Programación Aplicada y Lab.
Proyecto Final
SISTEMA INTEGRAL DE INFORMACIÓN CLÍNICA

Maestro Jorge Rodríguez

I. Descripción del Sistema

El Sistema Integral de Información Clínica pretende ser un sistema computarizado gráfico que permita simplificar la manipulación de información en un consultorio médico. Se desea un sistema por medio del cual se pueda almacenar y recuperar, de manera simple y confiable, información general sobre los doctores que laboran en el consultorio, sobre los pacientes del mismo, así como su historia clínica.

II. Alcances y Limitaciones

El diseño de la interfaz gráfica de usuario queda a consideración de cada equipo de desarrollo. La funcionalidad y estética de la interfaz es importante y tiene peso en la calificación.
A su vez, el programa deberá tener el máximo de validación posible en función de lo que se ha visto en el curso. Las pruebas que harán los sinodales de cada proyecto serán destructivas.
El sistema estará conformado por tres módulos, a través de los cuales deberá administrarse la información y actividades correspondientes.
Además deberá de contemplar los siguientes aspectos:

- Debe ser simple de usar.
- El límite de la información será limitado por la capacidad del dispositivo de almacenamiento masivo.
- Las diferentes bases de datos deben poder comunicarse entre sí.
- Las interfaces de usuario serán en modo de gráfico.
- Las interfaces de usuario deberán manipularse por medio de estructuras de menús.

III. Requisitos Funcionales

El programa debe manipular la información de tres grandes elementos:

1. Doctores
2. Pacientes
3. Historia Clínica

1. Doctores.

A continuación se enlista la información que se debe manipular para cada doctor registrado en el sistema:

- Nombre
- Especialidad 1
- Especialidad 2
- Estatus (activo / inactivo)
- Teléfono para urgencias
- Dirección particular
- Teléfono
- Consultorio asignado
- Días de consulta
- Horario de consulta

2. Pacientes.

A continuación se enlista la información general que el sistema debe manipular de cada paciente:

- Nombre
- Dirección
- Teléfono
- Sexo
- Fecha de nacimiento (dd/mm/aaaa)
- Edad (campo calculado)
- Estatura
- Alergias
- Tipo de sangre
- Padecimientos crónicos

3. Historia Clínica.

La historia clínica se conforma de información del paciente más información del doctor que atendió al paciente, más la información relacionada al padecimiento tratado, tal como se enlista a continuación:

- Nombre del paciente
- Nombre del doctor tratante
- Fecha de cita (campo automático de S.O.)
- Diagnóstico
- Tratamiento
- Anotaciones

Se sugiere que cada elemento de información sea manipulado en un módulo independiente. Por la naturaleza del sistema, no podrán ser borrados elementos, es decir, no será posible dar de baja doctores, pacientes, ni eliminar registros de la historia clínica de los pacientes. Para los tres elementos será posible hacer los siguientes movimientos: altas, consultas y modificaciones.
Las bases de datos deben de ser manipuladas a través de

estructuras dinámicas. Deben de existir dos lista dinámicas independientes, es decir, la que corresponde a los doctores y la que corresponde a los pacientes. La lista dinámica que corresponde a la historia clínica de cada paciente debe de estar "colgada" de los datos generales del paciente correspondiente.

Al iniciar, el programa primero creará las listas dinámicas a partir de la información que está almacenada en los archivos de base de datos.

Deberá contar con una opción en el menú 'Acerca de…' que muestre el nombre de los desarrolladores del sistema y la fecha de desarrollo. Debe contar con un menú que permita seleccionar el tipo de información que se desea trabajar, es decir, Doctor, Paciente o Historia Clínica. A su vez, para cada opción que se elija del menú principal, se deberá pasar al menú específico, que deberá contemplar las opciones que sean necesarias (altas, modificaciones, consultas).

Las siguientes deben ser consultas incluidas en el área que el equipo decida (reportes o en doctores/pacientes):

Comportamiento
Despliega un listado de todos los doctores registrados, desplegando nombre del doctor, especialidad 1, especialidad 2 y teléfono de emergencia, ordenada alfabéticamente por nombre del doctor
Muestra la información general del doctor especificado
Despliega una lista con el nombre de todos los pacientes, ordenada alfabéticamente por nombre, mostrando nombre y teléfono
Muestra la información general del paciente especificado
Despliega la historia clínica completa del paciente, mostrando un registro por pantalla
Muestra una lista de todos los doctores que cuentan con la especialidad indicada, mostrando nombre del doctor y teléfono de emergencia
Ayuda general del sistema

IV. Estructura de archivos

La información que se vaya generando con el uso del sistema debe quedar debidamente registrada en archivos. Las siguientes indicaciones son obligatorias:

1) La información de los doctores debe de guardarse en un archivo de texto. Nombre al archivo como desee.

