**Pontificia Universidad Católica de Valparaíso**

**Facultad de Ingeniería**

**Escuela de Ingeniería Informática**

**Organización y Administración de Hospitales Privados**

**Alberto Cabrera, Jorge Sepúlveda, Víctor Muñoz, Juan Moreno**

**Carrera: Ingeniería Ejecución en Informática**

**25-10-2013**

**Índice**

**1 Resumen ............................................................................................................ 3**

**2 Introducción....................................................................................................... 4**

**3 Descripción Del Problema................................................................................. 5**

**4 Análisis Del Problema........................................................................................ 6**

**4.1 Estructura Hospital................................................................................ 7**

**4.2 Estructura Doctores................................................................................ 8**

**4.3 Estructura Tratamientos........................................................................ 9**

**4.4 Estructura Paciente…………………………………………………… 10**

**4.5 Estructura Persona…………………………………………………… 11**

**4.6 Estructura Costos…………………………………………………….. 12**

**4.7 Diagrama General……………………………………………………. 13**

**5 Manual Programador………………………………………………………... 14 5.1 Documentación Interna……………………………………………… 14**

**5.2 Documentación Externa……………………………………………... 14**

**6 Planificación Grupal………………………………………………………… 17**

**6.1 Planificación Individual…………………………………………….... 18**

**7 Manual del Operador………………………………………………………… 20**

**7.1 Manejo de la Consola………………………………………………… 20**

**8 Manual del Usuario…………………………………………………………... 22**

**9 Diseño del Software………………………………………………………….. 22**

**9.1 Diagrama agregar paciente………………………………………….. 23**

**9.2 Diagrama buscar paciente…………………………………………... 26**

**9.3 Diagrama eliminar tratamiento…………………………………….. 27**

**10 Conclusión…………………………………………………………………… 29**

**1.- Resumen**

El ser humano a lo largo de su vida, siempre ha buscado la forma de vivir más. Por su esfuerzo y logros desde los inicios de las civilizaciones, han surgido estudios y especializaciones para la mayoría de las cosas tangibles e intangibles que nos rodean.  
 Por el esfuerzo de ayudar al prójimo, mejorar la especie y vivir sanamente, se han formado los médicos, los cuales antiguamente no tenían una especialización ni tratamientos como hoy en día, por lo cual no existía una maestría en todos los aspectos de la medicina.  
 Ahora ya existen médicos especializados para las distintas partes del cuerpo, internas y externas, además de especializaciones para casi todas las enfermedades que han ido surgiendo a lo largo del tiempo.

En vista del tiempo, organización y administración que necesitan los médicos para cumplir su labor, se ha pensado en realizar un software que, a nivel de hospitales privados pueda organizar a los distintos pacientes que puede tener un médico, según la especialidad del doctor, el tratamiento que ejerce, cuantos pacientes tiene y cuál es la gravedad de cada paciente para su posterior atención o incursión en una lista de espera.

El comienzo de esta idea fue planteada por los miembros del grupo, pensando en las necesidades de los pacientes y la mejor forma en que el doctor pudiera organizar su tiempo para una posterior atención. La solución al principio no se veía muy complicada, pero con el pasar del tiempo, los detalles aparecieron y cada vez que nos planteábamos el problema desde la raíz, aparecía una nueva forma de solución o eliminábamos el punto que podría estar chocando con los demás datos.

El programa lo estamos desarrollando con el lenguaje “ANSI C” y la interfaz grafica esta implementada con el programa “Microsoft Visual Studio”. El programa es bastante amigable, ayuda de una manera bastante ordenada y compresible la solución del problema.

Ahora que ya ha pasado bastante tiempo desde el comienzo del software, podemos decir que se ha logrado lo esperado, la interfaz gráfica le da un carácter más profesional al programa y por ende, una forma más fácil para el usuario de ingresar, eliminar, cambiar, (etc.), los datos.

El software está funcionando correctamente, lo cual trae bastante satisfacción a los miembros del grupo, ya que a medida que avanzaba el tiempo se fue complicando, ya sea por el mismo lenguaje de programación o el uso e implementación de la interfaz.

Después de 2 meses desde el inicio de este proyecto, podemos decir que hemos logrado el objetivo, cumpliendo las metas que nos propusimos en el planteamiento del problema.

**2.- Introducción**

Como anteriormente mencionamos en el resumen, nuestro software esta direccionado en la administración y organización de hospitales privados, por lo que haremos todo este diseño en la perfección sobre el tiempo de los médicos, pacientes con tratamientos especificados, gravedad del tratamiento y un máximo de pacientes por médico.

En este informe se describirá el problema detalladamente, pensando en una mejor atención al cliente, al cual se le pedirán 5 datos, nombre, edad, domicilio, número telefónico y la gravedad de su enfermedad. Para el tratamiento habrá 3 elementos importantes, el nombre del tratamiento, el costo del tratamiento y el identificador del tratamiento. Para el médico se ingresaran 5 datos, nombre del médico, identificador de este, cantidad de pacientes totales que tiene, y la cantidad de pacientes que se están atendiendo con él en ese momento. Todo esto para facilitar el trabajo del operador, el cual ingresa, modifica o elimina datos y que el médico tenga un horario más eficiente y eficaz.

Luego de todo este análisis del problema, veremos que estructuras son las necesarias para poder crear este software.

Después mostraremos los respectivos diagramas de flujo para cada estructura planteada para cada parte del problema y los elementos mínimos que estaban en la pauta del proyecto, todo esto se hará en el programa “DIA”.  
  
Posteriormente se mostrara un anexo con el avance del código fuente, como nos planificamos para poder crear este proyecto, horarios, tabla Gantt y los avances que fuimos entregando en ayudantía, lo que mostraban nuestro progreso a medida que iban pasando las semanas. Y finalmente se dará a conocer la solución del problema, con el software en perfectas condiciones y la interfaz gráfica, mas una conclusión general de todo este proceso, opiniones de los integrantes, acotaciones sobre el programa o simplemente un análisis retrospectivo del problema y su solución.

