PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Informe Proyecto Parte B
"Atención a Público"

Cristian Andrés Vargas Acevedo
Peter Ignacio Jorquera Guillen
David Eduardo Espinoza Olguín
Sergio Andrés Gaete Carrasco

Asignatura: inf-245 Estructuras de Información
Profesor: Claudio Cubillos
Fecha: 19-11-2012

Índice

1 Introducción

En la actualidad se presentan problemas en el tratamiento de atención al público. Provocando muchos inconvenientes en la relación entre personas y empresas, ya que se vuelve engorroso y poco ágil la comunicación de las inquietudes o problemas que tienen las personas en las distintas interacciones que producen con las empresas.

El presente proyecto a continuación va a tratar este problema proponiendo soluciones para que estas empresas tengan una eficiente gestión de la información que le brindan las personas y para ello se ocupan un modelado, diseño y aplicaciones varias de estructuras de datos en un software de gestión.

Aquí aplicaremos varios métodos de búsqueda y ordenamiento de datos de esta empresa para que el usuario no tenga que gastar más tiempo del necesario en la ejecución ya que todo a medida que se van ingresando datos inmediatamente estos son ordenados de manera tal que al buscaros sean encontrados de una manera más eficaz en tiempos de ejecución de software.

2 Descripción del Problema

Se pretende crear un programa que sea capaz de gestionar la llegada de público para una empresa de cualquier tipo por esto se han analizado diferentes sitios web de alta concurrencia para visualizar el problema

Los aspectos básicos que notamos fueron:

- Servicio de reclamos y o consultas.
- Información de contacto (fono, mail, dirección, horarios de atención).
- Registro de clientes (publico), por Rut o código de atención.
- Sistema de valoración cualitativa para determinadas sucursales.

A partir de estas funciones implementaremos otras como:

- Sucursales con mayor número de reclamos o peor calificación.
- Sucursal mejor evaluada.

3 Análisis del problema

Para solucionar el problema del Software de Gestión de sistema de encuestas a empleados para recursos humanos, lo que se propone en este trabajo es utilizar un Conjunto de Estructuras de Datos Anidadas, de tal forma que sea símil a la disposición jerárquica que posee la Empresa.

A continuación la Figura Nº 2 muestra la jerarquía del problema:

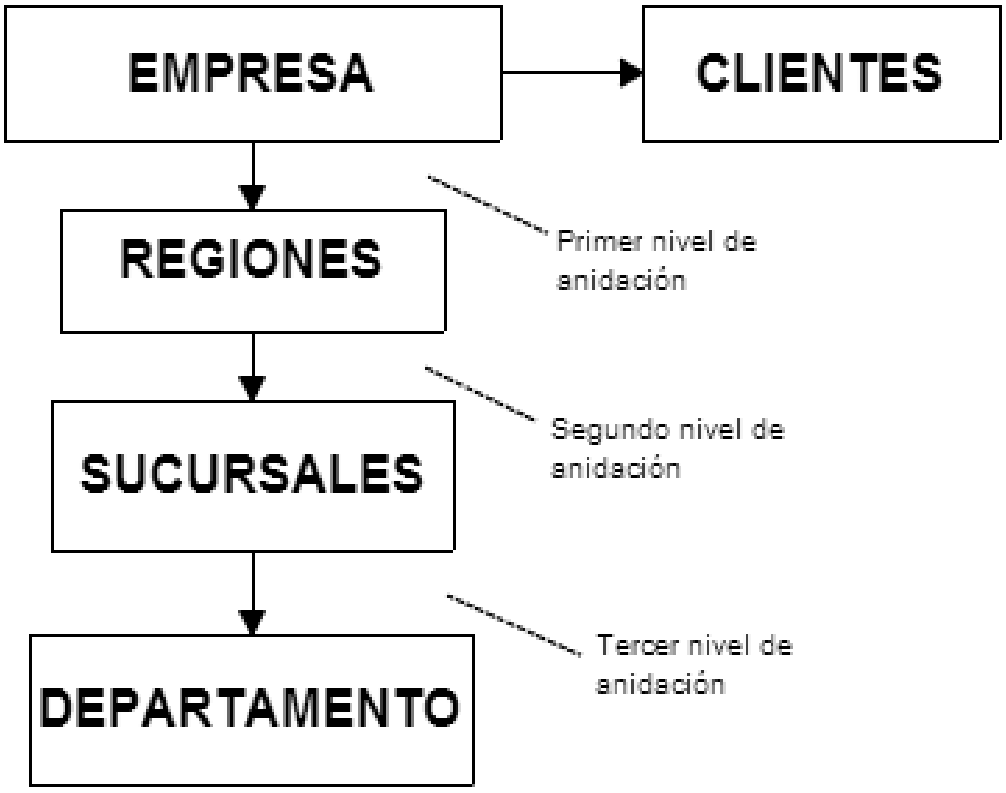


Figura Nº2

En la figura N° 2 se muestra la jerarquía de los datos que desarrollamos en el problema, esto a su vez representa cada una de las anidaciones de las estructuras del código del problema.

3.1 Análisis de datos de entrada

3.1.1 Datos de entrada de la Empresa

- Nombre y Rut.
- Para controlar el vector de punteros de Región.
- Utiliza clave para ingresar modo administrador.
- Además de un enlace a la estructura Región y Cliente.

Estructura que almacenara los datos de la empresa:

```
struct Empresa
{
   char *nombre;
   char *rut;
   int     ultimo;
   int     clave;

   struct Regiones *Region [15];
   struct Cliente * Arbol_clientes;
};
```

3.1.2 Datos de entrada de las Regiones

a) Nombre: Almacena el nombre de la región.
b) id: Código identificador de cada región.
c) Además de un enlace a la estructura Sucursal.

Estructura que almacenara los datos de las Regiones:

```
struct Region
{
   char *nombre;
   int ID;
   struct Sucursales *Lista_sucursales;
};
```

3.1.3 Datos de entrada de las Sucursales

- Nombre, dirección, comuna, ciudad y fono de la sucursal.
- id: Identificador de una sucursal en especial.
- Dirección: Almacena la dirección de la sucursal.
- También el nombre, Rut del jefe de cada sucursal.
- id_boss: identificador del jefe.
- Además de un enlace a departamento.|

Estructura que almacenara los datos de la sucursal:

```
struct sucursal
{
   char *nombre_suc;
   char *direccion_suc;
   char *comuna_suc;
   char *ciudad_suc;
   char *fono_suc;
   int  ID;

   char  *name_boss; // nueva modificaciones
   char  *rut_boss;
   int    id_boss ;

   struct Departamento *Lista_dpto;

   struct Sucursales *sig, *ant;
};
```

3.1.4 Datos de entrada de los Departamentos

a) Nombre y teléfono del departamento.
b) Además de un enlace a la estructura empleado.

