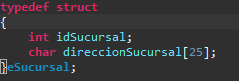
Alumno : Ian Sein

Division : 1°C

**Recuperatorio Primer Parcial Laboratorio**

**Estructura añadida**

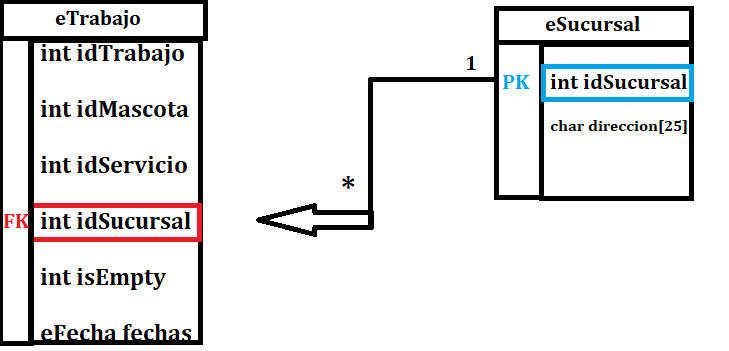
La estructura añadida, trae consigo la características de lugares, siendo más específico, las sucursales de la veterinaria.



Esta estructura nos permite organizar los trabajos solicitados dependiendo en la sucursal que haya sido solicitado el mismo de manera que se logre manejar y organizar de una forma más eficiente los datos si hay más veterinarias.

La relación que conlleva esta estructura con las otras es el poder organizar cada mascota, trabajo o servicio dentro de cada sucursal dependiendo el listado que se genere y hacer distintos cálculos dependiendo la sucursal, listas con los turnos de cada una, etc. Todo esto lleva a una mejor organización dentro de la veterinaria.

**Diagrama entidad relación**

****

**Enunciado de los informes anexados**

El primer informe permite la organización de ganancias en las distintas sucursales de la veterinaria. Muestra por cada sucursal, las mascotas con su respectivos nombres que se realizaran un servicio, mostrándolo, y las ganancias futuras que tendrá la veterinaria al terminar con los trabajo solicitados.

El segundo informe permite la organización de turnos.

Muestra por cada sucursal, las mascotas con sus respectivos nombres, el servicio que fue solicitado y el turno del trabajo (día, mes y año) además de generar un total de turnos entre todas las sucursales.

**Prototipo de las funciones con su respectiva documentación**

**void InicializarDatos(eTrabajo[], int);**

Esta función inicializa a todos los espacios del array de trabajo en un estado de “desocupado” (0), pasando como parámetros el array de trabajos y su tamaño.

**int BuscarLibre(eTrabajo[], int);**

Esta función busca dentro del array de trabajos, que haya un espacio “desocupado” (0),

pasando como parámetros su array y su tamaño, retornando el índice del espacio.

**void CargarTrabajoMascota(eTrabajo[], int, int, eServicio[], eMascota[], int, eSucursal[], int);**

Esta función recibe el índice del espacio “desocupado” (0), y carga un trabajo dentro del sistema en el índice que recibió, pidiendo los datos y cambiando el estado a “ocupado” (1).

**eTrabajo PedirDatosTrabajo(eServicio[], int, int, eTrabajo[], eMascota[], int, eSucursal[], int);**

Esta función pide al usuario todos los datos acerca del servicio, trabajo, mascota, sucursales, pasando como parámetro todos los arrays correspondientes a sus estructuras. Retorna la estructura de un trabajo con todos estos datos.

**int BuscarIndice(eTrabajo[], int, eServicio[], int, eMascota[],int, eSucursal[], int);**

Esta función nos permite buscar un ID ingresado por el usuario para corroborar si existe dentro de los trabajos. Retorna el índice si se encontró el ID y -1 si no se encontró.

**int DarDeBajaTrabajo(eTrabajo[], int, eServicio[], int, eMascota[], int, eSucursal[], int);**

Esta función da la baja a un trabajo buscando el ID ingresado por el usuario.

Si se pudo encontrar al trabajo se le pide nuevamente una confirmación al usuario sobre la baja del trabajo y luego se le cambia el estado a la posición a “desocupado”(0),

Si no se encontró el ID o el usuario no quiso, vuelve al menú. Retorna 1 si se pudo dar de baja y 2 si el usuario no quiso.

**void ModificarTrabajo(eTrabajo[], int, int, eServicio[], eMascota[], int, eSucursal[], int);**

Esta función permite la modificación de un servicio buscando el ID escogido por el usuario

Si se pudo encontrar el trabajo, se le hace nuevamente una confirmación al usuario para modificar el servicio y luego se modifica el servicio. Si no se encontró el ID o el usuario no quiso, vuelve al menú.