**3.- Descripción Del Problema**

La organización de los hospitales en general, siempre ha repercutido en los tiempos que debe emplear un paciente respecto a doctores, lista de espera, tratamientos u otros elementos importantes dentro de un hospital.

Especificando y siendo más acorde a nuestro proyecto, los hospitales privados también tienen este problema de organización y administración. Por más que el tratamiento sea más caro, los tiempos que se emplean en estos, son aproximadamente parecidos, por lo cual queremos llevar esta administración a un punto clímax, en el cual los doctores no tengan sobre carga de pacientes, que los tiempos para ejecutar su profesión sean los justos y necesarios, de manera que se puedan atender el máximo de pacientes en un día, semana o mes.

En este último tiempo como las tecnologías se han desarrollado exponencialmente, los informáticos en general, han tratado de solucionar estos problemas mediante software que puedan ayudar en la organización de la información y que para los usuarios sea más fácil operar con estos.

Estos Hospitales privados necesitan tener un archivador con las listas de doctores, pacientes, tratamientos, los costos de los tratamientos u otros elementos que definiremos próximamente.

Si en un caso hipotético no se implementara este software, sería mucho menos eficiente y eficaz el trabajo general de estos hospitales. Tratamientos que podrían ser solucionados en mucho menor tiempo o sobrecarga de actividades respecto a los doctores, se podrían evitar perfectamente.

Para lograr estos objetivos y que el trabajo sea más preciso, eficiente y eficaz, implementaremos este software, pensado específicamente para hospitales privados, el cual necesite una administración más profesional, de manera que tendremos el hospital como tal, el cual tendrá doctores (con sus nombres, con ciertos tratamientos (cada doctor puede tener 1 o más tratamientos), la cantidad de pacientes que posee y la cantidad de pacientes que está atendiendo en este momento). Los tratamientos tendrán su nombre respectivo, un costo (el cual verá si el paciente puede pagar o no y se le asignará una deuda al nombre del paciente que se atendió con ese doctor). También estarán los pacientes, los cuales serán identificados por el nombre que tienen, edad, número telefónico, domicilio y su estado (El cual muestra la gravedad el paciente, para ver si necesita esta dentro de la lista de los pacientes que atiende el doctor en vez de estar en la lista de espera porque el doctor llego a su límite de pacientes (Existe un rango para este estado, el cual va de 1 a 10)

Con este software las secretarias y administradores de la información, podrán agregar, modificar, eliminar datos del programa, ya sea doctores, tratamientos o pacientes.  
Además de tener un rápido acceso a esta información para su implementación y los enlaces que existen entre doctores, pacientes y tratamientos, el Hospital subirá su nivel de atención respecto a la Salud y a medida que avance el tiempo, tendrá más clientes ya que su eficiencia y eficacia serán su sello.

Además el software es flexible, por ejemplo en el caso de que un hospital que tenga nuestro software quiera expandirse, ya sea poniendo nuevos médicos o nuevos tratamientos para un medico en especifico, o incluso si desea agregar más pacientes aun a médicos más experimentados y que sus tiempos de atención son sumamente rápidos, nuestro programa está apto para que puedan hacer todos estos cambios en cualquier hospital de Chile, por lo que si una empresa y/o empresas tiene pensado instalar un nuevo hospital, sea a corto o mediano plazo, nuestro software está completamente capacitado para este motivo, sin la necesidad de instalar otro software diferente.

Este software no solo serviría para pequeños Hospitales (o clínicas en el sentido privado), sino que también para los hospitales que ya llevan una estructura definida y que quieran cambiar su sistema para uno innovador y sobre todo preciso para lo que quiere y necesita cada paciente.

**4.- Análisis Del Problema**

El problema que nosotros queremos tratar es la falta de organización en los hospitales, ya que en variados medios de comunicación se muestra a gente que se queja, ya que deben esperar muchas horas para poder ser atendidos. Los datos que se necesitan para resolver este problema son variados, como por ejemplo los nombres de los doctores(los cuales tendrán una cantidad de pacientes especifica y al mismo tiempo solo un parte de estos anteriores serán los que se están ateniendo con él en un rango de tiempo determinado) y pacientes, se necesitaba saber la gravedad del paciente, los tratamientos que puede atender el doctor y el costo de los tratamientos.

**4.1.- Estructura Hospital.**

Lo primero que se pensó fue la estructura del Hospital, la cual es un estructura de encapsulamiento, porque el hospital nunca cambia, es decir que siempre el hospital será el mismo, tal vez con el pasar del tiempo éste cambie, pero no es común que se modifiquen un hospital cada año. Pero con una buena mantención en el software, se le puede modificar aspectos a éste, en el caso de que se quisiera modificar la estructura.

struct Hospital

{

intult\_doctor;

intult\_costo;

struct Doctor \*doctor[30];

struct Costo \*costos[50];

};

Figura 4.1.1 Estructura Hospital

Aquí se puede observar lo que contiene la estructura hospital, que es 2 enteros que contienen la posición del último dato de los doctores y la posición del último dato de los costos, además posee 2 estructuras las cuales son, doctor y costo, que apuntan a una lista contigua de largo 30 llamada doctor y costos, respectivamente.

Con esta estructura tendremos acceso a la estructura de los doctores, en los cuales aparecen todos los datos de estos. Y también acceso a los costos de los tratamientos según el médico de la estructura anterior.

Representación gráfica de la estructura hospital anteriormente señalada:

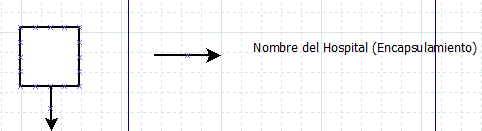


Figura 4.1.2 Representación Estructura Hospital.

**4.2.- Estructura Doctores**

Luego del hospital vienen los doctores, que son todos los doctores que posee el hospital con sus respectivos datos. La estructura doctores es una lista contigua, ya que sabemos específicamente que hay solo un hospital el cual está ligado a los doctores, de la misma forma que sabemos cuántos pacientes hay por doctor. Además porque puede haber que algún doctor que se retire o "pida vacaciones", por lo cual en este caso habría que contratar a mas doctores por demasiada demanda de pacientes.

