Algoritmos y Programación II (75.41) - Cátedra Lic. Gustino Carolo

1° Cuatrimestre 2009

Trabajo Práctico Nº 2: Listas, pilas y colas

Condiciones de entrega

El presente trabajo práctico deberá cumplir con todos los requisitos detallados en el reglamento de la cátedra. La fecha de entrega es el martes 5 de mayo de 2009.

Introducción

Un **sistema de gestión de incidencias** es un software que permite gestionar las incidencias (que pueden ser errores, mejoras o nuevas tareas) de un sistema, permitiendo su control, administración y resolución. Permitie monitorear el avance en su resolución, conocer la persona asignada a la misma a lo largo de la incidencia desde su registro hasta su finalización, registrar comentarios relacionados con la misma, etc.

En este trabajo práctico se pedirá a los alumnos que desarrollen un sistema de gestión de incidencias.

Descripción

Incidencia

Una incidencia puede representar un error en un software, una tarea en un proyecto, etc.

Cada incidencia posee una serie de atributos:

- ID Incidencia
- Tipo: Puede ser: Error, Mejora, Nueva funcionalidad, Tarea
- Estado: Es el estado actual de la incidencia. Puede ser: abierta, en progreso, resuelta, reabierta, cerrada.
- Prioridad: Es la importancia de la incidencia en relación con otras incidencias. Puede ser: bloqueante, critica, mayor, menor, trivial.
- Resolución: Es un registro de la resolución de la incidencia. Puede ser: Corregida, No corregir, Incompleta.
- Reportante: Es la persona que registra la incidencia.
- Responsable: Es la persona a la cual esta asignada actualmente la incidencia.
- Proyecto: Es el proyecto al cual pertenece la incidencia.
- Resumen: Es una breve descripción de la incidencia.
- Descripción: Es la descripción completa de la incidencia.

A su vez, para cada incidencia, además de poder crearla, cambiar su estado y asignar un responsable, es posible registrar comentarios sobre la misma. Un comentario de una incidencia registrará la fecha en la que fue realizado, el usuario que lo realizo y el comentario del mismo.

Al mismo tiempo, se conserva una historia con los cambios de estado de la incidencia, con la fecha en que se produjo y el usuario que la realizó.

Gestor de Incidencias

El gestor de incidencias poseerá la información de todas las incidencias, además de los usuarios que pueden reportar incidencias o resolverlas y los proyectos a los que pueden pertenecer las mismas.

TDA Incidencia (TIN)

Diseño del tipo de dato

La estructura que definirá al TDA Incidencia será la siguiente:

```
typedef struct
{
   char*      comm_date;      /* fecha en la que se realiza el comentario*/
   int        comm_user_id;   /* ID del usuario que realiza el comentario*/
   char*      comm_desc;      /* Comentario*/
} inci_comment_t;

typedef struct
{
   char*      stat_date;      /* Fecha en la que se cambio el estado*/
   int        stat_user_id;   /* Usuario que cambio el estado*/
   stat_inci_t stat_desc;     /* Estado*/
} inci_status_t;

typedef struct {
   int        inci_id;         /* ID de la incidencia */
   tipo_inci_t  inci_type;     /* Tipo de incidencia*/
   prior_inci_t inci_priority; /* Prioridad de la incidencia*/
   int        inci_user;       /* ID del usuario que creo la incidencia*/
   int        inci_assigned_user; /* ID del usuario al que se le asigno */
   char*      inci_date_created;
   int        inci_project_id;
   char*      inci_summary;
   char*      inci_description;
   resol_inci_t inci_resolved;
   char*      inci_date_solved;
   TLista     inci_stat_hist;  /* historial de estados */
   TLista     inci_comm;       /* comentarios de la incidencia */
} TIN;


typedef enum { TIN_BUG, TIN_IMPRO, TIN_TASK, TIN_NEWFUNC} tipo_inci_t;

typedef enum { TIN_FIXED, TIN_DONT_FIX, TIN_INCOMPLETE} resol_inci_t;

typedef enum { TIN_OPEN, TIN_PROGRESS, TIN_RESOLVED, TIN_REOPEN,
TIN_CLOSED} stat_inci_t;

typedef enum { TIN_BLOCKING, TIN_CRITICAL, TIN_MAJOR, TIN_MINOR,
TIN_TRIVIAL} prior_inci_t;
```