2) La información general de los pacientes debe de guardarse en un archivo de texto. Nombre al archivo como desee.

3) La historia clínica de cada paciente debe de guardarse en un archivo de texto; el nombre del archivo deberá de ser el nombre del paciente, sustituyendo espacios por guiones de subrayado (_).

V. Restricciones de Programación

El proyecto final tiene como objetivo que los alumnos del curso apliquen de manera práctica todos los conocimientos que fueron transmitidos a lo largo del semestre, así como las "mejores prácticas" de la programación. Por lo anterior, se deberán considerar los siguientes elementos:

1) Uso del lenguaje de programación:

- Uso de funciones con argumentos.
- Uso de archivos de texto.
- Uso de módulos y automatización de la compilación.
- Estructuras.
- Estructuras dinámicas.
- Diferenciación del modelo Modelo-Vista-Controlador,

Nota: todos estos elementos serán evaluados.

2) Uso de buenas prácticas de programación:

- Identificadores significativos (es decir, nombres de variables y funciones deben referirse a la utilidad o uso de los mismos).
- Alineación de código para garantizar la legibilidad del código fuente (debe ser la alineación que se genera de manera automática por medio de emacs).
- Documentación del código fuente, es decir, cada archivo de código fuente debe contener todos los comentarios que sean necesarios para garantizar que cualquier otro programador competente pueda comprender el código. En si, se recomienda que los comentarios se usen para explicar aspectos no obvios del código, o que no se comprendan a primera vista.
- Para mayor referencia, consultar el documento anexo a esta especificación, en el que se explica una técnica formal de documentación de código fuente, misma a la que deberá de apegarse el proyecto.

VI. Documentación

Se deberá entregar la documentación por escrito con el análisis, la definición y la planeación (diagramas a bloques y pseudocódigo) del problema a resolver, además el manual de usuario, debiendo considerar los siguientes puntos:

1) La documentación deberá presentarse usando el formato de artículos del *Institute of Electrical and Electronics Engineers* (IEEE).
2) Cada error ortográfico será penalizado con 0.5 puntos menos en la calificación de la documentación. Nota: Exceptuando el código fuente, todo el texto escrito en el documento, incluyendo el pseudocódigo y las palabras escritas con letras mayúsculas (por ejemplo, en títulos), deberá acentuarse correctamente.

VII. Fecha de Entrega

Consulte la fecha de entrega, así como el horario, en el sistema.

VIII. DOCUMENTACIÓN DEL PROYECTO

Documentar código es un elemento importante para dar mayor claridad a los algoritmos y a la forma como fue codificado un problema.

La documentación se hará por medio de un formato especial de los comentarios. En este caso se seguirá la siguiente convención:

Comentarios a nivel del archivo:
Estos comentarios proveen una descripción general del problema y elementos generales como son el autor y fecha de creación y modificación.
Se deben colocar en la parte superior, antes de cualquier línea de código (arriba de las directivas #include ..)

```
/**
* @file arreglos.c
*
* @brief Este programa permite capturar dos arreglos de tipo entero
* para despues sumarlos en un tercer arreglo, y obtener el promedio
* de la suma de valores del tercer arreglo, imprimiendo la salida en
* tipo flotante.
*
* @author Pablo Segovia
* @date 12/03/2010
*/
```

@file debe declarar el nombre del archivo
@brief una descripción de lo que hace el programa. La descripción debe estar en un párrafo y terminar con "."
@author debe especificar el nombre del o los autores del código
@date es la última referencia con la fecha en que se creó el archivo

Comentarios a nivel de función:
Arriba de la implementación de cada función (no donde se declara el prototipo ni donde se invoca), se debe documentar cada función, de acuerdo al autor, parámetros, fecha y tipo de retorno. Por ejemplo:

```
/**
* Esta función recibe un caracter en el primer argumento y si
* este es una letra minúscula lo convierte a mayúscula y regresa
* el nuevo valor en el segundo argumento.
* Regresa un 1 si pudo convertir a mayúscula el caracter
* y 0 si no pudo hacerlo.
* @author Iggy Pop
* @param chrData        El caracter a convertir (minúscula a mayúscula)
* @param *may           El caracter convertido en mayúscula
* @return int
*
*/
int carMinAMay(char chrData, char *may)
{
   …….
}
```

@param Nombre_Variable – Descripción del argumento.
@return - Identifica el tipo de valor que es regresado por la función.

Nota: Si hubiera algún algoritmo importante dentro de una función, se podrán incluir comentarios antes de la implementación de dicho comentario.