Para trabajar con los doctores asignamos una casilla dentro del vector para cada doctor, los cuales tienen una memoria específica en la cual ingresaremos los datos del doctor, meramente necesarios para desarrollar el problema y no gastar memoria en datos irrelevantes.

struct Doctor

{

char \*nombre;

int ID;

char \*especialidad;

intcant\_pacientes;

intcant\_pac\_atend;

struct Tratamientos \*Tratamientos[3];

};

Figura 4.2.1 Estructura doctor.

El contenido de cada doctor es un nombre (\*nombre), el respectivo identificador (ID) que posee para poder diferéncialo de los demás doctores, su especialidad (\*especialidad), la cual puede ser más de 1, también tendrá la cantidad de pacientes totales que posee, para ver que no tenga una sobrecarga de pacientes y solo pueda atender el máximo que le hemos asignado. Además de la cantidad de pacientes, estará la cantidad de pacientes que se están atendiendo, para que dentro de un rango de tiempo no sean sobrellevados algunos pacientes y que la gravedad o estado que poseen se respete. (Todo esto organizado por el software)  
Finalmente estará ligado a la estructura de los tratamientos que apuntara a una lista contigua de largo 3, ya que por rango máximo hemos planteado que 3 tratamientos para un doctor es suficiente.

Representación gráfica de la estructura doctores anteriormente señalada:

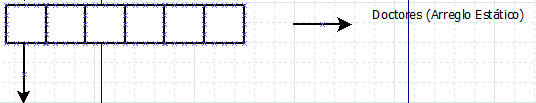


Figura 4.2.2 Representación Estructura doctores.

**4.3.- Estructura Tratamientos**

Posterior a la estructura doctores está la estructura tratamientos, que es una de las estructuras más importantes dentro del software, porque aquí se encuentra lo que es visible para el usuario, es decir, valga la redundancia los tratamientos y sus costos respectivos, que es lo más importante para el usuario del programa, ya que el cliente de nuestro software lo compró para una organización y administración más profesional de cómo escoger médicos según los pacientes que posee y los tratamientos que tiene.

struct Tratamientos

{

char \*nombre;

long costo;

intid\_tratamiento;

struct Paciente \*paciente;

};

figura 4.3.1 Estructura tratamiento.

El contenido de la estructura tratamiento comienza con el nombre del tratamiento, el costo del tratamiento, el cual está en “long”, ya que la cantidad puede ser considerable.  
 Luego tenemos el identificador del tratamiento (ID), el cual nos servirá para diferéncialo de los demás.

Finalmente estará ligado a una estructura pacientes que apunta a los pacientes respectivos de ese tratamiento.



Figura 4.3.2 Representación Estructura tratamiento.

**4.4.- Estructura Paciente**

Esta es una estructura creada posterior a todas las demás estructuras, ya que ella contiene a los usuarios o más bien pacientes que serán atendidos por doctores especializados en ciertas áreas.

Esta es la más importante de todas las estructuras, ya que contiene los datos más importantes del sistema, los pacientes y/o personas para las cuales fue creado este software y poder satisfacer una necesidad o mejor la calidad de servicio.

struct Paciente

{

struct Persona \*persona;

struct Paciente \*sgte;

};

figura 4.4.1 Estructura paciente.

Lo que contiene esta estructura son 2 estructuras importantísimas, las personas y los pacientes. En la estructura persona, se está apuntado a las personas, las cuales terminaran siendo los pacientes a lo largo de los enlaces y traspasos de datos a lo largo del software.  
Por último la estructura paciente que apunta al siguiente paciente.

Representación gráfica de la estructura Paciente anteriormente señalada:

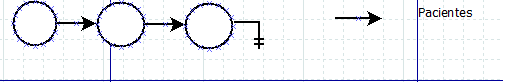


Figura 4.4.2 Representación Estructura Pacientes.

Se puede observar que es una estructura simplemente enlazada, la cual no tiene ningún enlace con otra estructura, este podría ser un nodo final, ya que no tiene a nadie que lo proceda.

Ahora que están representadas todas las estructuras se puede tener una imagen general del programa, más que nada una imagen de diseño del software, de cómo se relacionan sus partes, cuales son las estructuras que le deben importar más al usuario, etc.

Concluyendo este punto se mostrará una representación global del programa, es decir una representación que contenga todas las estructuras enlazadas, dando una visión global del software.

**4.5.- Estructura Persona**

Esta sirve para crear o más bien agregar a las personas dentro del sistema, las cuales se convertirán en pacientes.

struct Persona

{

char \*nombre;

int edad;

char \*num\_telefonico;

char \*domicilio;

char \*estado;

int nivel\_estado;

};

Figura 4.5.1 Estructura Persona.

Esta estructura consta de una variable tipo char, llamada \*nombre, en la cual se ingresará el nombre de la persona, luego se ingresara la edad respectiva de la persona, su numero telefónico, el domicilio y el estado o gravedad en la que se encuentra para el posterior tratamiento, además del numero entre 1 y 10 que representa el rango del estado.

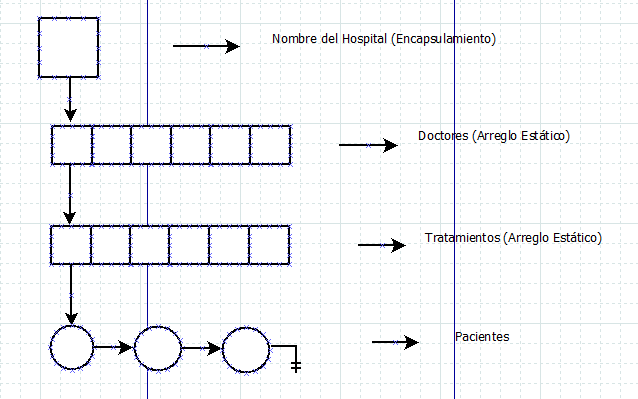
**4.6.- Estructura Costos**

Esta estructura es una lista contigua en la cual se guardan los pacientes que no han pagado su tratamiento por completo.