Estructura que almacena los datos de los departamentos:

```
struct departamentos
{
   char *nombre;
   char *telefono;
   int valoracion;

   struct Departamento * sig;
};
```

3.1.5 Datos de entrada de los Clientes

- Nombre, Rut, dirección, ciudad, teléfono, email, edad de los clientes.
- ID_consulta: Identificador de cada consulta.
- estado_reclamos: Indica la cantidad de reclamos.

Estructura que almacenara los datos de los empleados:

```
struct Cliente
{
   char *nombre;
   char *rut;
   char *direccion;
   char *ciudad;
   char *telefono;
   char *email;
   int   edad;
   int   ID_consulta;
   int   estado_reclamos;
   struct Cliente *izq,*der ;
};
```

3.2 Restricciones

a) Las regiones son limitadas.

b) El sistema ocupa solo un nodo departamento.

c) Los archivos de reclamo no se muestran solo se almacenan.
.

4 Diseño

4.1 Desarrollo de solución

Para el desarrollo de la solución de este problema utilizaremos las siguientes estructuras anidadas:

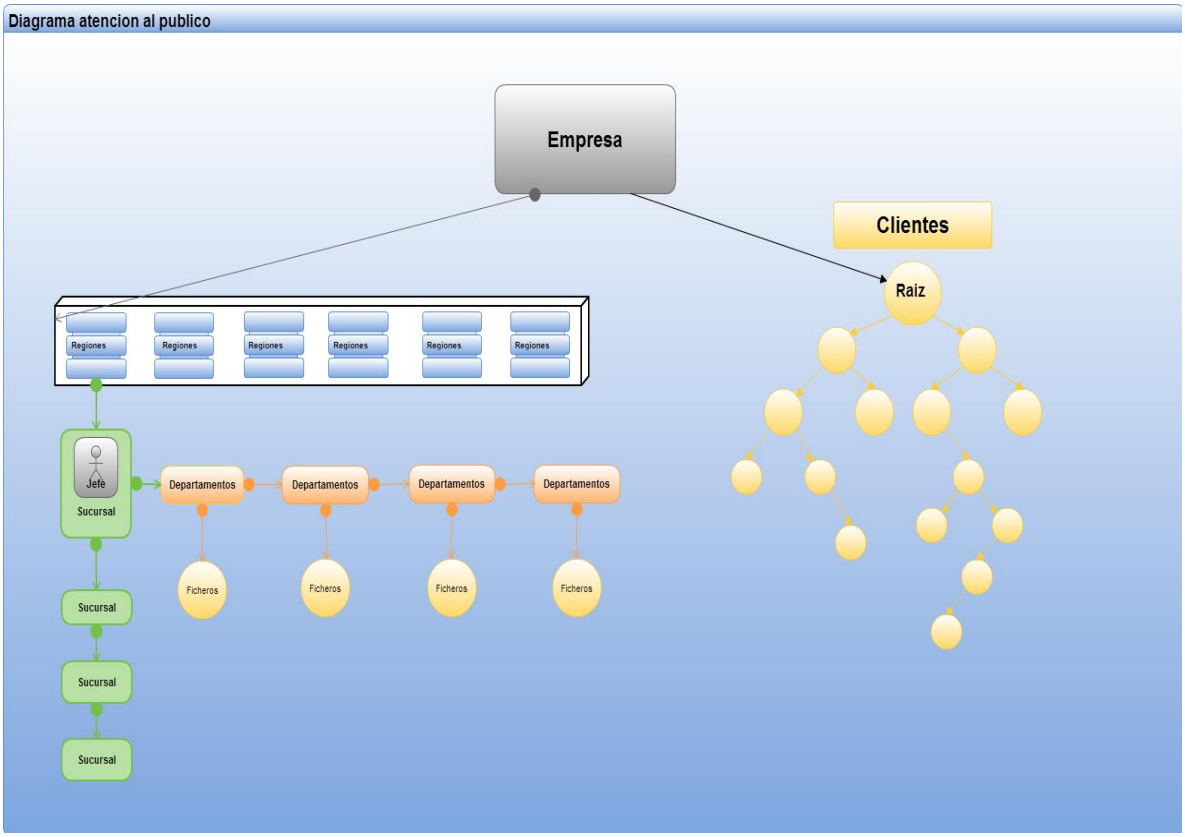


Figura Nº 3

En la figura N° 3 se muestra la estructura principal del programa. La estructura principal que corresponde a la empresa, la contiene los cuatro niveles de anidación. El primer nivel de anidación corresponde a las distintas regiones donde está la posibilidad de crear, eliminar o existir una sucursal de la empresa, la cual es el segundo nivel de anidación (máximo 15 regiones en todo el país), cada sucursal tiene un enlace a distintos departamentos el cual corresponde al tercer nivel de anidación.

4.2 Desarrollo del problema (lenguaje c)

Las estructuras que se utilizaran son un vector de punteros, una lista doblemente enlazada, una lista simplemente enlazada, un árbol abb y una lista circular unida a la primera estructura:

4.2.1 Empresa:

La estructura empresa contiene los datos formales, tales como el nombre la empresa, el rut, la clave y esta tendrá un puntero hacia la primera posición del arreglo que corresponde a las regiones. Cabe destacar que la estructura empresa es única, por esto para poder crear una nueva empresa es necesaria borrar toda la información de la ya existente.

Figura Nº 4

La Figura Nº4 posee la estructura empresa, con todos los datos que esta contiene además de un arreglo unidimensional de 15 regiones de tipo estructura región y un puntero apuntando a un árbol abb enlazada tipo estructura Cliente.

4.2.2 Región:

Esta estructura se manejara con una estructura estática, arreglo unidimensional, de largo 15. La cual contiene una estructura con los datos de cada una de las regiones como su nombre, id. Además de un puntero a la cabecera de “Sucursales”.

Figura Nº 5

La figura Nº 5 posee la estructura región, y las variables que la componen, además de un puntero de tipo sucursales.

