



Tarea 2

Base de Superhéroes

Entrega: martes 3 de mayo a las 21:00 hrs.

1. Introducción

Los superhéroes que vemos en las películas, comics y series tienen diferentes niveles de fuerza, velocidad, inteligencia, etc. Un grupo de fanáticos lo han contratado para que finalmente puedan resolver qué superhéroe es el mejor de todos. Para esto usted dispone de una base proporcionada por la API (Application Programming Interface) de superhéroes, ya procesada.

Naturalmente el flujo de datos en cuestión es abundante, por lo que es necesario utilizar herramientas eficientes tanto en velocidad como en memoria, con el propósito de optimizar los recursos computacionales al realizar informes a partir de ellos. Su tarea es desarrollar un programa en C que actúe de forma similar a una Base de Datos, almacenando en memoria los datos y permitiendo realizar distintas consultas a ellos, de tal forma que sea posible obtener resultados más trabajables para los fanáticos que lo contrataron.

2. Formato de los documentos

El programa deberá leer un archivo que contiene toda la información necesaria. El formato del documento es el siguiente:

```
id(int); name(char*); intelligence(int); strength(int); speed(int); durability(int);  
power(int); combat(int); full-name(char*); alter-egos(char*); aliases__001(char*);  
place-of-birth(char*); first-appearance(char*); publisher(char*); alignment(char*);  
gender(char*); race(char*); height__001(char*); height__002(char*); weight__001(char*);  
weight__002(char*); eye-color(char*); hair-color(char*); work__occupation(char*);  
work__base(char*); connections__group-affiliation(char*); connections__relatives(char*)
```

Los elementos del archivo estarán separados por el carácter ';' y los datos deberán ser almacenados por cada superhéroe y posteriormente referenciados en arreglos de punteros para su manejo. Esto quiere decir que el arreglo solo va a guardar la referencia a cada superhéroe y no la información completa.

3. Opciones/Flujo del programa

Su tarea deberá poder ser operada de dos maneras: 1) mediante opciones entregadas por línea de comandos (argc y argv) y 2) mediante interacciones con el usuario por la entrada estándar (stdin).



Su programa deberá ser capaz de responder a un conjunto de posibles consultas, según se explica más abajo, siguiendo las opciones entregadas por el usuario por alguno de los dos mecanismos.

3.1. Nombres compuestos

Su programa deberá ser capaz de distinguir los nombres compuestos sin el uso de comillas, por ejemplo en la consulta por línea de comandos:

```
./programa -tophero power Black Cat
```

el valor de `argc` será 5, sin embargo el superhéroe referido se llama “Black Cat” y su nombre debe ser tratado como una unidad.

3.2. Modo terminal (interacción por entrada estándar)

En este caso, el programa deberá guardar los datos en memoria y recibir consultas por `stdin`. El programa no debe terminar hasta que se ingrese el comando “salir”. Para llamar al programa en modo terminal, debe llamárselo desde la línea de comandos con el flag `-terminal`:

```
./programa -terminal
```

3.3. Búsqueda por superhéroe en característica

Dado un superhéroe entregado por `argv` o `stdin`, debe mostrar los 10 superhéroes inmediatamente superiores al superhéroe en cuestión, ya sea en poder u otra característica dada. Después de ello, se debe ofrecer al usuario la opción de seleccionar un superheroe de esta lista para ver sus demás características.

```
./programa -tophero power Black Cat (si es por argv)
tophero power Black Cat (si es por stdin)
```

En la figura 1 se aprecia un ejemplo de output.

3.4. Búsqueda por valor en característica

Dado un valor entregado por `argv` o `stdin`, debe entregar los 10 superheroes que esten inmediatamente por arriba del primer superhéroe que tenga este valor. Después se debe poder seleccionar un superhéroe que esté en esta lista para ver sus demás características.

```
./programa -topvalue power 80 (si es por argv)
power 80 (si es por stdin)
```

En la figura 2 se aprecia un ejemplo de output.



```
Black Cat: 23
          1) Stephanie Powell: 24
          2) Katniss Everdeen: 24
          3) Hawkeye II: 24
          4) A-Bomb: 24
          5) Star-Lord: 25
          6) Nick Fury: 25
          7) Kraven the Hunter: 25
          8) James Bond: 25
          9) Forge: 25

          de que superheroe desea ver su informacion?
```

Figura 1: ejemplo de output de la función `tophero`



```
Bloodhawk: 80
Quicksilver: 81
Maxima: 81
Junkpile: 81
Toxin: 82
Red Hulk: 82
Mysterio: 82
Krypto: 82
T-850: 83
Professor Zoom: 83

de que superheroe desea ver su informacion?
```

