SSL 2005.1 – TP Simulación de un Año Académico de SSL – Entrega #2 – TAD MesaDeExamenFinal.

1.0.1.20050516

por José María Sola y Jorge Muchnik.

Este TAD puede ser extendido más adelante para cubrir las necesidades globales del TP.

Requisitos

- Haber aprobado TP #0.
- Haber leído y comprendido el documento anterior: "**SSL 2005.1 – Presentación General del Trabajo Práctico – Simulación de un Año Académico de SSL**"

Objetivos

- **Especificar** el TAD **MesaDeExamenFinal**.
- **Implementar** el TAD **MesaDeExamenFinal** en una **Biblioteca**.
- **Probar** la implementación del TAD **MesaDeExamenFinal** mediante una **aplicación de prueba**.
- **Formalizar** aún más la especificación de las operaciones mediante los conceptos de **precondición** y **poscondición**.
- Aplicar **parámetros out** y/o **inout** para implementar subprogramas con más de un resultado.
- Aplicar **números pseudo aleatorios** de ANSI C.
- Aplicar **archivos de texto** e introducir el concepto de **stream** (**flujo de datos**) de ANSI C.
- Implementar **arreglos con tamaño definido dinámicamente** (en tiempo de ejecución) en ANSI C.
- **Capturar** las **salidas** de los **procesos de traducción** y de la **aplicación de prueba**.
- **Continuar** con el análisis y el diseño del **TAD Calendario** y del programa de **Simulación**.

Sobre el TAD MesaDeExamenFinal

El TAD **MesaDeExamenFinal** representa un conjunto de **alumnos** que rinden **examen final** de SSL en un determinado **día**, con una mesa examinadora de tres **profesores**. Por cada alumno de ese conjunto se deberá conocer si **asistió o no** al examen y en el caso de haber asistido, se deberá conocer su **calificación**.

Los alumnos se identifican y representan por su **legajo**, un número entero perteneciente al intervalo abierto (40.000, 200.000). Las calificaciones son números naturales entre 1 y 10. Los tres profesores son el **presidente**, el **primer vocal** y el **segundo vocal**; representados por sus nombres y apellidos, de longitud variable.

Un valor del TAD MesaDeExamenFinal está definido por la 9-upla:
(Día, Mes, Presidente, Primer vocal, Segundo vocal, Cantidad de inscriptos, Inscriptos, Ausentes, Calificaciones).

Los conjuntos **Inscriptos** y **Ausentes** contienen elementos del tipo **Legajo.** La cardinalidad del conjunto **Inscriptos** se establece en la operación de **Creación**. El conjunto **Calificaciones** contiene pares ordenados del tipo **(Legajo, Calificación)**.

La operación **Creación** produce un valor del TAD MesaDeExamenFinal con los tres conjuntos **Inscriptos**, **Ausentes** y **Calificaciones** vacíos.

La operación **Inscripción de un alumno** se aplica a un valor del TAD MesaDeExamenFinal. Esta operación produce un valor del TAD idéntico al valor del argumento de la operación pero con la cardinalidad del conjunto **Inscriptos** incrementada en uno.

La *precondición* para la aplicación de la operación **Tomar asistencia** sobre un valor del TAD, es que el conjunto **Inscriptos** del valor dato esté completado. *Poscondición*: el conjunto **Ausentes** debe estar incluido en o ser igual al conjunto **Inscriptos**.

La *precondición* para la aplicación de la operación **Evaluación de alumnos** a un valor del TAD, es que ya se haya aplicado la operación **Tomar asistencia** y que por lo menos haya un alumno presente. *Poscondición*: el conjunto **Calificaciones** contiene los legajos y sus correspondientes calificaciones de todos los alumnos presentes.

Operaciones

Creación

- **Creación**. Dados un Natural (dia), una Cadena de tres caracteres (mes), los nombres y apellidos del presidente, primer vocal y segundo vocal (tres cadenas de longitud variable), y la cantidad de inscriptos, produce un valor del TAD MesaDeExamenFinal, con los conjuntos Inscriptos, Ausentes y Calificaciones vacíos. Utilizar reserva dinámica de memoria en la implementación de los nombres y apellidos de los presidentes y para el conjunto de alumnos.

