**Desarrollo del trabajo de TFI:**

Como sabemos desde el punto de vista de la escritura de un algoritmo, el mismo se puede dividirse en tres partes:

* Cabecera del programa
* Sección de Declaraciones
* Cuerpo del programa

L a Cabecera del programa: Comienzan con el nombre del programa o palabras reservadas que el lenguaje especifica. En nuestro caso, estamos usando el lenguaje C, que explicita los denominados archivos de cabecera que se indica que archivos de inclusión se usarán en el programa a continuación.

**Utilizamos las directivas:**

* Stdio.h: Archivo cabecera que contiene las definiciones de las macros, constantes, declaraciones de funciones de la biblioteca estandar de lenguaje de lenguaje de programación C para hacer operaciones, estandar, de entrada y salida.
* Stdlib.h: el archivo de cabecera de la biblioteca estándar de propósito general del lenguaje de programación C. Contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras.
* -string.h: Es un archivo de la Biblioteca estándar del lenguaje de programación C que contiene la definición de macros, constantes, funciones y tipos y algunas operaciones de manipulación de memoria.
* ctype.h: es un archivo de cabecera de la biblioteca estándar del lenguaje de programación C diseñado para operaciones básicas con caracteres. Contiene los prototipos de las funciones y macros para clasificar caracteres.
* Locale.h: Define la configuración especifica de la ubicación, como los formatos de fecha y los símbolos de monedas.
* LibrerasTFI.h: Esta librería fue diseñada de manera propia, ya que son archivos o bloques de códigos con funciones especiales que podemos incluir en nuestro código importando sus funcionalidades.

Ahora escribimos, las llamadas de las funciones:

Funciones: Se denomina así, a un conjunto de tareas que se ejecutan cuando el programador determina (llamada a la función, incluido dentro del programa principal), y que no forma parte del programa principal. Luego de la ejecución de las mismas, vuelve al programa principal para proseguir con el resto de las acciones.

* void RegistrarVet(FILE \*Vet);
* int UsuarioRepetido(char NomU[10],FILE \*&Vet);
* void ComprobarUsuario(char NomU[10],int &b,FILE \*&Vet);
* void ComprobarContra(char ConU[32],int &b);
* void RegistrarUs(FILE \*Vet);

Luego colocamos la función MAIN, que es la que se debe encontrarse en todo programa escrito en lenguaje C sin excepción, define la función principal del programa, que es básicamente su cometido.

Sección de declaraciones de la función principal:

Int: Es del tipo numero entero, que almacena dentro de un rango desde -32767 hasta 32768.

setlocale(LC\_ALL,"spanish"): Nos permite escribir caracteres especiales derivados del español, tales como los acentos y la letra Ñ en particular.

FILE es un puntero a un archivo, que unifica la entrada salida asociado con la memoria buffer. Su definición es que es un puntero a una información que define varias cosas sobre él, incluyendo el nombre, el estado y la posición actual del archivo.

System color en lenguaje C: Para nuestro desarrollo del trabajo utilizamos para el diseño del sistema de gestión de datos un color de fondo aguamarina con el color del texto en blanco brillante.

Para el modulo veterinario: Tenemos varias opciones: Por lo tanto vamos a implementar un switch case, que nos permite desarrollar 3 casos:

* Iniciar Sesión: CASE 1Se emplea el condicional IF que significa mientras se cumpla el valor 1, se ejecutará la sesión. De lo contrario no iniciará sesión
* Visualizar lista de espera de turnos: CASE 2: Se emplea el IF- ELSE, IF mientras que sea 1, significa que se inició correctamente la sesión y mostrará la lista de espera, y emplearemos ELSE cuando no es equivalente a 1 no se inició sesión y debe reintentar.
* Registrar evolución de la mascota: CASE 3: Mientras que el condicional IF se equivalente a 1, va a iniciar sesión y va a mostrar la evolución de la mascota, en caso de no ser correcto mostrará ERROR.
* Cerrar Menú: Si seleccionamos 4, terminará la ejecución del programa
* Ingrese una opción: Aquí el usuario deberá digitar una de las opciones, del 1 al 4 y ejecutará lo que desee presionar.

**void listaEspera(FILE \*Us)**

Variables tipo

Int band=1, declaramos el valor 1 como bandera

También declaramos las demás como del tipo entero a las demás.

Colocamos un while que su condición mientras que la bandera equivale a 1, ejecuta, en nuestro caso:

Para que el usuario coloque la fecha del turno, en DIA- MES y AÑO, cada una con sus respectivas condiciones IF que están anidadas en sí misma.