Figura 2: ejemplo de output de la función `topvalue`

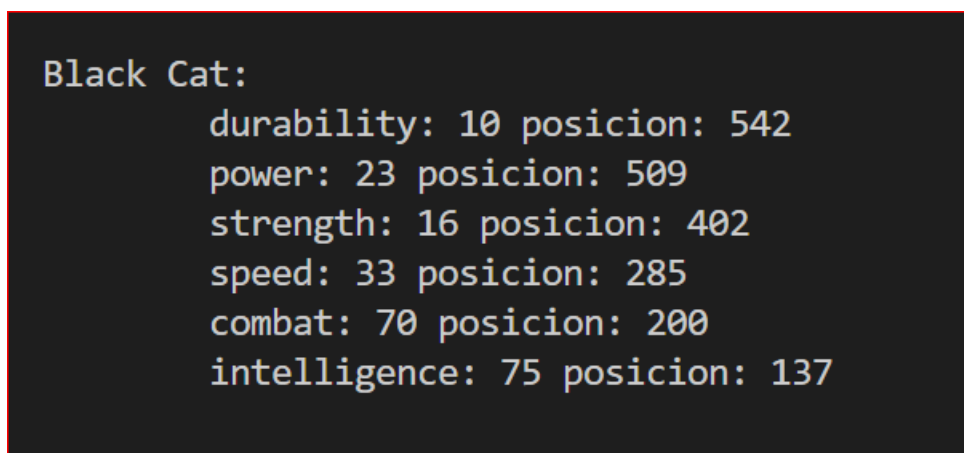


3.5. Búsqueda por superhéroe en todas sus características

Dado un superheroe entregado por `argv` o `stdin`, debe mostrar la posición y valor del superhéroe en cada una de las características comparables.

```
./programa -hero Black Cat (si es por argv)
hero Black Cat (si es por stdin)
```

En la figura 3 se aprecia un ejemplo de output.



```
Black Cat:
    durability: 10 posicion: 542
    power: 23 posicion: 509
    strength: 16 posicion: 402
    speed: 33 posicion: 285
    combat: 70 posicion: 200
    intelligence: 75 posicion: 137
```

Figura 3: ejemplo de output de la función `hero`

4. Consideraciones

Su programa no debe caerse frente a consultas mal realizadas (p.ej: numeros mayores al maximo o negativos o un superheroe inexistente) entregando el mensaje correspondiente informando por qué no se ejecutó la consulta. Además todo el ordenamiento de datos debe generarse por arreglos.

El uso de memoria, *memory leaks*, y velocidad el programa son factores importantes para esta entrega.

5. Evaluación

Los siguientes elementos serán considerados a la hora de evaluar:

1. Lectura y procesamiento de los archivos
2. Funcionamiento de las consultas posteriores (sección 3.2)



3. Funcionamiento de la consulta de superheroe en característica (sección 3.3)
4. Funcionamiento de la consulta de valor en característica (sección 3.4)
5. Funcionamiento de la consulta de superheroe en todas las características (sección 3.5)

Los programas que cumplan con estas funcionalidades correctas, tendrán nota 5,5 (sin contar descuentos por formalidades no funcionales) y se ganarán el derecho a competir con las otras tareas en una competición en que se ponderará la velocidad, el uso de memoria y la cantidad de *memory leaks*. La nota final estará dada por:

$$Nota = 5,5 + 1,5 * \frac{(n - k)}{n - 1} - d \quad (1)$$

Donde n es la cantidad de personas que ingresaron en la competencia, k el lugar obtenido, y d los descuentos no funcionales, en caso de haberlos. De esta forma el programa con mejor funcionamiento aspirará a la nota máxima (7) y el resto se irá distribuyendo de acuerdo a la eficiencia de su programa y la cantidad de personas que lograron implementar toda la funcionalidad.

Dado que se esta evaluando el uso de punteros **se descontará una décima por cada elemento en un arreglo de largo definido**, es decir, `char hero[4] = "Zoom"` tendrá un descuento de 4 décimas en la nota final.

6. Consideraciones de trabajo

Trabajo entre dos personas como máximo.

El trabajo en esta tarea debe ser hecho solo por Ud. y su pareja. 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 copia será penalizada, recibiendo el mismo castigo tanto quien copia como el autor original. También es considerada copia cualquier ayuda externa recibida directamente en la tarea, sin importar si proviene de un alumno del curso, de la universidad, o de otro lugar. El castigo será establecido por el Consejo de la Facultad, siendo como mínimo un 1,0 de promedio en el curso.

7. Compilación y Entrega

El plazo para la entrega de la tarea vence impostergablemente el martes 3 de mayo a las 21:00 hrs.

Formato de entrega: Subir un solo archivo con todo el código fuente de su programa al módulo correspondiente a la tarea en la página del curso en Canvas, con el nombre de archivo "APELLIDO1-APELLIDO2-Tarea2.c", reemplazando "APELLIDO1" y "APELLIDO2" según corresponda (ej. PRAT-O'HIGGINS-Tarea2.c). Los archivos compilados no serán tomados en cuenta, si se llega a subir solo un archivo compilado, este será ignorado, y evaluado con nota 1.



Su tarea deberá compilar sin *warnings*, con el estándar más estricto. Para asegurarse de ello, llame al compilador de la siguiente forma:

```
gcc -std=c99 -Wall -Wextra -Wundef -Werror -Wuninitialized -Winit-self archivo.c -o salida
```

Aquellas tareas que no compilen de la forma normal serán evaluadas con nota 1. Las que compilen, pero se caigan durante la ejecución, serán evaluadas con nota máxima 3.

Su programa podría ser 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.