Consta de una variable tipo char llamada \*nombre, en la cual se ingresará en nombre de la persona que pagara su tratamiento, siempre y cuando se esté atendiendo y no ha pagado el costo total del tratamiento y una variable tipo long llamada costo que guarda el costo pendiente del tratamiento. Ejemplo: si el tratamiento cuesta 200000 y el paciente no puede paga su costo total y solo paga 100000, el paciente queda registrado en la lista de los costo y su monto pendiente por pagar será de 100000.

**4.7.- Diagrama General**

En la siguiente imagen se mostrará la representación del programa entero, es decir todas las estructuras que se explicaron anteriormente se encuentran en este dibujo. El fin de esa representación es tener una idea más clara de cómo se estructura el software.



**5.- Manual Programador**

El fin o propósito del manual del programador es informarlo sobre el mantenimiento del programa, el cual consta de dos tipos de documentación, la interna y la externa.

En la documentación interna se tratará más bien del código en si del software, y la documentación externa se enfoca en los archivos de texto con los que trabaja el programa.

**5.1.- Documentación Interna**

Para la persona que sea el encargado de leer el programa, va a tener la facilidad de que el software está comentado en toda su extensión, para una mejor comprensión de este. Como ejemplo se mostrará un trozo de código del programa:

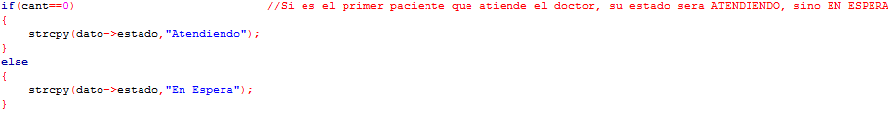


Figura 5.1.1 Representación Trozo De código.

Como vimos en el ejemplo anterior lo que aparece en rojo son los comentarios, que ayudan al programador a tener una mejor comprensión del código.

**5.2.- Documentación Externa**

Para que los datos ingresados por pantalla queden guardado mediante del software, se ocuparán distinto archivo de texto (\*txt). Primero que nada el software cargará el archivo de texto llamado “doctores.txt”, en el cual conlleva el nombre de los doctores junto con otros datos.

JORGE SEPULVEDA

1 Neurologia 4

ALBERTO CABRERA

2 Psiquiatria 5

JUAN MORENO

3 Geriatria 4

VICTOR MUNOZ

4 Oftalmologia 6

CLAUDIO CUBILLOS

5 Otorrinolaringologia 3

VIVI ALVARADO

6 Psiquiatria 6

ALAN GONZALEZ

7 Neurologia 3

MAX MOYANO

8 Dermatologia 5

Figura 5.2.1 Datos del fichero “doctores.txt”.

Una vez cargado los datos de los doctores, se cargara los datos de los tratamientos que atiende cada doctor, en total son 3 tratamientos por doctor. Estos datos se cargan al software por medio del archivo de texto llamado “tratamientos.txt”.

Neurologia

Meningitis 1 400000

Cerebro\_vasculares 2 500000

Epilepsias 3 300000

Psiquiatria

Fibrositis 1 300000

Transtornos 2 150000

Psicoterapia 3 400000

Geriatria

Hipertension 1 300000

Diabetes 2 200000

Articulaciones 3 400000

Oftalmologia

Conjuntivitis 1 100000

Glaucoma 2 300000

Queratitis 3 400000

Otorrinolaringologia

Alalia 1 300000

Laberantitis 2 400000

Otitis 3 200000

Figura 5.2.1 Datos del fichero “tratamientos.txt”.

Después de haber cargado al software los tratamientos de cada doctor, se comenzará a cargar los datos de los pacientes, siempre y cuando haya pacientes guardados en el fichero llamado “pacientes.txt”, esto conlleva el nombre del doctor, el nombre del tratamiento y los datos del paciente.

VICTOR MUNOZ

Glaucoma

Gonzalo Andrés

24

68676576

Vina\_del\_mar

3

Figura 5.2.1 Datos del fichero “pacientes.txt”.

Por último, se cargara en el software los datos de los costos o deudas de los pacientes por medio de un fichero llamado “costos.txt”, siempre y cuando haya pacientes con deudas pendientes.

Gonzalo Andrés 10000

Figura 5.2.1 Datos del fichero “costos.txt”.

Una vez cargados todos los datos y haber ocupado el programa, se actualizaran cada uno de los ficheros con la información correspondiente

**6.- Planificación Grupal**

|  |  |  |  |
| --- | --- | --- | --- |
| Administración de hospitales privados | |  |  |
| Tarea | Fecha Inicio | Duración (días) | Fecha Termino |
|  |  |  |  |
| Análisis de requerimientos del  proyecto + Análisis de fortalezas  y debilidades | 23-09-2013 | 4 | 27-09-2013 |
| Diseño gral estructuras (3  niveles) + Funciones extras +  Diseño de archivos I/O | 30-09-2013 | 4 | 04-10-2013 |
| Código para cargar los 3 niveles  + Código mantenedor en 1º  nivel | 07-10-2013 | 4 | 11-10-2013 |
| Diseño de menus e Interfaz GUI  + Código para grabar los 3  niveles | 14-10-2013 | 4 | 18-10-2013 |
| Entrega Proyecto Parte A | 25-10-2013 | 7 | 01-11-2013 |
| Defensa Proyecto Parte A | 04-11-2013 | 4 | 08-11-2013 |