Consulta

- **Presencia** (predicado). Dados una MesaDeExamenFinal y un legajo, retorna si el alumno estuvo presente o no.
- **Calificación**. Dados una MesaDeExamenFinal y un legajo, retorna la calificación del alumno ó cero si estuvo ausente.
- **Aprobación** (predicado). Dados una MesaDeExamenFinal y un legajo, retorna si fue aprobado o no.

- **Desaprobación** (predicado). Dados una MesaDeExamenFinal y un legajo, retorna si fue desaprobado o no.

Acceso

- **Dia**. Dada una MesaDeExamenFinal retorna un Natural.
- **Mes**.
  1. *NumeroMes*, dada una MesaDeExamenFinal retorna un Natural.
  2. *MesCorto*, dada una MesaDeExamenFinal retorna una cadena.
  3. *MesLargo*, dada una MesaDeExamenFinal retorna una cadena.
- **Profesores**. Dada una MesaDeExamenFinal retorna una cadena con el nombre y apellido del profesor
  1. *Presidente*.
  2. *Primer Vocal*.
  3. *Segundo Vocal*.
- **Cantidad de inscriptos**. Dada una MesaDeExamenFinal retorna la cantidad de inscriptos.

Modificación

- **Inscripción de un alumno**. Dados un legajo y una MesaDeExamenFinal, retorna una MesaDeExamenFinal idéntica a la dato pero con el conjunto de Inscriptos modificado.
- **Tomar Asistencia**. Dada una MesaDeExamenFinal, retorna una nueva MesaDeExamenFinal con el conjunto Ausentes modificado. *Ver operación privada GenerarAsistencia*.
- **Evaluación de alumnos**. Dada una MesaDeExamenFinal, retorna una nueva MesaDeExamenFinal con su conjunto de Calificaciones con los resultados de las evaluaciones de cada uno de los presentes. *Ver operación privada GenerarCalificacion.*

Salidas por streams de texto

Estas operaciones reciben como dato una MesaDeExamenFinal y un stream de texto abierto para escritura. Las operaciones escriben líneas de texto en sus streams datos, y retornan el mismo stream pero avanzado en función a las líneas de texto escritas. Para la aplicación de prueba, el stream dato será `stdout`.

- **Acta**. Escribe un encabezado con datos sobre la Mesa de Examen Final, un cuerpo con el detalle de las evaluaciones de los alumnos (si estuvo ausente o su calificación) y un pie con los totales.
  Encabezado. 4 líneas.
  ```
  "Examen Final SSL - Fecha %d %s\n"
  "Presidente   : %s\t
  "Primer Vocal : %s\tSegundo Vocal: %s\n"
  "Legajo\tCalificación\n"
  ```
  Cuerpo. 3 tipos de líneas posibles. Legajo seguido de "Ausente", ó seguido de una nota, seguido de "Aprobado" ó "Desaprobado"
  ```
  "%s\tAusente\n"
  ```
  ó
  ```
  "%s\t%d\tAprobado\n"
  ```
  ó
  ```
  "%s\t%d\tDesaprobado\n"
  ```
  Pie. 2 líneas.
  ```
  "Inscriptos: %d\tAusentes: %d\tExamindos: %d\n"
  "Aprobados: %d\tDesaprobados: %d\n"
  ```
- **Ausentes**. Una línea con el legajo por cada alumno ausente.
- **Aprobados**. Una línea con el legajo y nota por cada alumno aprobado.
- **Desaprobados**. Una línea con el legajo y nota por cada alumno desaprobado.

Privadas a la Implementación

Las siguientes operaciones no forman parte de la especificación del TAD. Son funciones necesarias para una mejor estructuración y diseño de la implementación y producen resultados estadísticos necesarios para la simulación. Debe ser declaradas con el especificador `static` para encapsularlas dentro de la biblioteca. Estudiar la generación de números pseudo aleatorios en ANSI C.

