Examen final 2020-09-07

95.11/75.02 - Algoritmos y Programación I - Curso Essaya

Objetivo

Implementar el TDA `tuit`, que representa una publicación de Tuiter (una plataforma sospechosamente similar a Twitter).

Se dispone de los siguientes archivos:

- **Archivos provistos con código:** – `tuit.h`: TDA `tuit` (header)
  - `usuario.h`: declaración adicionales
  - `pruebas.c`: función `main` + pruebas
  - `Makefile`
- **Archivos a completar con código:** – `tuit.c`: TDA `tuit` (implementación)

La idea es implementar en `tuit.c` todas las funciones declaradas en `tuit.h`.

Al compilar con `make`, se genera el archivo ejecutable `tuit_pruebas`. El mismo, al ejecutarlo, efectúa una serie de pruebas para verificar el correcto funcionamiento del TDA `tuit`.

Las pruebas están divididas en 5 ejercicios. Es condición necesaria (pero no suficiente) para aprobar el examen que haya 3 ejercicios OK.

Configuración de los ejercicios

Los ejercicios no están todos habilitados por defecto (para que se pueda compilar el programa sin haber implementado todas las funciones del TDA). Para habilitar la compilación de un ejercicio, por ejemplo el ejercicio 1, descomentar en `pruebas.c` la línea que dice:

```
//#define EJERCICIO1_HABILITADO
```

es decir, que quede de la siguiente manera:

```
#define EJERCICIO1_HABILITADO
```

y volver a compilar el programa.

Salida del programa

Al ejecutar el programa (`./tuit_pruebas`), se ejecutan todos los ejercicios habilitados, y se imprime el resultado de cada uno (OK o FAIL), junto con la cantidad de ejercicios OK. Ejemplo:

```
$ ./tuit_pruebas
ejercicio 1: OK
ejercicio 2: OK
ejercicio 3: OK
Prueba fallida en pruebas.c:148: tuit_cantidad_likes(tuits[i-1]) >= tuit_cantidad_likes(tuits[i])
ejercicio 4: FAIL
ejercicio 5: OK
```

```
Cantidad de ejercicios OK: 4
```

Recordar: es condición necesaria (pero no suficiente) para aprobar el examen que haya al menos 3 ejercicios OK.

Recordar: Correr el programa con `valgrind --leak-check=full` para mayor seguridad de que la implementación es correcta.

El TDA tuit

Un tuit representa una publicación de un mensaje por un usuario, que puede ser ”likeado” y respondido por otros usuarios.

Un tuit tiene:

- Un número identificatorio único (el ID del tuit)
- El número identificatorio del autor (el ID del autor)
- Un mensaje (una cadena de a lo sumo 144 caracteres)
- La lista de todos los IDs de los usuarios que le dieron like (que puede ser de tamaño arbitrariamente grande) * Un usuario no puede dar like a su propio tuit. * Un usuario no puede dar like dos veces al mismo tuit.
- El ID del tuit que responde (o -1 si no es una respuesta)
- La lista de todos los IDs de los tuits que son respuesta de este tuit (que puede ser a lo sumo de tamaño TUIT_MAX_RESPUESTAS = 10).

Descripción de los ejercicios

EJERCICIO 1 (TDA + memoria dinámica) Prueba el funcionamiento básico del TDA. Sin este ejercicio funcionando probablemente no se pueda pasar ninguna de las otras pruebas.

También se prueban las reglas del mecanismo de likes.

EJERCICIO 2 (archivos) Prueba que el tuit pueda ser grabado en un archivo binario, y luego pueda ser recuperado a partir del archivo.

EJERCICIO 3 (ordenamiento) Prueba que podamos ordenar tuits según la cantidad de likes (los más likeados primero).

Sugerencia: para el ordenamiento se puede implementar alguno de los algoritmos que vimos en clase (cualquiera), o bien utilizar la función `qsort` de la librería estándar de C.

EJERCICIO 4 (búsqueda binaria) Prueba que tuit_fue_likeado_por() es eficiente, para lo cual:

- `tuit_dar_like` debe asegurarse de que los IDs de usuarios están ordenados.
- `tuit_fue_likeado_por` debe efectuar una búsqueda binaria.

Sugerencia: `tuit_dar_like` debe ”insertar ordenado”, ¿te hace acordar a algo?

Sugerencia: para la búsqueda binaria, se puede implementar a mano como vimos en clase, o bien utilizar la función `bsearch` de la librería estándar de C.

EJERCICIO 5 (memoria estática) Pueba el mecanismo de respuestas de tuits.

Sugerencia: para el mecanismo de respuestas no es necesario usar memoria dinámica.