**6.2.-Planificación individual**

|  |  |  |  |
| --- | --- | --- | --- |
| Jorge Sepúlveda |  |  |  |
| Tarea | Fecha Inicio | Duración (días) | Fecha Termino |
| Análisis del programa | 23-09-2013 | 4 | 27-09-2013 |
| Creación del diseño de las estructuras | 30-09-2013 | 4 | 04-10-2013 |
| Planificación de las funciones extras | 07-10-2013 | 4 | 11-10-2013 |
| Código para cargar los 3 niveles | 14-10-2013 | 4 | 18-10-2013 |
| Finalización de los diagramas de flujo | 25-10-2013 | 6 | 31-10-2013 |
| Entrega Proyecto Parte A | 31-10-2013 | 1 | 01-11-2013 |
| Defensa Proyecto Parte A | 04-11-2013 | 4 | 08-11-2013 |

|  |  |  |  |
| --- | --- | --- | --- |
| Juan Moreno |  |  |  |
| Tarea | Fecha Inicio | Duración (días) | Fecha Termino |
|  |  |  |  |
| Comprender el código actual de mis compañeros | 23-10-2013 | 1 | 24-10-2013 |
| Avance en el informe de proyecto | 24-10-2013 | 1 | 25-10-2013 |
| Ayudar a solucionar errores en el código | 25-10-2013 | 2 | 27-10-2013 |
| Finalizar todos los detalles del código e informe de proyecto | 27-10-2013 | 3 | 30-10-2013 |
| Entrega Proyecto Parte A | 31-10-2013 | 1 | 01-11-2013 |
| Defensa Proyecto Parte A | 04-11-2013 | 4 | 08-11-2013 |

|  |  |  |  |
| --- | --- | --- | --- |
| Victor Muñoz |  |  |  |
|  |  |  |  |
| Tarea | Fecha Inicio | Duración (días) | Fecha Termino |
| Análisis del programa | 23-09-2013 | 4 | 27-09-2013 |
| Planificación del diseño de las estructuras | 30-09-2013 | 4 | 04-10-2013 |
| Creación de las funciones extras | 07-10-2013 | 4 | 11-10-2013 |
| Creación código para cargar los 3 niveles + Interfaz grafica | 14-10-2013 | 4 | 18-10-2013 |
| Modularización y arreglo de detalles | 25-10-2013 | 6 | 31-10-2013 |
| Entrega Proyecto Parte A | 31-10-2013 | 1 | 01-11-2013 |
| Defensa Proyecto Parte A | 04-11-2013 | 4 | 08-11-2013 |

|  |  |  |  |
| --- | --- | --- | --- |
| Alberto Cabrera |  |  |  |
|  |  |  |  |
| Tarea | Fecha Inicio | Duración (días) | Fecha Termino |
| Análisis del programa | 23-09-2013 | 4 | 27-09-2013 |
| Desarrollo Informe | 30-09-2013 | 4 | 04-10-2013 |
| Desarrollo avances de entrega en ayudantías | 07-10-2013 | 4 | 11-10-2013 |
| Casos de prueba al código del proyecto | 14-10-2013 | 4 | 18-10-2013 |
| Finalización informe + Apoyo interfaz grafica | 25-10-2013 | 6 | 31-10-2013 |
| Entrega Proyecto Parte A | 31-10-2013 | 1 | 01-11-2013 |
| Defensa Proyecto Parte A | 04-11-2013 | 4 | 08-11-2013 |

**7.- Manual Del Operador**

Este manual consta en el ingreso de datos y modificaciones de estos. Esto es para que el operador tenga una referencia al usar el programa por primera vez. Se realizará una división en este manual del operador, que será en el ingreso de datos y modificación de datos.

**7.1.- Manejo por consola**

Aquí se explicará en qué consisten las opciones de ingreso que tiene el programa, si bien deberían ser perfectamente entendibles para el operador mediante la interfaz, igualmente en este manual explicaremos estas opciones de ingreso de datos en el software, para que no haya malentendidos.

 Primero que nada haremos doble clic al archivo “nombre.exe” en el cual nos abrirá una ventana del proyecto.

Figura 7.1.1 Ventana de la interfaz del proyecto.

Al hacer clic el botón ejecutar, nos arroja la consola MS-DOS en el cual podemos trabajar con del software y al hacer clic al botón ejecutar se cierra la ventana de la interfaz.

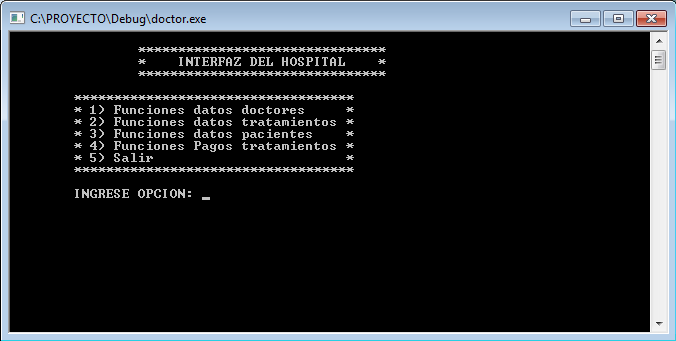
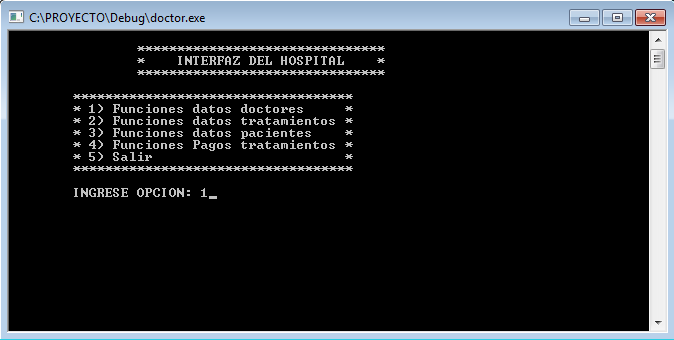
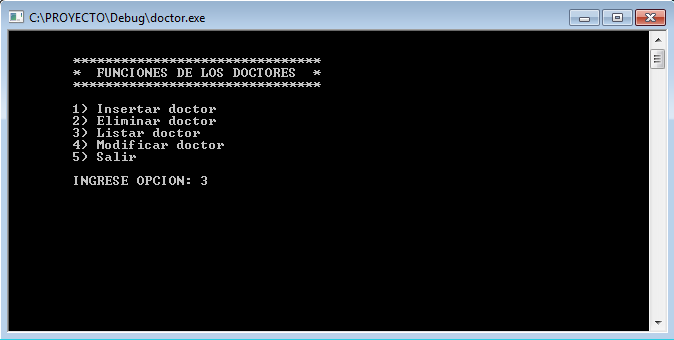