4.2.3 Sucursales:

Esta estructura también será manejada por una estructura dinámica, lista doblemente enlazada, cada uno de los nodos de la lista doble contiene los datos de las sucursales que posee o tendrá la empresa. Para aquello se le pedirán datos de cada sucursal.

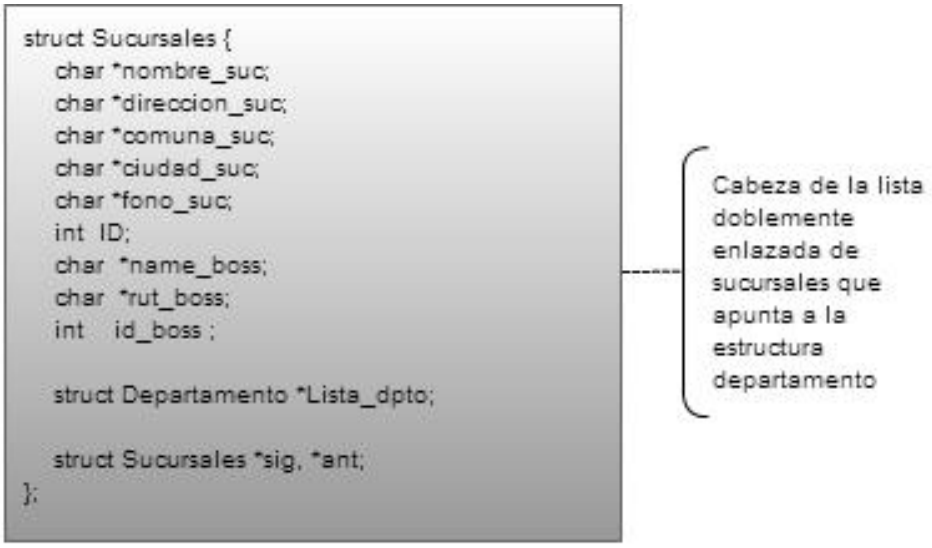


Figura Nº6

La Figura Nº 6 muestra la estructura de tipo sucursal, con cada una de las variables que le pertenecen, incluyendo los punteros con los cuales avanza esta lista. Además de un puntero a la estructura "Departamento".

4.2.4 Departamentos:

Esta estructura será llevada a cabo, a través de una estructura dinámica, lista simplemente enlazada. Cada nodo de esta lista contiene datos de cada uno de los departamentos de las distintas sucursales. Tiene un puntero hacia Ficheros.

Figura Nº 7

La Figura Nº 7 posee la estructura departamentos, esta al igual que las anteriores contiene punteros, con los cuales avanza a través de la lista enlazada. También contiene sus propias variables y un puntero a ficheros.

4.2.5 Clientes:

Esta estructura también será manejada por una estructura dinámica, lista circularmente enlazada, cada uno de los nodos de la lista contiene los datos del cliente que hará la el reclamos o consultas. Para aquello se le pedirán datos personales del cliente y sus respectivos reclamos asignados en un fichero, para luego ser llevado a el departamento correspondiente y luego el jefe responderá, también será utilizada una función para calcular los clientes con más reclamos.

Figura Nº 8

La Figura Nº 8 contiene a la Estructura árbol ABB cliente, esta posee dos punteros para avanzar en el árbol, además de las otras variables.

4.3 Funciones necesarias para el manejo de datos

El programa principal constara de un menú el cual permitirá realizar varias opciones tales como: Agregar, Eliminar, Modificar, Buscar, Listar. Y dos funciones extras la primera va a recorrer la lista cliente buscando el con más reclamos. Y una segunda…

Funciones dedicadas a crear un nuevo nodo en las listas, con sus datos respectivos:

- Crear Empresa: Retorna un nuevo nodo empresa.
- Crear regiones: Recibe un doble puntero empresas e inicializa el vector de punteros región.
- Crear sucursal: Retorna el nodo sucursal.
- Crear cliente: Retorna el nodo cliente.
- Crear Departamento: Retorna el nodo departamento.

Funciones dedicadas a enlazar los nodos en las listas:

- Enlazar Suc: Recibe el nodo empresa y el nuevo nodo sucursal.
- Enlazar clientes: Recibe el nodo empresa y el nuevo nodo cliente.
- Enlazar Departamentos: Recibe una lista de los departamentos y el nuevo nodo departamento.

Agregar regiones (Se encarga de agregar regiones si es requerido): Recibe el nodo empresa.

Funciones dedicadas a eliminar algún nodo en las listas. Todas reciben el nodo empresa:

- Eliminar Suc: También recibe un entero.
- Eliminar Clientes.
- Eliminar departamento.
- Eliminar región.

Funciones Administradoras (Se encarga de gestionar todo el programa): Recibe el nodo empresa.

Funciones dedicadas a modificar o editar algún dato de los nodos en las listas. Todas reciben el nodo empresa.

- Modificar Sucursal: También recibe un entero.
- Modificar Departamento: También recibe un entero.
- Editar Clientes.

Funciones dedicadas a mostrar los datos en los nodos de las listas:

- Mostrar Sucursales: Recibe el nodo empresa.
- Mostrar datos suc: Recibe el nodo empresa y un entero.
- Mostrar Clientes: Recibe el puntero a sucursal para mostrar sus datos.
- Mostrar Departamento: Recibe el nodo empresa y un entero.
- Mostrar datos departamentos: Recibe el puntero a departamento y un entero.
- Mostrar Clientes: Recibe el nodo empresa.

Funciones dedicadas a guardar los datos de los nodos de las listas. Reciben el nodo empresa menos la función Guardar Reclamos:

- Guardar Datos Regiones.
- Guardar Datos.
- Guardar Datos Sucursales.
- Guardar Datos Clientes.
- Guardar Datos Departamento.
- Guardar Reclamos: Recibe un puntero a clientes y un entero.

Funciones dedicadas a buscar nodos en las listas:

- Buscar sucursal: Recibe la lista de sucursales y el entero y retorna un puntero sucursal.

- Buscar sucursal dep: Recibe la lista de sucursales, el carácter y retorna un puntero sucursal.
- Buscar Cliente: Recibe en nodo empresa, un entero y retorna un puntero cliente.

