FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Informe TP1

Materia: Algoritmos y Programacion 2

Catedra: Wachenchauzer

Ayudante: Fabrizio Graffe
Alumno: Braian Hernán Vicente
Padron: 96542

IMPLEMENTACIÓN:

Para resolver la problemática planteada, se decidió trabajar con el TDA lista únicamente, ya que permite resolver todos los posibles problemas. Esta decisión la tomé porque prefería trabajar con un solo tipo de TDA, a tener que mezclar varios.
Se trabajó con una lista de participantes, que tenía la información necesaria para trabajar. Dicha lista contaba con el nombre del participante, las donaciones realizadas y con información extra que permite saber si se el participante se había tirado el balde o no, y si había o no donado dinero.
Conjunto con la lista, se trabajó con un iterador de la misma que me indicaba quien era el siguiente en la lista de participantes a tirar el balde lo cual me pareció muy conveniente.
Cabe destacar que cada operación posible, se utilizaba memoria dinámica(HEAP) para evitar la pérdida de información por un descuido.

DONAR:

A la hora de almacenar las donaciones se me ocurrio que podria ser una buena idea usar una lista en vez de la pila, lo que era más intuitivo, por la situación explicada anteriormente, que me era más conveniente trabajar con un solo tipo de TDA.

NOMINAR:

A la hora de realizar nominaciones por parte de los participantes, se creaba una lista que contenía el nombre de cada nominado y un estado que me servía para realizar las impresiones por pantalla pedidas. Esto lo decidi hacer de esta manera ya que imprimía los resultados en funciones distintas y esto me permitió hacerlo hacerlo sin problemas.

TIRAR_BALDE:

Para poder realizar la acción de tirar el balde se trabaja, conjuntamente con la lista de los participantes, con un iterador sobre la lista de participantes que siempre que me indicaba quien era el siguiente en la lista a poder decidir sobre tirar el balde.
Hubiera sido posible trabajar con un una cola de participantes, aunque era muy inconveniente ya que tendría que trabajar con 2 TDA distintos, por lo tanto cada vez que se necesite agregar un participante tendría que hacerlo en ambos y encima estaría mal gastando memoria.

ESTRUCTURAS:

Las estructuras creadas son las siguiente:

dato_t
usara para realizar la busqueda de un participante en la lista de participantes

participantes_t
usada para guardar la información de los participantes en la lista de participantes

nominaciones_t
usada para guardar las nominaciones realizadas por un participante

salida_t
usada para guardar la información de la cadena seccionada

PROBLEMÁTICAS:

SECCIONAR CADENA:
Al momento de seccionar la cadena se me ocurrió reutilizar parte del código del cual nos dispuso la catedra para el TP, utilice una parte del código de lectura.c que nos enviaron, para poder seccionar la línea leída y guardarla en memoria dinámica.

VERIFICACIÓN DE CORTE DE PROGRAMA:
Al tratar de verificar el corte del programa tuve un problema al querer comparar
char* entrada = leer_linea();
(entrada[0] != '\0')
lo que supuestamente devuelve el leer_linea() y con eso tendria que terminar el programa. Para solucionarlo verifique que
(strlen(entrada) != 0)
lo que funcionó adecuadamente.

ORDEN DE TIEMPO DE EJECUCIÓN:

O(1):
bool participante_no_esta
void* participante_agregar
void participante_borrar
void donaciones_agregar
void donaciones_destruir
bool nominaciones_imprimir
void nominados_participantes_destruir
void* tirar_balde(lista_iter_t* iter_nominados,char* participante, void* extra);

O(N):
void* participante_buscar
bool donar
void donaciones_imprimir
void* nominados_crear_lista

O(N^2);
void* nominar