Primitivas

Deben desarrollarse las siguientes primitivas para el TDA, respetando exactamente la interfaz y los requerimientos indicados.

int TIN_Crear (TIN* tin, int inci_id, tipo_inci_t inci_type, prior_inci_t inci_priority, int inci_user, int inci_assigned_user, char* inci_date_created, int inci_project_id, char* inci_summary, char* inci_description)		
FUNCIÓN	Crea una incidencia.	
PRE	tin tiene suficiente memoria reservada para una estructura de tipo TIN.	
POST	Si pudo crear la incidencia devuelve 0. Si hubo algún error devuelve -1.	
PARÁMETROS	tin	Manipulador de la incidencia.
	inci_id inci_type inci_priority inci_user inci_assigned_user inci_date_created inci_project_id inci_summary inci_description	

int TIN_Liberar(TIN* tin)		
FUNCIÓN	Libera la incidencia `tin`.	
PRE	`tin` creado.	
POST	Si devuelve 0, estructuras de control eliminadas y `tin` eliminada. Si devuelve -1 ocurrió algún error.	
PARÁMETROS	`Tin`	Manipulador de la incidencia.

int TIN_CambiarEstado(TIN* tin, tFecha fecha, char* usuario, char* estado)		
FUNCIÓN	Registra un cambio de estado para la incidencia.	
PRE	tin creado.	
POST	Si devuelve 0, estado actualizado. Si devuelve -1 ocurrió algún error.	
PARÁMETROS	tin	Manipulador de la incidencia.
	fecha	Fecha en la que se realiza el cambio de estado

int TIN_CambiarEstado(TIN* tin, tFecha fecha, char* usuario, char* estado)		
	Usuario	Usuario que registra el cambio de estado
	estado	Estado

int TIN_Resolver(TIN* tin, tFecha fecha, char* usuario, resol_inci_t resolucion)		
FUNCIÓN	Registra la resolucion de la incidencia, y pasa el estado de la misma a Resuelta (TIN_RESOLVED).	
PRE	tin creado.	
POST	Si devuelve 0, incidencia resuelta. Si devuelve -1 ocurrió algún error.	
PARÁMETROS	tin	Manipulador de la incidencia.
	fecha	Fecha en la que se resolvio la incidencia
	Usuario	Usuario que registra la resolucion
	resolucion	Tipo de resolucion

int TIN_Comentar(TIN* tin, tFecha fecha, char* usuario, char* comentario)		
FUNCIÓN	Registra un comentario para la incidencia.	
PRE	tin abierto.	
POST	Comentario registrado. Devuelve -1 si ocurrió algún error.	
PARÁMETROS	tin	Manipulador de la incidencia.
	fecha	Fecha en la que se realiza el comentario
	Usuario	Usuario que registra el comentario
	Comentario	Comentario de la incidencia.

int TIN_GetDatos (TIN* tin, int* inci_id, tipo_inci_t* inci_type, prior_inci_t* inci_priority, int* inci_user, int* inci_assigned_user, char* inci_date_created, int* inci_project_id, char* inci_summary, char* inci_description)		
FUNCIÓN	Obtiene los datos de una incidencia.	
PRE		
POST	Si pudo obtener los datos devuelve 0. Si hubo algún error devuelve -1.	
PARÁMETROS	tin	Manipulador de la incidencia.