Funciones dedicadas cargar, validar, cambiar y generar. Reciben el nodo empresa menos la función Cargar Empresa, Validar rut, validar índice y Generar ID CONSULTA:

- Cargar Empresa: Retorna un puntero empresa.
- Cargar Clientes.
- Buscar Region: También recibe un entero y devuelve una posición.
- Validar usuario: Retorna un un1 o un 0.
- Validar rut: Recibe un carácter y retorna un 1 o 0.
- Validar clave: También recibe un entero y retorna un 1 o 0.
- Cambiar clave.
- Validación de índice: Retorna la posición.
- Generar ID CONSULTA: Retorna una posición.

Funciones Extras:

Cliente reclamon:

Departamento mejor evaluado:

4.4 Diagramas de Flujo

4.4.1 Agregar en el último nivel de Anidación

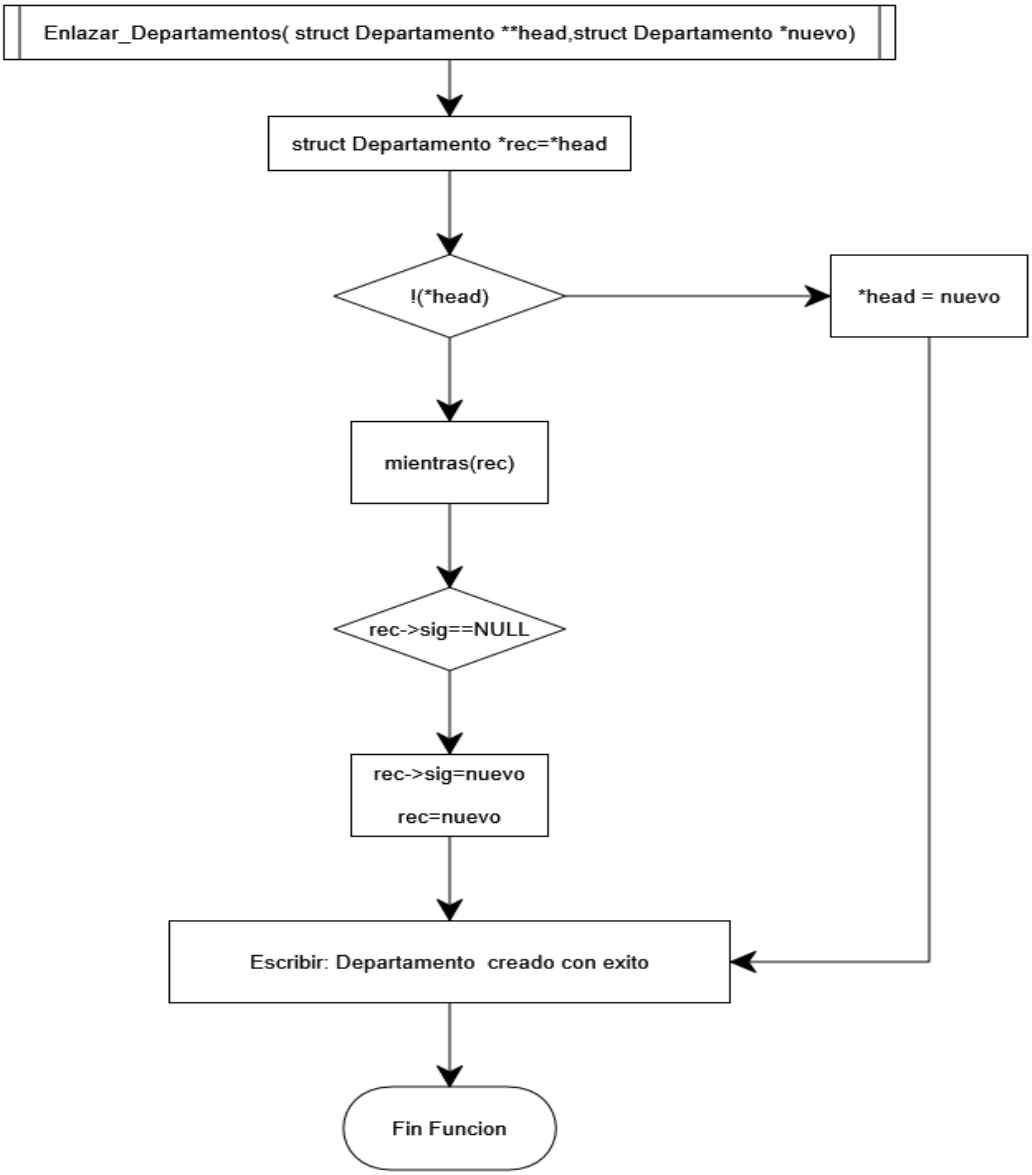


Figura Nº 10

En la Figura Nº 10 se muestra el diagrama de flujo de crear departamento donde se se recibe por parámetros la estructura Empresa y el nuevo departamento que queremos agregar.