**void InicializarDatosMascota(eMascota[], int);**

Esta función inicializa a todos los espacios del array de mascota en un estado de “desocupado” (0), excepto a los primeros 5 ya que están hardcodeados. Pasando como parámetros el array de mascotas y su tamaño

**int BuscarLibreMascota(eMascota[], int);**

Esta función busca dentro del array de mascotas, que haya un espacio “desocupado” (0),

pasando como parámetros su array y su tamaño, retornando el índice del espacio.

**void CargarMascota(eMascota[], int);**

Esta función recibe el índice del espacio “desocupado” (0), y carga una mascota dentro del sistema en el índice que recibió, pidiendo los datos y cambiando el estado “ocupado” (1).

**eMascota PedirDatosMascota(int, eMascota[]);**

Esta función pide al usuario todos los datos acerca de la mascota, pasando como parámetro su array. Retorna la estructura de la mascota con todos estos datos.

**int BuscarIndiceMascota(eMascota[], int);**

Esta función nos permite buscar un ID ingresado por el usuario para corroborar si existe dentro de las ID 's de las mascotas. Retorna el índice si se encontró el ID y -1 si no se encontró.

**int DarDeBajaMascota(eMascota[], int, eTrabajo[], int);**

Esta función da la baja a una mascota buscando el ID ingresado por el usuario.

Si se pudo encontrar a la mascota se le hace nuevamente una confirmación al usuario y se le cambia el estado de la posición a “desocupado”(0). Si no se encontró el ID o el usuario no quiso la baja se vuelve al menú. Retorna 1 si se pudo dar de baja y 2 si el usuario no quiso.

**void ModificarMascota(eMascota[], int);**

Esta función permite la modificación del nombre de una mascota buscando el ID ingresado por el usuario. Si se pudo encontrar el ID de la mascota, se le hace nuevamente una confirmación al usuario sobre la modificación y luego ingresa el nuevo nombre. Si no se logró encontrar el ID o el usuario no quiso modificarlo se vuelve al menú.

**eMascota ObtenerMascotaPorID(eMascota[], eTrabajo, int);**

Esta función permite obtener una estructura de mascota en una posición específica mediante el índice que le pasemos por parámetro al trabajo. Retorna una estructura de mascota.

**eSucursal ObtenerSucursalPorID(eSucursal[], eTrabajo, int);**

Esta función permite obtener una estructura de sucursal en una posición específica mediante el índice que le pasemos por parámetro al trabajo. Retorna una estructura de sucursal.

**eServicio ObtenerServicioPorID(eServicio[], eTrabajo, int);**

Esta función permite obtener una estructura de servicio en una posición específica mediante el índice que le pasemos por parámetro al trabajo. Retorna una estructura de sucursal.

**void OrdenarListadoPorAnio(eTrabajo[], int, eMascota[], int);**

Esta función permite ordenar a todos los trabajos por año de manera ascendente, y si los años de los trabajos solicitados son iguales, ordena por nombre.

**void OrdenarNombreSegundoCriterio(eTrabajo[], eMascota[], int, int);**

Esta función permite ordenar por nombre a todos los trabajos.

**void ListadoServiciosConSusNombres(eTrabajo[], int, eServicio[], int, eMascota[], int);**

Esta función permite obtener un listado de cada servicio con todos los nombres de las mascotas dependiendo cual utilizó cada una.

**void PromedioDeEdadesPorServicio(eTrabajo[], int, eMascota [], int, eServicio[], int);**

Esta función permite obtener un listado del promedio de edad de las mascotas dependiendo qué servicio haya sido solicitado

**void ServiciosConMasTrabajos(eTrabajo[], int, eServicio[], int);**

Esta función permite obtener el o los servicios que más trabajos han tenido buscando al mayor o mayores y mostrandolos por pantalla.

**void GananciasTotalesSucursal(eTrabajo[], int,eServicio[], int, eMascota[], int, eSucursal[], int);**

Esta función permite obtener un listado de las mascotas que han solicitado un trabajo dentro de una sucursal, y muestra por pantalla las ganancias futuras que tendrá la sucursal de la veterinaria luego de terminar con los trabajos solicitados.

**void TurnosSucursales(eTrabajo[], int, eServicio[], int, eMascota[], int, eSucursal[], int);**

Esta función permite obtener un listado de las mascotas que han solicitado un trabajo dentro de una sucursal, el turno específico de cada una indicando día, mes y año, y los turnos totales en general de la veterinaria.

**int ObtenerPrecioTotal(eServicio[], eTrabajo[], int, int);**

Esta función nos permite obtener el precio total por los servicios prestados. Retorna un acumulador de precios.

**Enlace a vídeo explicativo**

<https://drive.google.com/drive/folders/16CSyfOCFMhAmdggwjoprWWQ3ETXJ2web>