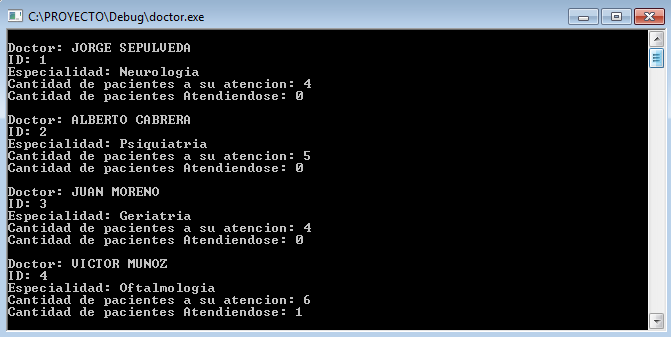
Figura 7.1.2 Consola MS-DOS con todas sus funciones.

Con la consola una vez abierta, podemos trabajar con el software a nuestra voluntad, solo se elige una opción presionando el número de la función correspondiente y presionando “enter”. Si que quiere elegir la opción de listar los doctores se debe seguir estos pasos:





****

****

**8.- Manual Del Usuario**

Este manual consta en aportar toda la información necesaria al usuario para que su uso sea de un 100%, el usuario tiene el poder de utilizar el programa como a el mejor le parezca, lograra adaptarse a las necesidades de cada usuario según su necesidad, ejemplos de esto es:

* El ingreso, modificación, y eliminación de doctores.
* El ingreso, modificación y eliminación de pacientes.
* Traslados de pacientes de un doctor a otro.
* Estado de deudas de pacientes.
* Etc.

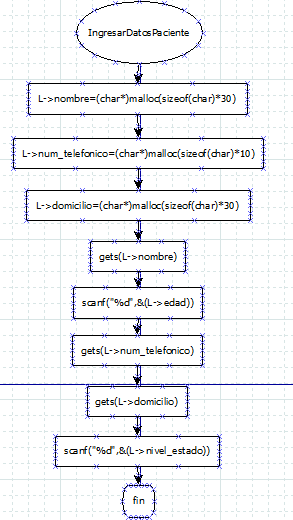
El uso de todos estos puntos y más, estarán explicados en el programa para que el usuario pueda desenvolverse de la manera que más le acomode para el desarrollo y mantenimiento de su hospital y software.

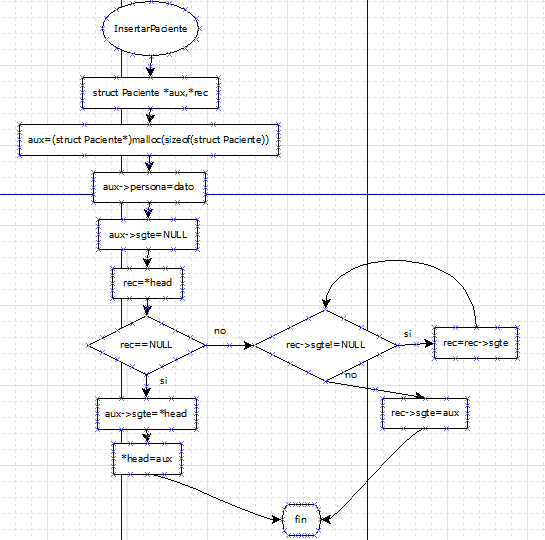
**9.- Diseño Del Software**

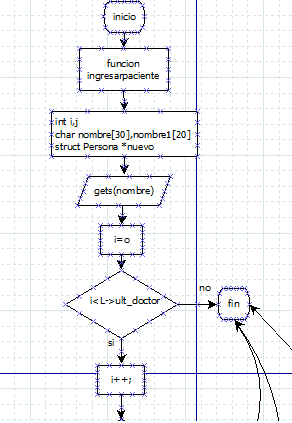
En esta sección se procederá a mostrar diagramas de flujo del programa de algunas partes del programa, estos diagramas fueron hechos con la herramienta DIA.

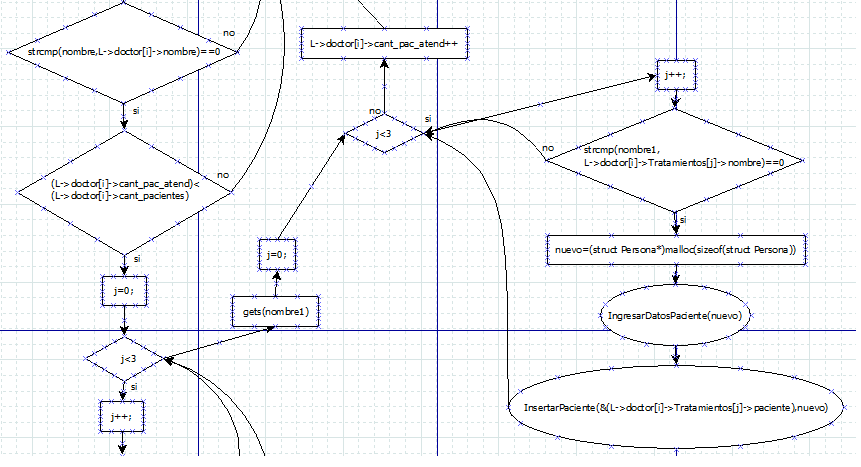
Aquí están los diagramas de flujo, los cuales están organizados uno en una página, de manera que los de agregar un paciente están primero, luego los diagramas de buscar un paciente y finalmente los diagramas de eliminar un tratamiento.

Figura 9.1.Diagramas de flujo de agregar un paciente.