4.4.2 Eliminación en el segundo nivel de Anidación

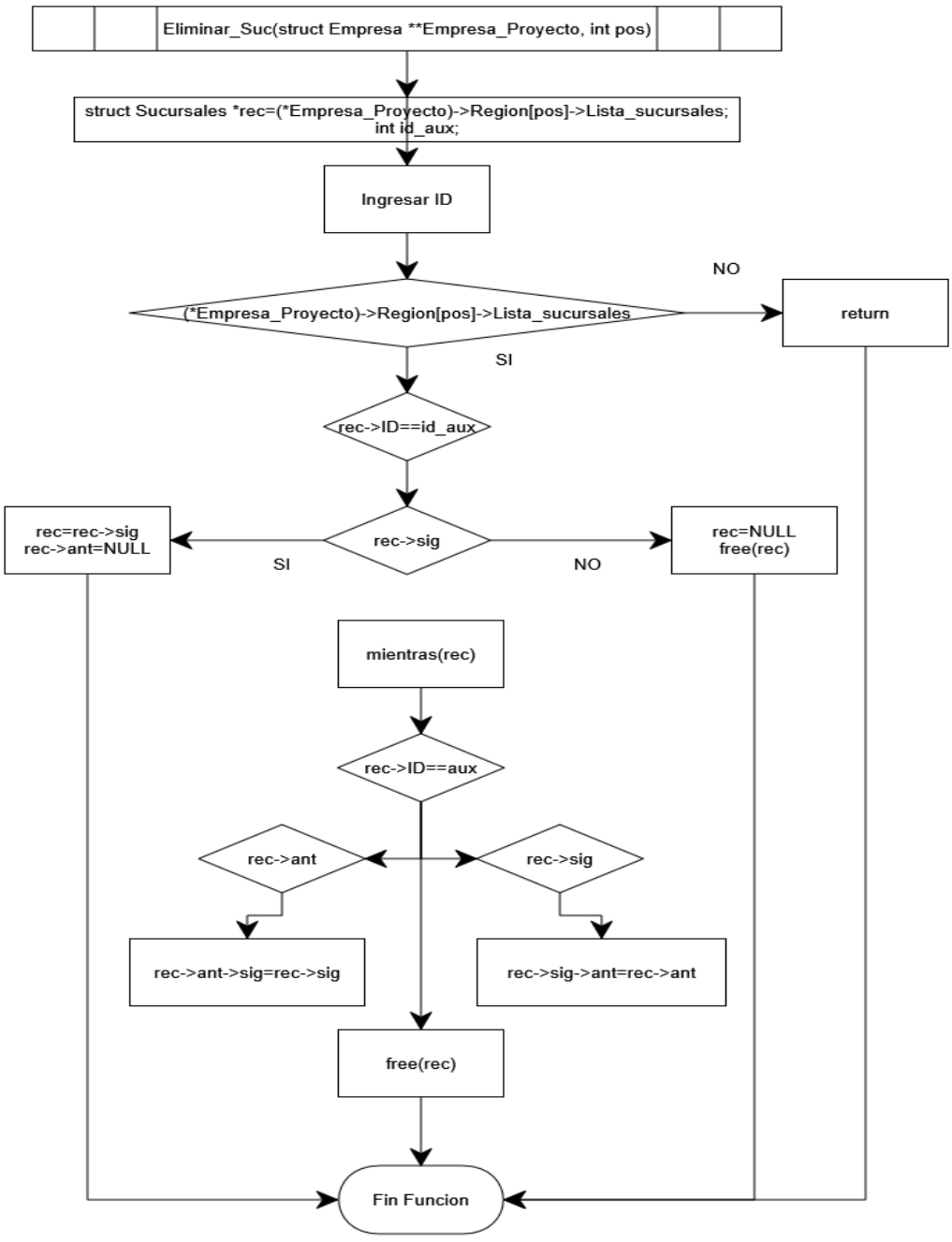


Figura Nº 11

En la Figura Nº 11 se muestra el diagrama de flujo de eliminar sucursal donde se recibe por parámetros la estructura Empresa y la posición de la región donde se encuentra la sucursal a eliminar.