DIA: Tiene que ser un día mayor que 0 y menor que 32.

MES: Mes mayor que 0 y menor que 13

AÑO: Mayor 2019 y menor 2300

Utilizamos un fopen() para abrir un archivo de escritura, entonces cualquier archivo con el mismo nombre se borrará y creará uno nuevo. Si no existe un archivo con el mismo nombre, se creará. La apertura de un archivo para las operaciones de lectura requiere que exista un archivo, si no existe, fopen() devolverá un error. En el programa utilizaremos la terminología fopen (“Turnos.dat”,”rb”)

NULL: Esta definida en STDIO.H este método detectará cualquier error al abrir un archivo, ejemplo: disco lleno o protegido contra escritura antes de comenzar a escribir en él. En nuestra aplicación en el programa, mostrará un error en el que no se encontramos turnos.

En caso de que no detecte ningún error en la sentencia IF, pasará a l ELSE:

Utilizamos el fread (), permite leer datos de un archivo abierto a través de la función fopen indicada con el puntero de lectura/escritura.

Utilizaremos un ciclo while mientras que sea distinto de la función feof, donde la declaración feof (); indicará si se ha llegado al final de un fichero, ejecutará.

Pediremos al usuario que ingrese su matrícula, una vez que ingresó la matricula, abrirá un archivo de escritura denominado “Veterinario.dat”, seguirá la misma lógica que el archivo fopen anterior, usaremos un if en caso de que haya algún error en la verificación, de no ser así prosigue con la ejecución de este.

Siguiendo la ejecución:

En el fread (), permite leer datos de un archivo abierto a través de la función fopen indicada con el puntero de lectura/escritura.

Dentro del fread, vamos a utilizar la función sizeof(), recibe como único parámetro y devuelve su tamaño en bytes, en nuestro caso práctico vamos a utilizarlo con tipo de datos estructurados, que correspondería a “DatVet” (Datos del veterinario).

Colocamos un while, mientras que ¡feof(aux), que indiciará si ha llegado al final del fichero aux, va a ejecutarlo. Dentro del while definimos un if que su la variable de la “matriAux.Mat debe ser equivalente a matri” en caso de ser correcto, utilizara la función fseek, que sitúa el puntero de lectura/escritura de un archivo en la posición indicada. En caso de no ser correcto nos mostrará por pantalla al usuario Matricula incorrecta.

Usamos el fread, para leer datos del archivo abierto en ese momento, en este caso de los turnos. Utilizaremos bandera equivalente a 1, que permitirá mostrar por pantalla al usuario que no hay turnos disponibles, y el nombre fclose () cerrará la secuencia que fue abierta mediante una llamada a fopen, escribe toda la información que se encuentra en el buffer del disco y realiza un cierre formal del archivo a nivel de sistema operativo. Si llegará a haber un error puede generar cualquier tipo de problemas, incluyendo la perdida de datos, destrucción de archivos y posibles errores intermitentes en el programa.

En nuestro caso, tenemos declarado que Us==NULL, us es equivalente a null, significa que la operación de cierre ha tenido éxito.

Utilizamos una serie de ELSE, en caso mencionado más arriba

DIA: Tiene que ser un día mayor que 0 y menor que 32.

MES: Mes mayor que 0 y menor que 13

AÑO: Mayor 2019 y menor 2300

En caso de no cumplir la condiciones, cada uno de manera individual sea cual fuere el error lo mostrará al usuario y preguntará si desea volver a ingresar nuevamente la fecha o regresar al menú.

Utilizamos la sentencia IF con la condición de que bandera es equivalente a CERO (0).

Vamos a utilizar fopen para abrir el archivos de “Turnos.dat”, que son los archivos de turnos, y como explicamos anteriormente en caso de que US==NULL (US equivale a NULL, arrojará un error y mostrará al usuario por pantalla, ya que hubo alguna falla en la verificación previo a abrir.

En caso de ser un else:

Utilizamos un fread que permite leer datos de un archivo abierto a través de la función fopen indicada con el puntero de lectura/escritura, ya que utilizamos la función fopen correspondiente a Turnos.dat, y también usamos la función sizeof(), recibe como único parámetro y devuelve su tamaño en bytes, en nuestro caso práctico vamos a utilizarlo con tipo de datos estructurados, que correspondería a “Turnos” (Turnos para mascotas).

Usamos el IF con la condición de que la matri == turnAux.MatV, significa que deben ser equivalente, vamos a declarar como auxiliar a la función fopen que abrir el archivo de “Mascotas.dat”, y nuevamente utilizamos como if aux==NULL, mostrará al usuario que no hay ninguna mascota registrada. Usamos un else si detecta que hay Mascotas registradas, por lo tanto usamos el fread para que lea los datos del archivo abierto en este momento y el sizeof para que nos devuelva su tamaño en bytes.

