej. `Santiago` (pero `SANTiago` fallará).
- `SALIR`: finaliza la ejecución del programa y libera la memoria.

En caso de empates, cualquier ciudad que cumpla el criterio se considerará como respuesta correcta. P.ej., si hubiera dos ciudades con el mismo nombre, puede mostrar una cualquiera entre las dos; si hubiera dos ciudades con la misma población empatadas en el k^{o} lugar, cualquiera de ellas podrá ocupar el lugar k o el k+1.

3. Recomendaciones

- No reinvente la rueda. Investigue sobre las funciones `qsort()` y `bsearch()` de la librería estándar (`<stdlib.h>`) de `C`, para ordenar y buscar respectivamente. Son un poco crípticas de usar, pero pueden ahorrarle trabajo y son razonablemente eficientes.

- Recuerde que su tarea será evaluada por funcionalidades. Esto tiene dos implicancias prácticas importantes: 1) más vale una tarea en que funcionan la mitad de las cosas que una que está a punto de funcionar pero no compila, y 2) antes de desesperarse o bloquearse, divida la tarea en funcionalidades e impleméntelas una a una de manera de ir asegurando puntaje.
- Comience el desarrollo con anticipación. Es muy probable que en el camino encuentre problemas que requieran tiempo para pensar o discutir con el profesor o los ayudantes.

4. Consideraciones Generales

Su programa debe ser robusto frente a datos que no corresponden. Esto quiere decir que su programa no debe caerse en caso de que el usuario haga operaciones matemáticas no válidas (p.ej., dividir por cero si fuera el caso). Sin embargo, se puede asumir que el usuario será amigable, en el sentido de que cuando le soliciten un número este no ingresará letras, por ejemplo.

5. Evaluación y Entrega

Esta tarea puede ser resuleta en grupos de máximo 2 personas. El plazo para la entrega de la tarea vence impostergablemente el miércoles 24 de abril a las 21:00 hrs..

Formato de entrega: Subir un único archivo con el código de su programa al módulo de tareas de la página del curso, con un nombre de archivo que incluya ambos apellidos de los autores de la tarea separados por guiones, seguido de "Tarea2.c", de la forma "`Prat Chacon-Carrera Pinto-Tarea2.c`". El incumplimiento de este requisito de nombre de archivo será penalizado con un descuento de un punto en la nota. Los archivos compilados no serán tomados en cuenta, si se llega a subir sólo un archivo compilado, este será ignorado, y será evaluado con nota 1.

En el momento de compilación, se deberá indicar las siguientes *flags* al compilador:

```
-Wall -Wextra -Wundef -Werror -Wuninitialized -Winit-self
```

Aquellas tareas que no compilen de la forma básica (`gcc -o salida entrada`) serán evaluadas con nota 1. Las que compilen, pero se caigan durante la ejecución, serán evaluadas con nota máxima 3.

Su programa será evaluado con múltiples casos de prueba y deberá ser capaz de ejecutarlos todos de manera correcta. De fallar en algún caso de prueba serán descontados los puntos correspondientes a dicho caso.

6. Consideraciones de Trabajo

El trabajo en esta tarea puede hacerse en parejas. Cuide su tarea para que no sea copiada parcial o íntegramente por otros. Todas las tareas entregadas serán comparadas por un sistema

automático de detección de plagio. Cualquier sospecha de copia (de otras tareas o de internet) podrá ser denunciada, investigada y eventualmente penalizada de conformidad al Reglamento del Alumno.