4.4.3 Mostrar en el último nivel de anidación

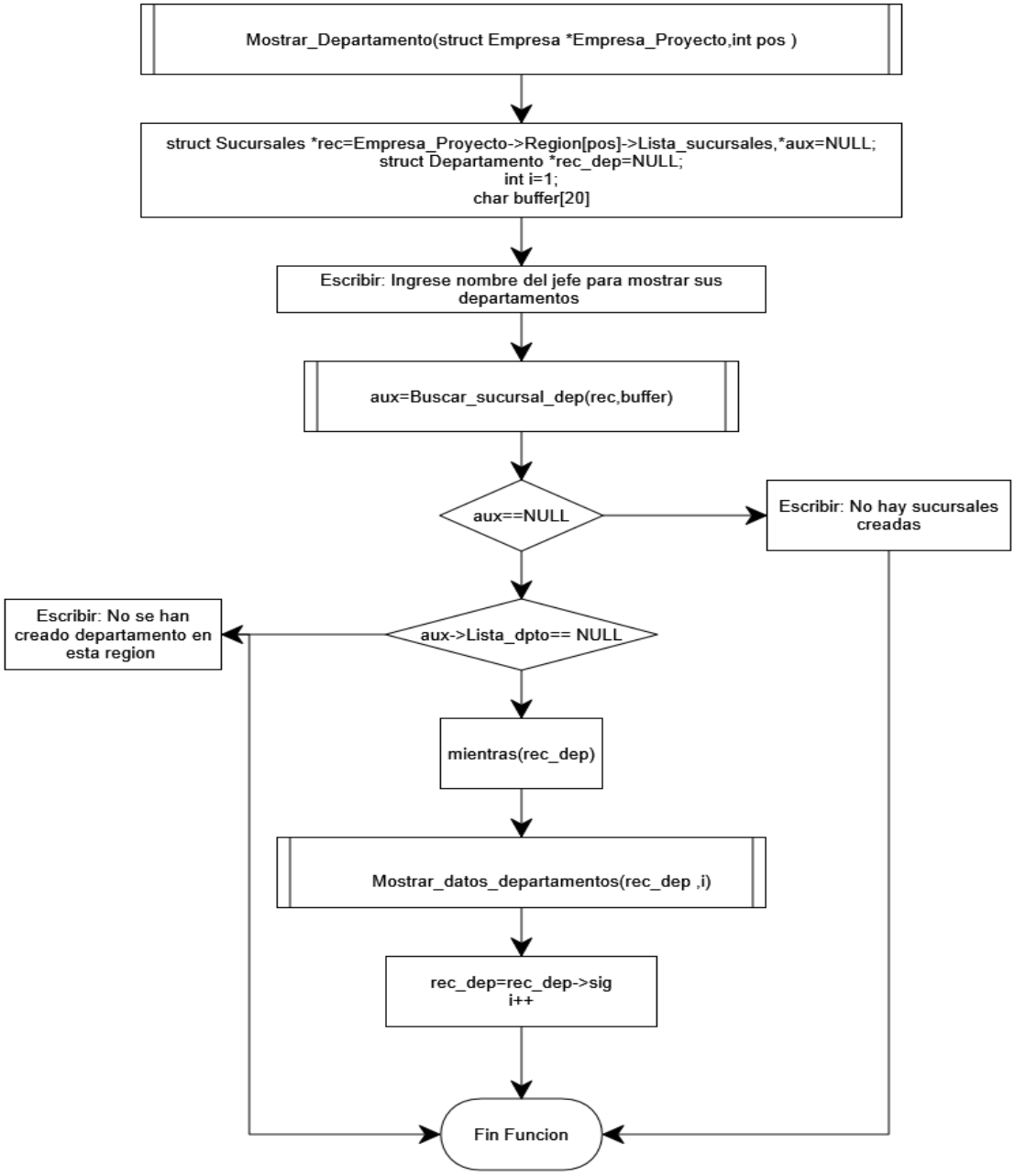
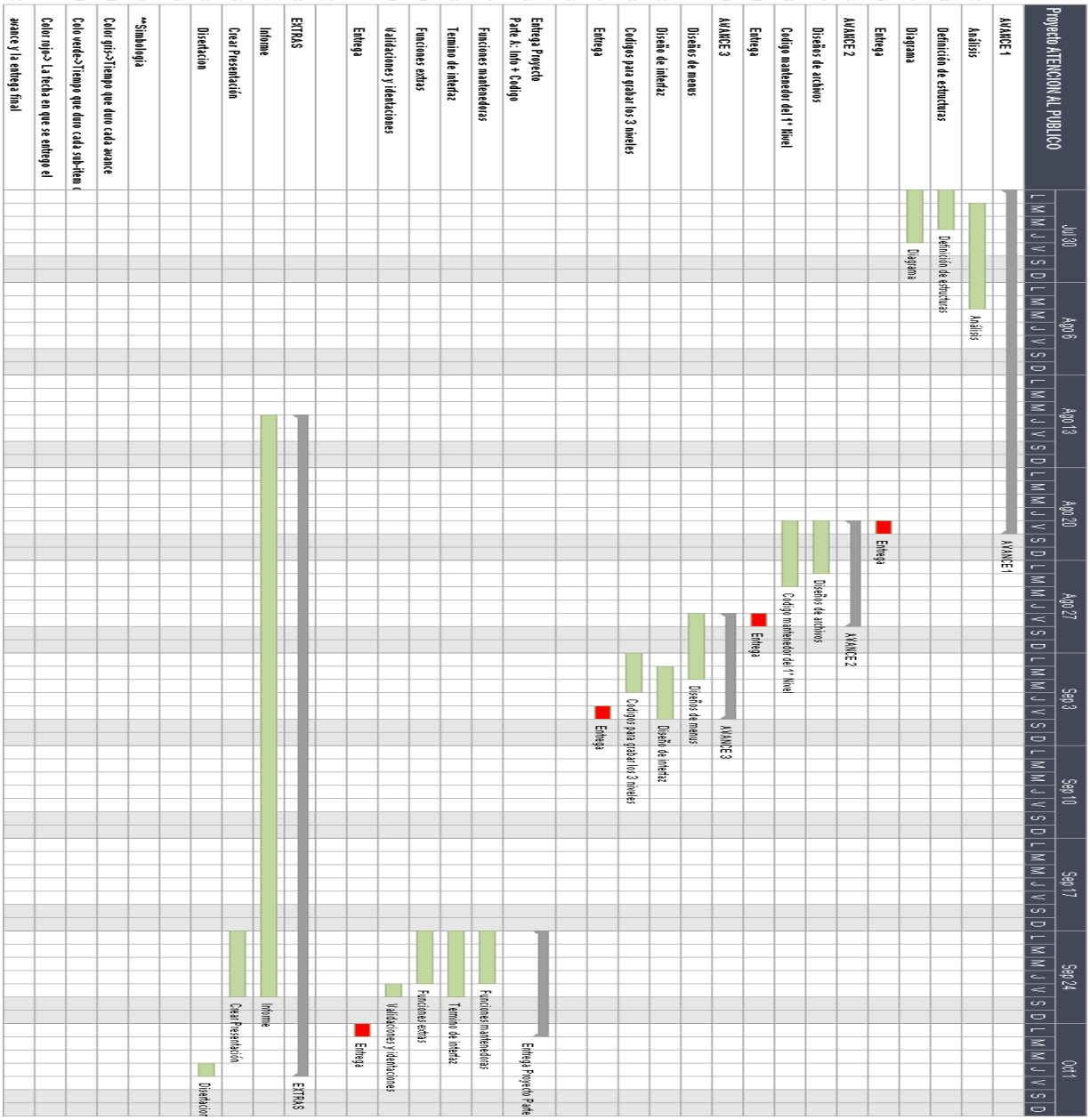


Figura Nº 12

En la Figura Nº 12 se muestra el diagrama de flujo de mostrar departamento donde se recibe por parámetros la estructura Empresa y la posición de la región donde se encuentra los departamentos a comprar.