Usamos el ¡feof (aux) que nos indicará si llegó al final del fichero, para su posterior ejecución.

Pediremos al usuario, en este caso: DATOS DEL TURNO

* Nombre de la mascota
* Edad de la mascota en partes:

1. Año
2. Meses
3. Días

* Peso de la mascota
* Localidad
* DNI del dueño/propietario.
* Evolución registrada de la mascota: Breve descripción de parte del personal interno para quedar registrado momento a momento como va su evolución.

Luego utilizamos el fread nuevamente para leer los datos ingresados del archivos, y por ultimo utilizar un fclose() cerrará la secuencia que fue abierta mediante una llamada a fopen, escribe toda la información que se encuentra en el buffer del disco y realiza un cierre formal del archivo a nivel de sistema operativo. Si llegará a haber un error puede generar cualquier tipo de problemas, incluyendo la perdida de datos, destrucción de archivos y posibles errores intermitentes en el programa.

En nuestro caso, tenemos declarado que sea Us

**void evolucionMascota(FILE \*Us)**

Declaramos como variables

Char: Permite ingresar caracteres. Esta comprendido en rango 127 (Positivo y negativo)

FILE: es un puntero a un archivo, que unifica la entrada salida asociado con la memoria buffer. Su definición es que es un puntero a una información que define varias cosas sobre él, incluyendo el nombre, el estado y la posición actual del archivo.

INT: Variables del tipo entero, permite ingresar números enteros y está comprendido dentro de un rango: -32.767 hasta 32.768

Vamos a usar fopen, para abrir el archivo “Mascotas.Dat”, ya explicamos anteriormente como es el funcionamiento.

En este caso el ELSE, le dirá al usuario que digite el nombre de la mascota a registrar.

Usamos los comandos:

\_flushall: Elimina todos los búferes asociados con los flujos de entradas y escribe los búferes asociados con los flujos de salida.

Gets: Recibe como único argumento la variable en la que debe almacenarse una entrada de usuario que va a ser considerada como cadena de caracteres.

Puts: Recibe un único argumento: la cadena de caracteres a mostrar por pantalla

Strupr: Conversión de letras minúscula de la cadena en letras mayúsculas, siempre de acuerdo a la categoría LC-CTPYE que definimos anteriormente como idioma ESPAÑOL, para que nos permitiera utilizar los acentos y le letra Ñ.

Strcmp: Realiza una comparación del contenido de las cadenas, a partir del primer carácter de la cadena y continua con los siguientes caracteres hasta que los caracteres que corresponden difieren o alcanzan el final del mismo. En este caso debe ser equivalente a CERO (0).

Declaramos en esta llamada como aux a que abra el archivo “Turnos.dat”, la lógica del funcionamiento ya la explicamos anteriormente, ahora vamos a explayar, cuando ELSE

Fread: Utilizamos el fread (), permite leer datos de un archivo abierto a través de la función fopen indicada con el puntero de lectura/escritura.

Luego vamos a usar el condicional while (mientras que: haya llegado al final de dicho archivo comenzará su posterior ejecución de la variable AUX.

Usamos el IF que establecemos como condición: auxM.DNID==auxT.DNIT en caso de ser así, permitirá al usuario digitar la evolución de la mascota, y luego el comando flushall para eliminar el buffer del flujo de entrada asociado y escribirá los búferes asociados con los flujos de salida.

Utilizamos también el gets, que está en caracteres, para desarrollar una breve reseña de la evolución de la mascota.

Frwrite: Permite escribir datos de un archivo con la función ya empleada anteriormente en la llamada fopen, a partir de la posición indicada por el puntero de lectura/escritura.

Fseek: Realiza la ubicación del puntero de lectura/escritura de un archivo en la posición indicada.

Una vez realizada la breve descripción del paciente el programa mostrará que quedó grabado y el paciente pasa a estar en condición de ATENDIDO.

En caso de que sea ELSE: Utilizaremos bandera igual a 1.

Luego ya cerramos mediante la función fclose del archivo aux.

Ya a lo último, utilizamos un IF bandera equivalente a CERO, mostrará al usuario mediante un printf, como “ERROR, la mascota no se encuentra registrada”

Y luego usamos otra sentencia else if con la condición bandera equivalente a 1, mostrara mediante pantalla al usuario “ERROR, no hay turno registrado, por favor regístrese”, y cerramos mediante fclose al archivo “Us”