Índice

1. MÓDULOS

1.1. TIPOS

1.1.1. Estructuras

1.1.1.1 Incidencias

int	Id_viaje
int	Id_us_registra
int	Id_us_incidencia
char*	Desc_indicencia
int	Est_incidencia

1.1.1.2 Pasos

int	Id_viaje
char*	Poblacion

1.1.1.3 Usuarios

int	Id_usuario
char*	Nomb_usuario
char*	Localidad
int	Perfil_usuario
char*	User
char*	Login
int	Estado

1.1.1.4 Vehiculos

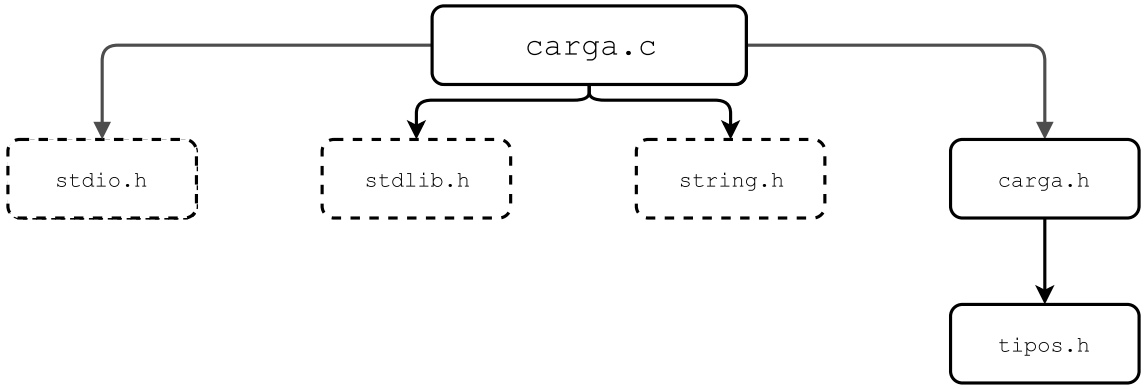
char*	Id_mat
int	Id_usuario
int	Num_plazas
char*	Desc_veh

1.1.1.5 Viajes

int	Id_viaje
char*	Id_mat
char*	F_inic
char*	H_inic
char*	H_fin
int	Plazas_libre
int	Viaje
float	Importe
int	Estado

1.2. CARGA

1.2.1. Gráfico de Dependencia



1.2.2. Funciones

```
static int  estadoUsuario(char** c);
static int  perfilUsuario(char** c);
static Usuarios*  initUsuarios(int* n);
static Vehiculos*  initVehiculos(int* n);
static Viajes*  initViajes(int* n);
static Pasos*  initPasos(int* n);
static Incidencias*  initIncidencias(int* n);
static int  idaVuela(char** c);
static int  estadoViaje(char** c);
static int  estadoIncidencia(char** c);
void  cargar(Usuarios** usuarios,
Vehiculos** vehiculos,
Viajes** viajes,
Pasos** pasos,
Incidencias** incidencias,
int* tam);
```

1.2.3. Definiciones

- ```
  void cargar(Usuarios** pUsuarios,
              Vehiculos** pVehiculos,
              Viajes** pViajes,
              Pasos** pPasos,
              Incidencias** pIncidencias,
              int* pInt);
  ```