	`inci_id`
	`inci_type`
	`inci_priority`
	`inci_user`
	`inci_assigned_user`
	`inci_date_created`
	`inci_project_id`
	`inci_summary`
	`inci_description`	

int TIN_GetListaComentarios(const TIN* tin, TListaSimple* comentarios)		
FUNCIÓN	Lee el comentario, usuario y la fecha de todos los comentarios de la incidencia.	
PRE	`tin` abierto. `Comentarios` apunta a una lista simple creada y vacía para contener objetos de tipo `inci_comment_t`	
POST	Si devuelve 0 devolvió en `comentarios` una lista de objetos `inci_comment_t` con los datos de cada uno de los comentarios de la incidencia. La memoria reservada para los strings debe ser liberada por el invocador. Si devuelve -1 ocurrió un error y la lista permanece inalterada.	
PARÁMETROS	`tin`	Manipulador de la incidencia.
	`comentarios`	Lista donde se guardará la información de los comentarios.

int TIN_GetListaEstados(const TIN* tin, TListaSimple* estados)		
FUNCIÓN	Lee el estado, usuario y la fecha de todos los cambios de estado de la incidencia.	
PRE	`tin` abierto. `Estados` apunta a una lista simple creada y vacía para contener objetos de tipo `inci_status_t`.	
POST	Si devuelve 0 devolvió en `estados` una lista de objetos `inci_status_t` con los datos de cada uno de los estados de la incidencia. La memoria reservada para los strings debe ser liberada por el invocador. Si devuelve -1 ocurrió un error y la lista permanece inalterada.	
PARÁMETROS	`tin`	Manipulador de la incidencia.
	`estados`	Lista donde se guardará la información de los estados.

TDA Gestor Incidencias (TGI)

Diseño del tipo de dato

La estructura que definirá al TDA Incidencia será la siguiente:

```
typedef struct user_t
{
     int   user_id;
     char* user_name;
} user_t;

typedef struct project_t
{
     int   project_id;
     char* project_name;
} project_t;

typedef struct {
    TLista incidencias;
    TLista proyectos;
    TLista usuarios;
} TGI;
```

Primitivas

int TGI_Crear (TGI* tgi, const char* proyectos, const char* usuarios)		
FUNCIÓN	Crea un gestor de incidencias.	
PRE	tgi tiene suficiente memoria reservada para una estructura de tipo TGI. Proyectos es la ruta al archivo de proyectos, alocada y terminada en \0. Usuarios es la ruta al archivo de proyectos, alocada y terminada en \0.	
POST	Si el archivo existe y está versionado, lo abre y devuelve 0. Si el archivo existe y no está versionado, inicializa la funcionalidad de versionado y devuelve 1. Si hubo algún error devuelve -1.	
PARÁMETROS	tgi	Manipulador del gestor de incidencias.
	ruta	Ruta al archivo a abrir.

int TGI_Cerrar(TGI* tgi)		
FUNCIÓN	Cierra el gestor `tgi`.	
PRE	`tgi` creado.	
POST	Si devuelve 0, gestor cerrado. Si devuelve -1 ocurrió algún error.	
PARÁMETROS	`tgi`	Manipulador del gestor de incidencias.

int TIN_AgregarIncidencia (const TGI* tgi, tipo_inci_t inci_type, prior_inci_t inci_priority, int inci_user, int inci_assigned_user, char* inci_date_created, int inci_project_id, char* inci_summary, char* inci_description)		
FUNCIÓN	Agrega una incidencia en el gestor de incidencias.	
PRE	`tgi` creado.	
POST	Devuelve el ID de la incidencia creada si no hubo error. Si devuelve -1 no existe el ID del proyecto. Si devuelve –2 no existe el id del usuario. Si devuelve –3 ocurrió otro error.	
PARÁMETROS	`tgi`	El manipulador del gestor de incidencias.

int TIN_GetIncidencia(const TGI* tgi, int inci, TIN* tin)		
FUNCIÓN	Recupera los datos de una incidencia del gestor de incidencias.	
PRE	`tgi` creado. `tin` tiene memoria reservada para una estructura de tipo `TIN`.	
POST	Si devuelve 0 devolvió en `tin` la incidencia. La memoria de los strings debe ser liberada por el invocador. Si devuelve -1 no existe la incidencia `inci`. Si devuelve -2 ocurrió otro error.	
PARÁMETROS	`tgi`	El manipulador del gestor de incidencias.
	`inci`	El Id de la incidencia que se desea consultar.
	`tin`	Estructura donde se guardará la información de la incidencia.