5 Planificación

5.1 Tabla Gantt Grupal A



5.2 Tabla Gantt Grupal B

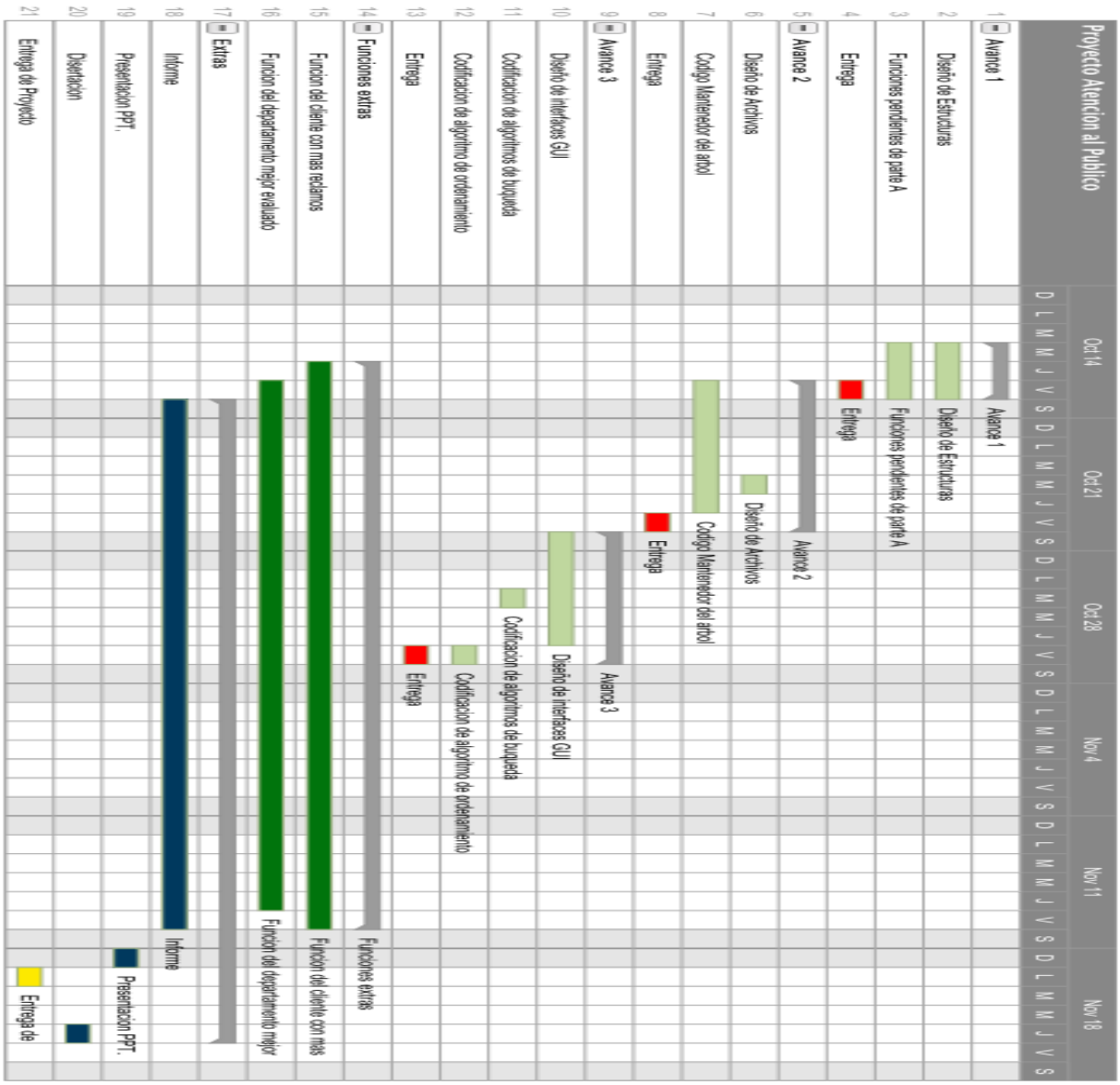


Figura Nº 13-14

La figura Nº 22, muestra el diagrama de Gantt en forma general, las tareas realizadas para llevar a cabo nuestro proyecto; las barras azules muestran los días utilizados en cierta tarea.

5.2 Tabla de Avance Individual A

Proyecto ATENCION AL PUBLICO	**Fecha de inicio**	**Fecha de finalización**	**Asignado a**	**Duración**
AVANCE 1	**30-07-12**	**24-08-12**	Grupo	**20**
Análisis	31-07-12	07-08-12	Grupo	6
Definición de estructuras	30-07-12	01-08-12	Grupo	3
Diagrama	30-07-12	02-08-12	Cristian	4
Entrega	24-08-12	24-08-12		1
AVANCE 2	**24-08-12**	**31-08-12**		6
Diseños de archivos	24-08-12	27-08-12	Peter,Cristian	2
Código mantenedor del 1° Nivel	24-08-12	28-08-12	Peter,David	3
Entrega	31-08-12	31-08-12		1
AVANCE 3	**31-08-12**	**07-09-12**		6
Diseños de menús	31-08-12	04-09-12	David,Cristian	3
Diseño de interfaz	04-09-12	07-09-12	Sergio,Peter	4
Códigos para grabar los 3 niveles	03-09-12	05-09-12	Peter,Cristian	3
Entrega	07-09-12	07-09-12		1
Entrega Proyecto Parte A: Info + Codigo	**24-09-12**	**01-10-12**		**6**
Funciones mantenedoras	24-09-12	27-09-12	Grupo	4
Termino de interfaz	24-09-12	28-09-12	Cristian	5
Funciones extras	24-09-12	27-09-12	Grupo	4
Validaciones y identaciones	28-09-12	28-09-12	Grupo	1
Entrega	01-10-12	01-10-12		1
EXTRAS	16-08-12	04-10-12		36
Informe	16-08-12	28-09-12	Sergio	32
Crear Presentación	24-09-12	28-09-12	Grupo	5
Disertacion	04-10-12	04-10-12	Grupo	1

5.3 Tabla de Avance Individual B

Proyecto Atencion al Publico	Fecha de inicio	Fecha de finalización	Duración	Asignado a
Avance 1	17-10-12	19-10-12	3	Grupo
Diseño de Estructuras	17-10-12	19-10-12	3	Cristian
Funciones pendientes de parte A	17-10-12	19-10-12	3	Peter,David
Entrega	19-10-12	19-10-12	1	Grupo
Avance 2	19-10-12	26-10-12	6	Grupo
Diseño de Archivos	24-10-12	24-10-12	1	Sergio
Codigo Mantenedor del arbol	19-10-12	25-10-12	5	Peter,Cristian
Entrega	26-10-12	26-10-12	1	Grupo
Avance 3	27-10-12	02-11-12	6	Grupo
Diseño de interfaces GUI	27-10-12	01-11-12	5	Cristian
Codificacion de algoritmos de buqueda	30-10-12	30-10-12	1	Peter
Codificacion de algoritmo de ordenamiento	02-11-12	02-11-12	1	Peter
Entrega	02-11-12	02-11-12	1	Grupo
Funciones extras	18-10-12	16-11-12	22	
Funcion del cliente con mas reclamos	18-10-12	16-11-12	22	Grupo
Funcion del departamento mejor evaluado	19-10-12	15-11-12	20	Grupo
Extras	20-10-12	22-11-12	25	
Informe	20-10-12	16-11-12	21	Grupo
Presentacion PPT.	18-11-12	18-11-12	1	Grupo
Disertacion	22-11-12	22-11-12	1	Grupo

6 Conclusión

Al terminar el proyecto Parte B nos dimos cuentas que existen otras estructuras de “listas” en las cuales se puede trabajar llamadas árboles que a través de funciones recursivas pueden facilitar el manejo de éstas en los códigos fuentes, dicho esto se nos facilitó un poco el trabajo.

Además con este parte B soltamos la mano para lo que se viene ya sea la Cátedra N° 2 y el Examen los cuales deberían hacérsenos más fácil o por lo menos poder saber qué hacer en algunos casos.

En conclusión pensamos que creamos un código bien completo en lo que se refiere a las necesidades de un cliente ya sea registrado en nuestro programa como externo ya que en los 2 casos pueden interactuar libremente por sus menús correspondientes ya sea buscando información como también agregando consultas y reclamos a los archivos de texto.

7 Anexo

7.1 Interfaz

Enfocados en entregar una interfaz relacionada con nuestro tema pusimos énfasis en las ventanas del programa para que el usuario sepa que hacer fácilmente sin la guía de terceros.