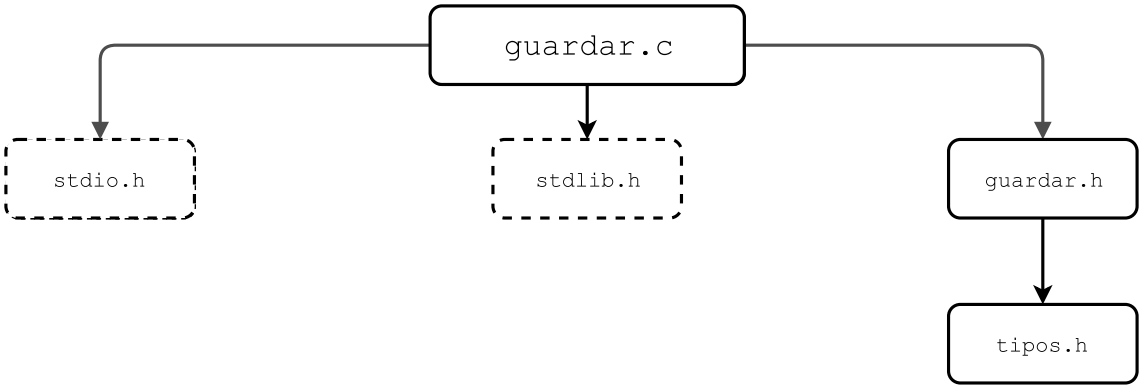
  - **Parametros**
    - `pUsuarios` → Puntero a puntero de usuarios.
    - `pVehiculos` → Puntero a puntero de vehiculos.
    - `pViajes`→ Puntero a puntero de viajes.
    - `pPasos`→ Puntero a puntero de pasos.
    - `pIncidencias`→ Puntero a puntero de incidencias.
    - `pInt`→ Puntero a vector para almacenar el tamaño de las estructuras.
  ```

- `int estadoIncidencia(char** c);`
  - **Parametros**
    - `c` → Recibe una cadena de caracteres por referencia.
  - **Devuelve**
    - 0 si `c` = `'cerrada'`
    - 1 si `c` = `'abierta'`
    - 2 si `c` = `'validada'`
- `int estadoUsuario(char** c);`
  - **Parametros**
    - `c` → Recibe una cadena de caracteres por referencia.
  - **Devuelve**
    - 0 si `c` = `'bloqueado'`
    - 1 si `c` = `'activo'`
- `int estadoViaje(char** c);`
  - **Parametros**
    - `c` → Recibe una cadena de caracteres por referencia.
  - **Devuelve**
    - 0 si `c` = `'cerrado'`
    - 1 si `c` = `'abierto'`
    - 2 si `c` = `'iniciado'`
    - 3 si `c` = `'finalizado'`
    - 4 si `c` = `'anulado'`
- `int idaVuelta(char** c);`
  - **Parametros**
    - `c` → Recibe una cadena de caracteres por referencia.
  - **Devuelve**
    - 0 si `c` = `'vuelta'`
    - 1 si `c` = `'ida'`
- `Incidencias* initIncidencias(int* n);`
  - **Parametros**
    - `n` → Referencia a la posición del vector que almacena el número de incidencias.
  - **Devuelve**
    - Un vector con los datos contenidos en el fichero **Incidencias.txt**.
- `Pasos* initPasos(int* n);`
  - **Parametros**
    - `n` → Referencia a la posición del vector que almacena el número de pasos.
  - **Devuelve**
    - Un vector con los datos contenidos en el fichero **Pasos.txt**.

- `Usuarios* initUsuarios(int* n);`
  - **Parametros**
    - `n` → Referencia a la posición del vector que almacena el número de usuarios.
  - **Devuelve**
    - Un vector con los datos contenidos en el fichero **Usuarios.txt**.
- `Vehiculos* initVehiculos(int* n);`
  - **Parametros**
    - `n` → Referencia a la posición del vector que almacena el número de vehiculos.
  - **Devuelve**
    - Un vector con los datos contenidos en el fichero **Vehiculos.txt**.
- `Viajes* initViajes(int* n);`
  - **Parametros**
    - `n` → Referencia a la posición del vector que almacena el número de viajes.
  - **Devuelve**
    - Un vector con los datos contenidos en el fichero **Viajes.txt**.
- `int perfilUsuario(char** c);`
  - **Parametros**
    - `c` → Recibe una cadena de caracteres por referencia.
  - **Devuelve**
    - 0 si `c` = `'administrador'`
    - 1 si `c` = `'usuario'`

1.3. GUARDAR

1.3.1. Gráfico de Dependencia



1.3.2. Funciones

```
static void  saveUsuarios(int n,Usuarios* usuarios);
static void  saveVehiculos(int n,Vehiculos* vehiculos);
static void  saveViajes(int n,Viajes* viajes);
static void  savePasos(int n ,Pasos* pasos);
static void  saveIncidencias(int n,Incidencias* incidencias);
       void  guardar(Usuarios** pUsuarios,
                     Vehiculos** pVehiculos,
                     Viajes** pViajes,
                     Pasos** pPasos,
                     Incidencias** pIncidencias,
                     int* pInt);
```

1.3.3. Definiciones

- ```
  void guardar(Usuarios** pUsuarios,
  Vehiculos** pVehiculos,
  Viajes** pViajes,
  Pasos** pPasos,
  Incidencias** pIncidencias,
  int* pInt);
  ```

  - **Parametros**
    - `pUsuarios` → Puntero a puntero de usuarios.
    - `pVehiculos` → Puntero a puntero de vehiculos.
    - `pViajes`→ Puntero a puntero de viajes.
    - `pPasos`→ Puntero a puntero de pasos.
    - `pIncidencias`→ Puntero a puntero de incidencias.
    - `pInt`→ Puntero a vector para almacenar el tamaño de las estructuras.
  ```

- `int saveIncidencias(int n,Incidencias* incidencias);`
  - **Parametros**
    - `n` → Tamaño del vector incidencias.
    - `incidencias` → Puntero del vector incidencias.
  - **Procedimiento**
    - Guarda la el vector de incidencias en el fichero **Incidencias.txt** y libera la memoria.
- `int savePasos(int n,Pasos* pasos);`
  - **Parametros**
    - `n` → Tamaño del vector pasos.
    - `pasos` → Puntero del vector pasos.
  - **Procedimiento**
    - Guarda la el vector de pasos en el fichero **Pasos.txt** y libera la memoria.
- `int saveUsuarios(int n,Usuarios* usuarios);`
  - **Parametros**
    - `n` → Tamaño del vector usuarios.
    - `usuarios` → Puntero del vector usuarios.
  - **Procedimiento**
    - Guarda la el vector de usuarios en el fichero **Usuarios.txt** y libera la memoria.
- `int saveVehiculos(int n,Vehiculos* vehiculos);`
  - **Parametros**
    - `n` → Tamaño del vector vehiculos.
    - `vehiculos` → Puntero del vector vehiculos.
  - **Procedimiento**
    - Guarda la el vector de vehiculos en el fichero **Vehiculos.txt** y libera la memoria.
- `int saveViajes(int n,Viajes* viajes);`
  - **Parametros**
    - `n` → Tamaño del vector viajes.
    - `viajes` → Puntero del vector viajes.
  - **Procedimiento**
    - Guarda la el vector de viajes en el fichero **Viajes.txt** y libera la memoria.