- `static int GenerarAsistencia(void)`. El 60% de las veces que es invocada, retorna el `int 1`, el resto, el `int 0`.
- `static int GenerarCalificacion(void)`. El 40% de las veces que es invocada, retorna un `int` entre `1` y `3`, el resto, un `int` entre `4` y `10`.

Específicas de la implementación

- **Destrucción**. Dada una variable del TAD MesaDeExamenFinal, libera los recursos previamente asignados.

Restricciones y Guías para la Implementación

Los conjuntos **Inscriptos**, **Ausentes** y **Calificaciones** se implementarán en un único arreglo de `structs`. Cada elemento del arreglo contendrá dos campos: legajo y un campo que indica si estuvo presente ó no y, de corresponder, su nota. El día y el mes se representaran en un solo campo `int`.

```
typedef struct {
    char legajo[6+1];
    char nota; /* 0      ==> ausente;
                * 1 a 10 ==> presente */
} Evaluacion;

typedef struct {
    int          fecha; /* día y mes */
    const char*  presidente;
    const char*  primerVocal;
    const char*  segundoVocal;
    int          inscriptos;
    Evaluacion*  evaluaciones;
                 /* puntero al primer elemento de
                  * un arreglo de evaluaciones */
} MesaDeExamenFinal;
```

```
static const char* meses[][2] = {
   {"Ene", "Enero"     },
   {"Feb", "Febrero"   },
   {"Mar", "Marzo"     },
   {"Abr", "Abril"     },
   {"May", "Mayo"      },
   {"Jun", "Junio"     },
   {"Jul", "Julio"     },
   {"Ago", "Agosto"    },
   {"Sep", "Septiembre"},
   {"Oct", "Octubre"   },
   {"Nov", "Noviembre" },
   {"Dic", "Diciembre" }
};
```

Identificadores para las Funciones que Implementan las Operaciones y guías para algunos Prototipos

1. `MesaDeExamenFinal MesaDeExamenFinal_Crear(`
   `int unDia,`
   `const char* unMes,`
   `const char* unPresidente,`
   `const char* unPrimerVocal,`
   `const char* unSegundoVocal,`
   `int unaCantidadDeInscriptos`
   `);`
2. `void MesaDeExamenFinal_InscribirAlumno(`
   `MesaDeExamenFinal* unaMesaDeExamenFinal, /* inout */`
   `const char* unLegajo`
   `);`
3. `void MesaDeExamenFinal_TomarAsistencia(`
   `MesaDeExamenFinal* unaMesaDeExamenFinal /* inout */`
   `);`
4. `void MesaDeExamenFinal_Evaluar(`
   `MesaDeExamenFinal* unaMesaDeExamenFinal /* inout */`
   `);`
5. `int MesaDeExamenFinal_IsPresente(`
   `const MesaDeExamenFinal* unaMesaDeExamenFinal, /* in */`
   `const char* unLegajo`
   `);`
6. `MesaDeExamenFinal_GetCalificacion`
7. `MesaDeExamenFinal_IsAprobado`
8. `MesaDeExamenFinal_IsDesaprobado`
9. `void MesaDeExamenFinal_EmitirActaTexto(`
   `const MesaDeExamenFinal* unaMesaDeExamenFinal, /* in */`
   `FILE* unStream /* inout */`
   `);`
10. `MesaDeExamenFinal_EmitirAusentesTexto`
11. `MesaDeExamenFinal_EmitirAprobadosTexto`
12. `MesaDeExamenFinal_EmitirDesaprobadosTexto`
13. `int MesaDeExamenFinal_GetDia(`
    `const MesaDeExamenFinal* unaMesaDeExamenFinal /* in */`
    `);`
14. `MesaDeExamenFinal_GetNumeroMes`
15. `MesaDeExamenFinal_GetMesCorto`
16. `MesaDeExamenFinal_GetMesLargo`
17. `const char* MesaDeExamenFinal_GetPresidente(`
    `const MesaDeExamenFinal* unaMesaDeExamenFinal /* in */`
    `);`
18. `MesaDeExamenFinal_GetPrimerVocal`
19. `MesaDeExamenFinal_GetSegundoVocal`
20. `MesaDeExamenFinal_GetCantidadDeInscriptos`
21. `void MesaDeExamenFinal_Destruir(`
    `const MesaDeExamenFinal* unaMesaDeExamenFinal /* in */`
    `);`