7.1.1Inicio.exe:

Da la bienvenida al programa el botón da comienzo.

Figura Nº 15
Ventana de bienvenida al sistema.

7.1.2 Terminal:

Espera que la ventana retorne un valor entero para ejecutar acciones mientras que muestra información de la empresa cargada. (Figura Nº 16)

Figura Nº 16
Terminal sistema esperando respuesta.

7.1.3 Menu.exe:

El programa está cargado contiene datos de la empresa algunos clientes regiones etc. Solo espera una acción. (Figura Nº 17)

Usuario Registrado: confirma la afiliación del cliente por medio de su ID de consulta, luego se muestra la ventana Clientes que se encarga de atenderlo.

Regístrese: Si no se posee de una ID de consulta el programa necesita agregarlo como usuario para poder atender sus necesidades, la ID se asigna automáticamente.

Administrador: Posee todas las funciones mantenedoras del sistema

INFO: Despliega un txt con información de ayuda.

SALIR: Finaliza el programa.

Figura Nº 17
Menú principal.

7.1.4 Clientes.exe:

Esta ventana es la encargada de atender al usuario crea un fichero de consultas y otro de reclamos, en estos se van almacenando junto con los datos del usuario (Rut, ID de consulta). (Figura Nº 26)

Figura Nº 18
Ventana clientes.

7.1.5 Terminal

Luego de escoger una opción la terminal captura el reclamo/consulta del usuario (Figura Nº 19)

Figura Nº 19
Ventana clientes.

7.1.6 Administrador.exe

Se en encarga de modificar los niveles de estructuras por medio de la funciones mantenedoras.

Figura Nº 20
Ventana Administrador.

7.1.7 Funciones extras:

1.-**Cliente con más reclamos** en donde con el estado de resultado inicializado en 0 cada vez q un cliente haga un reclamos ese estado aumentara en 1 en el código a través de un recorrido post orden y una búsqueda pudimos verificar quien es el cliente con más reclamos y mostrarlo por pantalla.

```
1  void cliente_reclamon(struct Empresa *Empresa_Proyecto){
2      struct Cliente *cliente_reclamon;
3      struct Cliente *tmp=Empresa_Proyecto->Arbol_clientes;
4              ////// postorden_cliente_reclamon(tmp, &cliente_reclamon);//////
5              if(tmp){
6              printf("el cliente : %s\n con %d reclamos ",cliente_reclamon->nombre,
7  cliente_reclamon->estado_reclamos);
8              printf("es el cliente con mas reclamos en toda la empresa\n");
9              }else{
10                 return;
11             }
12 }
13 void postorden_cliente_reclamon(struct Cliente *p, struct Cliente**cliente_reclamon){
14 struct pila *cima;
15 int may=-1;
16    printf("\n postorden_cliente_reclamon: \n");
17     cima=NULL;
18     push(&cima,NULL);
19
20         while(cima){
21             while(p){
22                 push(&cima,p);
23                 if(p->der){
24                     push(&cima, p->der);
25                     push(&cima, NULL);
26                 }
27                 p=p->izq;
28             }
29             p=pop(&cima);
30             while(p){
31                 if(p->estado_reclamos > may){
32                 may=p->estado_reclamos;
33                 *cliente_reclamon=p;
34                 }
35             p=pop(&cima);
36             }
37             if(!p){
38                 p=pop(&cima);
39             }}}
```

2.- **Departamento mejor evaluado** en donde a través de una variable llamada valoración inicializada en 10 cada vez q el cliente haga un reclamo se pedirá el nombre del departamento en el cual se querrá poner ese reclamo y así a la variable valoración se le restara 1 después a través de una función de buscar, a través del nombre de la sucursal se buscara en la lista de departamentos correspondientes el departamento mejor evaluado entre todos.

```
1  struct Departamento* mejorevaluado(struct Empresa *Empresa_Proyecto,int pos){
2  struct Sucursales *rec=(Empresa_Proyecto)->Region[pos]->Lista_sucursales;struct Departamento
3  *dep=rec->Lista_dpto,*mejor_evaluada=NULL, *aux=dep ;
4    if (dep==NULL){
5        return NULL;
6     }else{
7   while(dep)  {
8            if(aux->valoracion<=dep->valoracion){
9                 mejor_evaluada=dep;
10   }
11           printf("%d", mejor_evaluada->valoracion);
12           dep=dep->sig;
13        }
14    }
15
16    return mejor_evaluada;
17 }
```

Llamada:

```
242  case(2):
243  Usuario_rec=busqueda_clientes(Empresa_Proyecto->Arbol_clientes ,&aux_buscar, ID_cliente);
244                          flag=0;
245                          if( Usuario_rec == NULL ){
246                             if(flag == 0 ){
247                                printf("ID INVALIDA");
248                                system("pause");
249                                flag=1;
250                             }
251                          }else{
252                             pos=Pos()-1;
253                             fflush(stdin);
254                             printf("\n\t\20 INGRESE EL ID DE LA SUCURSAL \n");
255                             scanf("%d",&s);
256
257                             buscar=(Empresa_Proyecto->Region[pos])->Lista_sucursales;
258                             if(buscar){
259
260                                suc= Buscar_sucursal( buscar,s);
261                                if(suc==NULL){
262                                   return;
263                                }else{
264                                  printf("Ingrese nombre de depto donde quiere reclamar\n");
265                                  scanf("%s", buffer);
266                                  buscar_dep=buscar_depto(suc->Lista_dpto,buffer);
267                                  if(buscar_dep==NULL){
268                                  return;
269                                  }else{
270                                   buscar_dep->valoracion--;
271                                   Guardar_Reclamos( &Usuario_rec );
272                                   //Guardar_Datos_Clientes(Empresa_Proyecto );
273                                   guardar_clientes(Empresa_Proyecto
274                                  }
275                                }
276                             }
277                          }
278
```

Funcion búsqueda ABB :

```
1   struct Cliente* busqueda_clientes(struct Cliente *arbol, struct Cliente **ant, int ID){

2     int enc=0;
3     while(!enc && arbol){
4       if(arbol->ID_consulta==ID){
5         enc=1;
6       }else{
7         *ant=arbol;
8         if(ID<arbol->ID_consulta){
9           arbol=arbol->izq;
10        }else{
11          arbol=arbol->der;
12        }
13      }
14    }
15     return árbol;
```