****

****

****

****

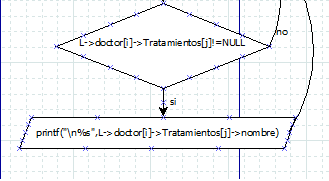
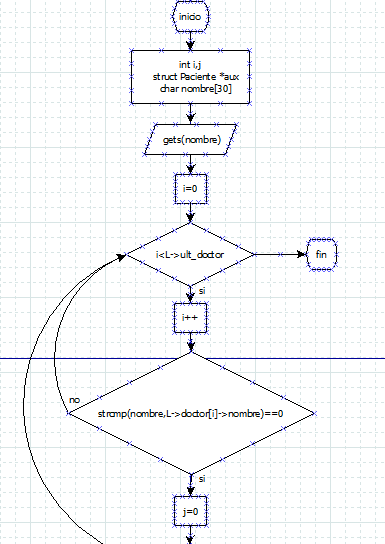
****

Figura 9.2.Diagrama de flujo de buscar un paciente.

****

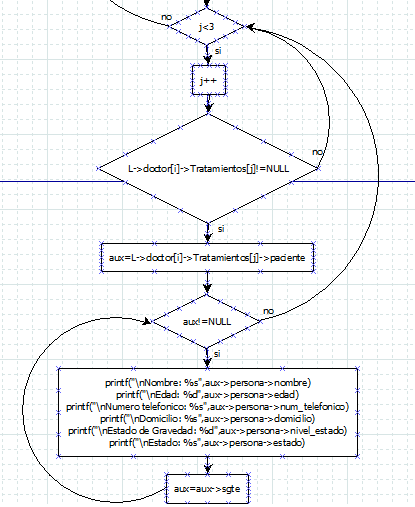
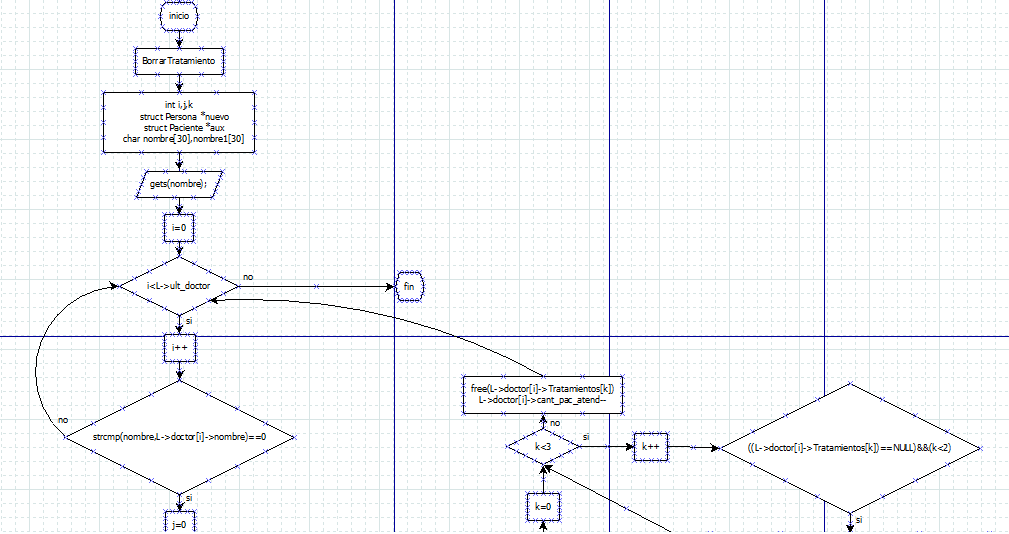
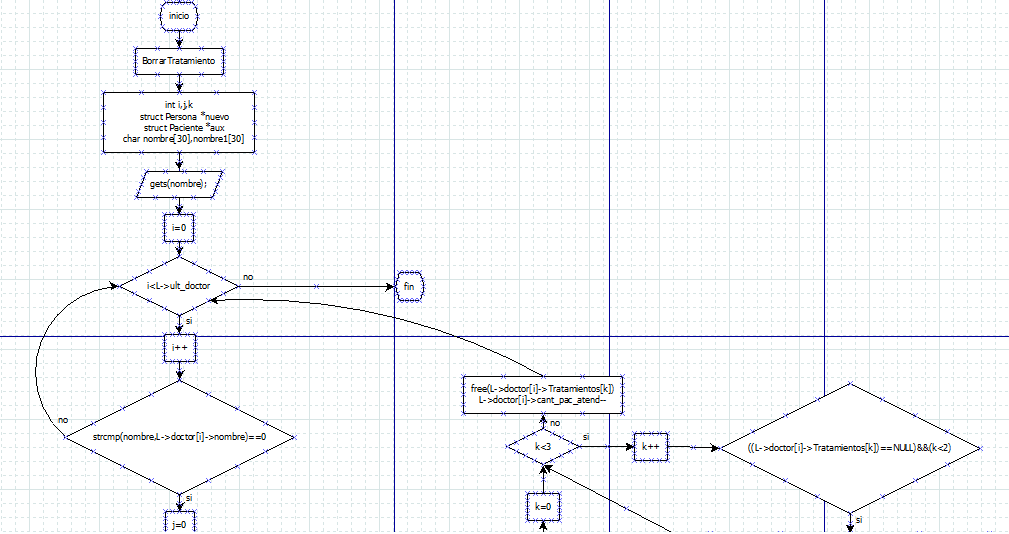
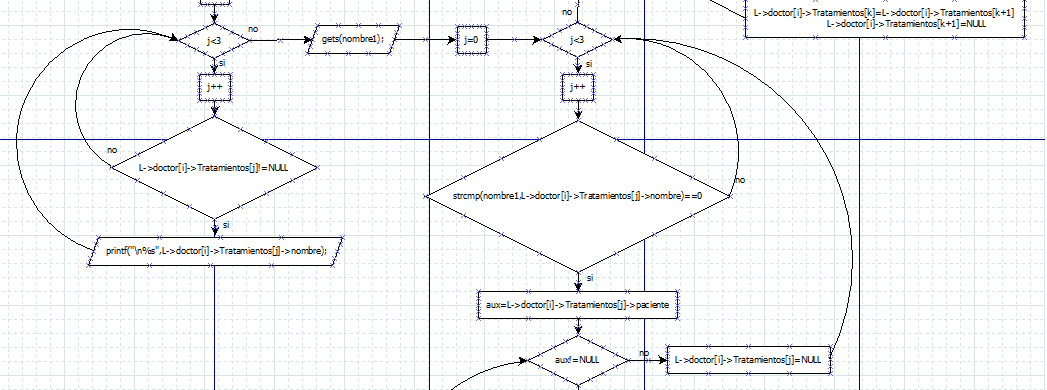
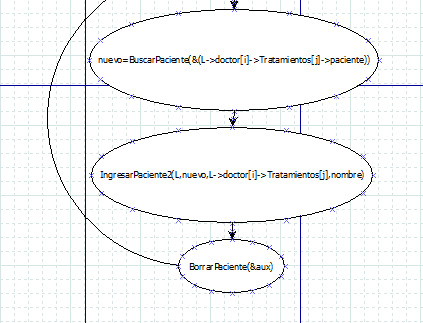
****