`/* Operaciones privadas */`

22. `static int GenerarAsistencia(void);`
23. `static int GenerarCalificacion(void);`

Conceptos de ANSI C Necesarios

- Los necesarios para la entrega anterior.
- Archivos de texto
- Salida y Entrada Standard de ANSI C
- Streams (flujos de datos)
- Arreglos alocados dinámicamente.
- Aplicación de parámetros in, out e inout en ANSI C.
- Números Pseudoaleatorios en ANSI C
- Encapsulamiento de funciones con `static`.

Sobre la Prueba

Se debe diseñar una aplicación de prueba con datos constantes, que verifique la corrección de todos los casos posibles. Utilice el concepto de partición de conjuntos, y pruebe los casos extremos y un caso representativo de cada partición. Para las pruebas de las operaciones de streams, utilizar `stdout` como argumento.

Otras Restricciones

- Para la implementación debe utilizarse el lenguaje y los elementos de la biblioteca estándar especificados en ANSI C [M3].
- Luego de que la biblioteca y la aplicación de prueba hayan sido construidas con la herramienta de desarrollo elegida por el equipo, deben ser verificadas mediante la herramienta de desarrollo “Borland C++ Compiler 5.5 with Command Line Tools” (BCC32) configurada tal como dicta la cátedra [TP #0].
- Los procesos de compilación con BCC32 no deben emitir warnings (mensajes de advertencia) ni, por supuesto, mensajes de error.
- Los identificadores de las funciones que implementan las operaciones, de las variables, tipos y demás elementos así como los nombres de los archivos deben ser los indicados a lo largo de este enunciado.
- Cualquier decisión que el equipo tome sobre algún punto no aclarado en el enunciado debe ser agregada como hipótesis de trabajo.

Sobre La Entrega

Tiempo

- Siete (7) días después de analizado el enunciado en clase.

Forma

El trabajo debe presentarse en hojas **A4 abrochadas** en la esquina superior izquierda. En el encabezado de cada hoja debe figurar el **título** del trabajo, el título de **entrega**, el código de **curso** y los **apellidos** de los integrantes del equipo. Las hojas deben estar enumeradas en el pie de las mismas con el formato "**Hoja n de m**".

El código fuente de cada componente del TP debe comenzar con un comentario encabezado, con todos los datos del equipo de trabajo: **curso**; **legajo**, **apellido** y **nombre** de cada integrante del equipo y **fecha de última modificación**. La fuente a utilizar en la impresión debe ser una fuente de ancho fijo (e.g. `Courier New`, `Lucida Console`).

1. TAD MesaExamenFinal

1.1. **Especificación**. Especificación completa, extensa y sin ambigüedades de los valores y de las operaciones del TAD.
1.2. **Implementación**
1.2.2. Biblioteca que implementa el TAD
1.2.2.1. Listado de código fuente del archivo encabezado, **parte pública**, **MesaDeExamenFinal.h.**
1.2.2.2. Listado de código fuente del la definición de la Biblioteca, **parte privada**, **MesaDeExamenFinal.c.**
1.2.3. **Salidas**. Captura impresa de la salida del proceso de traducción (BCC32 y TLIB). Utilizar fuente de ancho fijo.

2. Aplicación de Prueba

2.1. **Código Fuente**. Listado del código fuente de la aplicación de prueba, **MesaDeExamenFinalAplicacion.c.**
2.2. **Salidas**
2.2.1. Captura impresa de la salida del proceso de traducción (BCC32). Utilizar fuente de ancho fijo.
2.2.2. Captura impresa de las salidas de la aplicación de prueba. Utilizar fuente de ancho fijo.

3. Copia Digitalizada

CD ó disquette con copia de solamente los 3 archivos de código fuente (MesaDeExamenFinal.h, MesaDeExamenFinal.c y MesaDeExamenFinal Aplicacion.c). No se debe entregar ningún otro archivo.

4. Formulario de Seguimiento de Equipo.