int TIN_GetIncidenciasUsuario(const TGI* tgi, int usuario, Tcola* incidencias)		
FUNCIÓN	Lee las incidencias asignadas a un usuario.	
PRE	`tgi` creado. `incidencias` apunta a una cola creada y vacía para contener objetos de tipo `TIN`.	
POST	Si devuelve 0 devolvió en `incidencias` las incidencias del usuario. La memoria de los strings debe ser liberada por el invocador. Si devuelve -1 no existen incidencias para el usuario. Si devuelve -2 ocurrió otro error.	
PARÁMETROS	`tgi`	El manipulador del gestor de incidencias.
	`usuario`	El ID del usuario para quien se desean consultar las incidencias.
	`incidencias`	Cola donde se guardará la información de las incidencias.

int TIN_GetIncidenciasProyecto(const TGI* tgi, int proyecto, TListaSimple* incidencias)	
FUNCIÓN	Lee las incidencias pertenecientes a un proyecto.

int TIN_GetIncidenciasProyecto(const TGI* tgi, int proyecto, TListaSimple* incidencias)		
PRE	`tgi` creado. `incidencias` apunta a una cola creada y vacía para contener objetos de tipo `TIN`.	
POST	Si devuelve 0 devolvió en `incidencias` las incidencias del usuario. La memoria de los strings debe ser liberada por el invocador. Si devuelve -1 no existen incidencias para el usuario. Si devuelve -2 ocurrió otro error.	
PARÁMETROS	`tgi`	El manipulador del gestor de incidencias.
	`proyecto`	El ID del proyecto para quien se desean consultar las incidencias.
	`incidencias`	Lista donde se guardará la información de las incidencias.

Programa de aplicación

Se debe desarrollar un programa que, utilizando el **TDA Gestor Incidencias** definido más arriba, proponga las siguientes opciones al usuario:

- Agregar una incidencia
- Cambiar el estado de una incidencia
- Registrar la resolución de una incidencia
- Obtener los datos de una incidencia
- Obtener el historial de estados de una incidencia
- Obtener los comentarios de una incidencia
- Obtener las incidencias de un usuario
- Obtener las incidencias de un proyecto
- Agregar un proyecto
- Agregar un usuario
- Registrar la información del gestor de incidencias en archivos de proyectos, usuarios e incidencias

El programa podrá ser invocado con las siguientes opciones

- Inicializar la funcionalidad del gestor de incidencias vacio
    - En este caso el programa se invocará como:
      `tp2`

- Inicializar la funcionalidad del gestor de incidencias con datos.
    - En este caso el programa se invocará como:
      `tp2 <nombre_archivo_proyectos> <nombre_archivo_usuarios> <nombre_archivo_incidencias>`

      <*nombre_archivo_proyectos*> es la ruta al archivo con los datos de los proyectos

      <*nombre_archivo_usuarios*> es la ruta al archivo con los datos de los usuarios

      <*nombre_archivo_incidencias*> es la ruta al archivo con los datos de las incidencias

Observaciones

- El programa debe informar al usuario si la ejecución fue realizada con éxito o no. **Los mensajes de error deben ser claros y brindar la mayor cantidad de información posible al usuario.** Esto incluye, pero no se limita a, errores de sintaxis, errores originados en el manejo de archivos, y errores al invocar a las primitivas.
- Los mensajes de error deben emitirse por STANDARD ERROR (stderr).
- Los mensajes informativos deben emitirse por STANDARD OUTPUT (stdout).
- El programa NO debe solicitar **ningún** dato al usuario durante su ejecución. Esto es, el programa nunca debe quedar bloqueado esperando una respuesta del usuario (y esto obviamente incluye solicitarle al usuario una tecla al finalizar el programa).
- Antes de procesar un archivo, el mismo debe guardarse en memoria utilizando el tipo de estructura que mejor se adapte al problema.
- Para leer y escribir archivos con formato, es **obligatorio** utilizar las funciones *fprintf* y *fscanf*. No se permite realizar el parseo de las líneas manualmente.