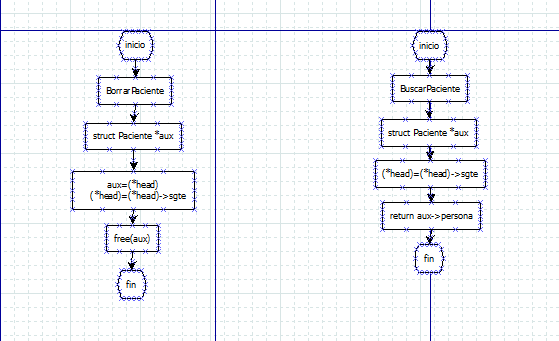
Figura 9.3.Diagrama de flujo de eliminar un tratamiento.

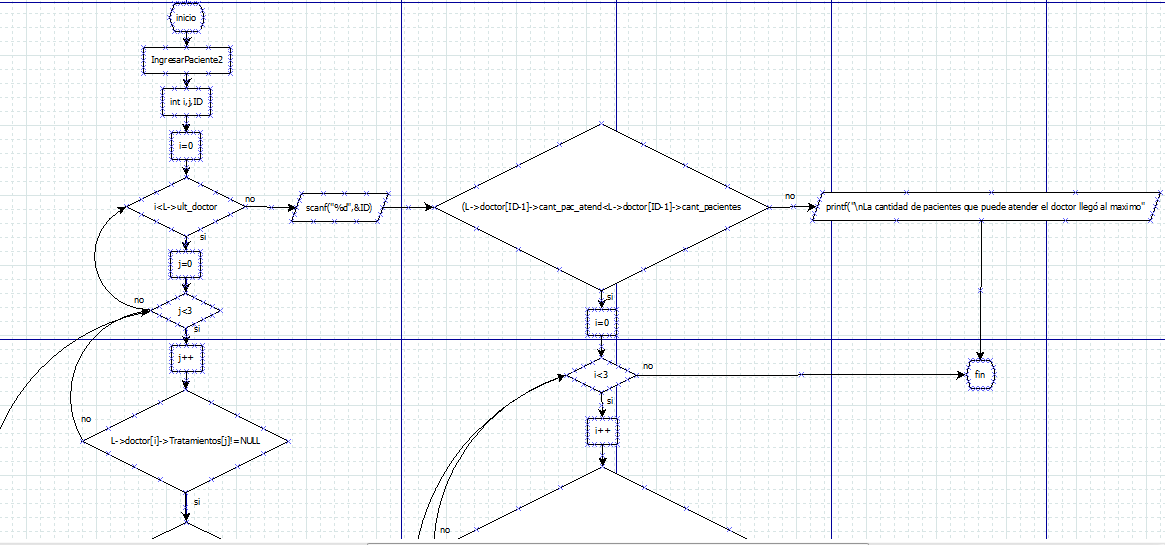
****

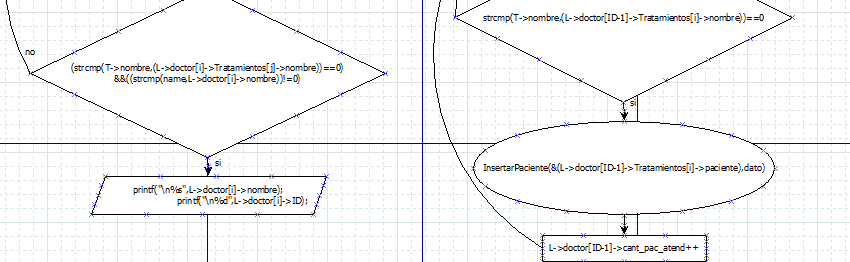
****

****

****

****

****

****

**10.- Conclusión**

Mientras desarrollábamos este proyecto tuvimos la oportunidad de mejorar nuestros conocimientos en el lenguaje de programación C, también nos sirvió para aprender a trabajar en grupo de forma más organizada y además nos ayudó a indagar sobre las interfaces. Nosotros como grupo superamos nuestras expectativas, ya que pensábamos que sería un proyecto muy complejo y largo, pero gracias a los avances que nos iban pidiendo en las ayudantías semanalmente, avanzábamos todas las semanas un poco, eso ayudo a que hiciéramos el trabajo con tiempo y sin estar tan presionados. A medida que avanzaba el proyecto íbamos encontrando complicaciones y en ocasiones era bastante estresante, pero supimos seguir adelante y terminar bien el proyecto, a este proyecto es necesario dedicarle mucho más tiempo que a otros trabajos, porque a pesar de los avances de las ayudantías, se necesitaba bastante tiempo para terminar los avances en la fecha que se pedía, además que no todos los integrantes del grupo tenemos el mismo nivel de conocimiento en C, algunos saben mucho más y otros van a su propio ritmo. Otra cosa que aprendimos durante la realización de este proyecto fue a pensar de una manera más abstracta, a pesar de que no todos tenemos el mismo nivel de conocimiento en C, con el proyecto tuvimos la oportunidad de mejorar, aprender nuevos conceptos como la modularización para que los programas queden más ordenados y sean más amigables y también el encapsulamiento que es otra forma de hacer la estructura más ordenada.