TICS311: Tarea #3

Universidad Adolfo Ibañez

Ahmad Armoush
ahmad.armoush@edu.uai.cl

Danilo Bórquez Paredes
danilo.borquez.p@uai.cl

Wilson González
wilson.gonzalez@edu.uai.cl

Miguel Romero
miguel.romero.o@uai.cl

26 de abril de 2022

Objetivos

- Internalizar conceptos de tablas de hash y listas
- Trabajo con memoria dinámica y estructuras
- Uso de archivos de cabecera

1. Vamos bien, pero debemos mejorar

Estamos en buen camino! Su trabajo ha impresionado a sus superiores, pero han habido problemas de rigidez. Las listas de enemigos llegan cada vez con más frecuencia, por lo que se necesita una estructura más dinámica que la presentada en la tarea anterior. Los archivos siguen siendo los mismos, como se muestra a continuación:

mensaje.txt

```
first_name,danger_category,attack_prob
Kearney Carding,1,0.2733129
Antonin Revans,1,
Conney Lochrie,5,
Anton Sellack,2,
Lory Brydie,3,0.3140912
```

Para cumplir este objetivo se le pide que guarde cada enemigo en una estructura compuesta por el nombre (**string**) y la probabilidad de ataque (**float**). Estas estructuras serán acomodadas en **listas**, que a su vez estarán en una tabla Hash donde cada "balde" corresponderá a la categoría de peligro que pertenece el enemigo, como muestra el ejemplo de la figura 1. *NOTA: Personajes sin probabilidad son mostrados con un **-1** en la figura, pero esto es sólo un ejemplo y no necesariamente corresponde a un detalle de implementación*.

El programa debe leer el archivo con los datos de entrada, guardarlos en la estructura definida por la Figura 1 y escribir en un archivo los N nombres más peligrosos. **El único arreglo permitido en la implementación será el utilizado por la tabla de hash.**

1.1. Objetivo del juego

Generar la estructura de datos mostrada en la Figura 1, y guardar en un archivo el nombre de los N nombres más peligrosos. Como refresco de memoria, se adjuntan los criterios:

- Un número mayor de danger_category implica mayor peligrosidad

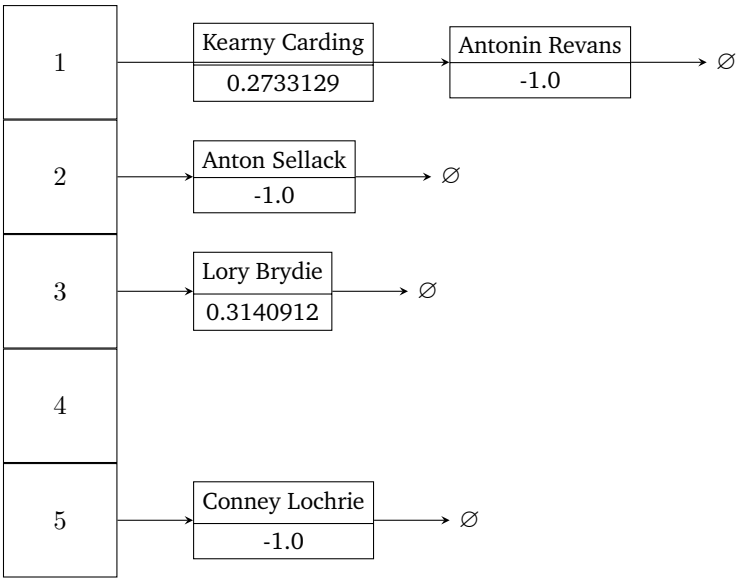

Figura 1: Estructura de datos de los alumnos

- Un número mayor de attack_prob implica mayor peligrosidad
- Si existe un nombre sin attack_prob, entonces su peligrosidad será medida dependiendo de la categoría:
  - Si se encuentra en las categorías 1 o 2, entonces no se considera peligroso, quedando al final de la lista **de su respectiva categoría**.
  - si se encuentra en las categorías 4 o 5, entonces se considera muy peligroso, quedando al principio de la lista **de su respectiva categoría**.
  - si se encuentra en la categorías 3, entonces se considera medianamente peligroso, quedando en la mitad de la lista **de su respectiva categoría**
- Si existen dos nombres que empatan en peligrosidad (danger_category y attack_prob), entonces se ordenarán por orden alfabético.

1.2. Entrada del programa

Su programa recibirá como entrada sólo tres parámetros, correspondientes al nombre del archivo que contiene la lista de nombres, la cantidad de nombres que se quieren priorizar, y el archivo de slaida que contendrà la lista generada.

A continuación un ejemplo de ejecución:

```
TICS311@localhost:~$     ./tarea2 hash.txt 30 hash_salida.txt
```

El ejemplo mostrado más arriba utilizará el archivo *hash.txt* que contiene la lista de nombres, y generará la estrcutura de datos mostrada en el inicio con 30 de esos nombres, los cuales escribirá en el archivo *hash_salida.txt*. El archivo de entrada siempre tendrá **a lo menos** la cantidad de nombres que se piden.

1.3. Salida del programa

El programa debe generar un archivo con los nombres (y sólo los nombres!) más peligrosos de la lista entregada

2. Sobre la entrega

- La tarea debe ser hecha en lenguaje de Programación C.
- Cada grupo puede ser de 2 o 3 personas.
- La tarea se debe entregar el día **Domingo 15 de Mayo** a las 23:59.
- Por cada día de atraso se descontará 1 punto, comenzando a las 00:00 horas del siguiente día. Por ejemplo si entrega la tarea a las 00:00 del siguiente día, la nota máxima que puede obtener es un 6.0
- Para la corrección se utilizará un compilador gcc v 5.1 o superior
- La entrega se realiza por la plataforma Webcursos[1]
- El archivo a entregar debe ser un zip que contenga una carpeta en su interior (y sólo una carpeta) con el nombre **tarea3**. Dentro de esa carpeta debe haber un Makefile[2] y por lo menos un main.c. Además el nombre de su zip debe ser grupoX-tarea3.zip, donde X es el número de su grupo.
- IMPORTANTE: el directorio **tarea3** debe contener el Makefile que generará el archivo ejecutable **tarea3**.

[1] `http://webc.uai.cl`

[2] Este archivo deben generarlo. Información útil pueden encontrarla en `https://stackoverflow.com/questions/1484817/how-do-i-make-a-simple-makefile-for-gcc-on-linux`. Pueden probar su Makefile en un computador con MAC o Linux, o en VSCode en la terminal con UBUNTU. También puede probarse en la shell de `https://repl.it/`