

Índice

1. Estructura de Datos	2
1.1. Incidencias	2
1.2. Pasos	2
1.3. Pasos	2
1.4. Usuarios	2
1.5. Vehiculos	2
1.6. Viajes	2
1.7. vIncidencias	2
1.8. vUsuarios	2
1.9. vVehiculos	2
1.10. vViajes	2
2. ACCESO	3
2.1. Gráfico de dependencias	3
2.2. Funciones	3
2.3. Definiciones	3
3. CARGAR	4
3.1. Gráfico de dependencias	4
3.2. Funciones	4
3.3. Definiciones	4
4. MENU	5
4.1. Gráfico de dependencias	5
4.2. Funciones	5
4.3. Definiciones	5
5. VIAJES	6
5.1. Gráfico de dependencias	6
5.2. Estructura de Datos	6
5.3. Funciones	6
5.4. Definiciones	7
6. USUARIOS	10
6.1. Gráfico de dependencias	10
6.2. Estructura de Datos	10
6.3. Funciones	10
6.4. Definiciones	11
7. UTILIDADES	13
7.1. Gráfico de dependencias	13
7.2. Funciones	13
7.3. Definiciones	13
8. Test	16
8.1. Test Viajes	16
8.1.1. Diagrama de flujo	16
8.2. Test Viajes	17
8.2.1. Diagrama de flujo	17
8.3. Test Viajes	17
8.3.1. Diagrama de flujo	17

1. Estructura de Datos

1.1. Incidencias

```
int    Id_viaje
int    Id_us_registra
int    Id_us_incidencia
char*  Desc_incidencia
int    Est_incidencia
```

1.2. Pasos

```
int    Id_viaje
int    Id_viajero
```

1.3. Pasos

```
int    Id_viaje
char*  Poblacion
```

1.4. Usuarios

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

1.5. Vehiculos

```
char*  Id_mat
int    Id_usuario
int    Num_plazas
char*  Desc_veh
```

1.6. 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
```

1.7. vIncidencias

```
Incidencias* inci
int          tam
```

1.8. vUsuarios

```
Usuarios* user
int       tam
```

1.9. vVehiculos

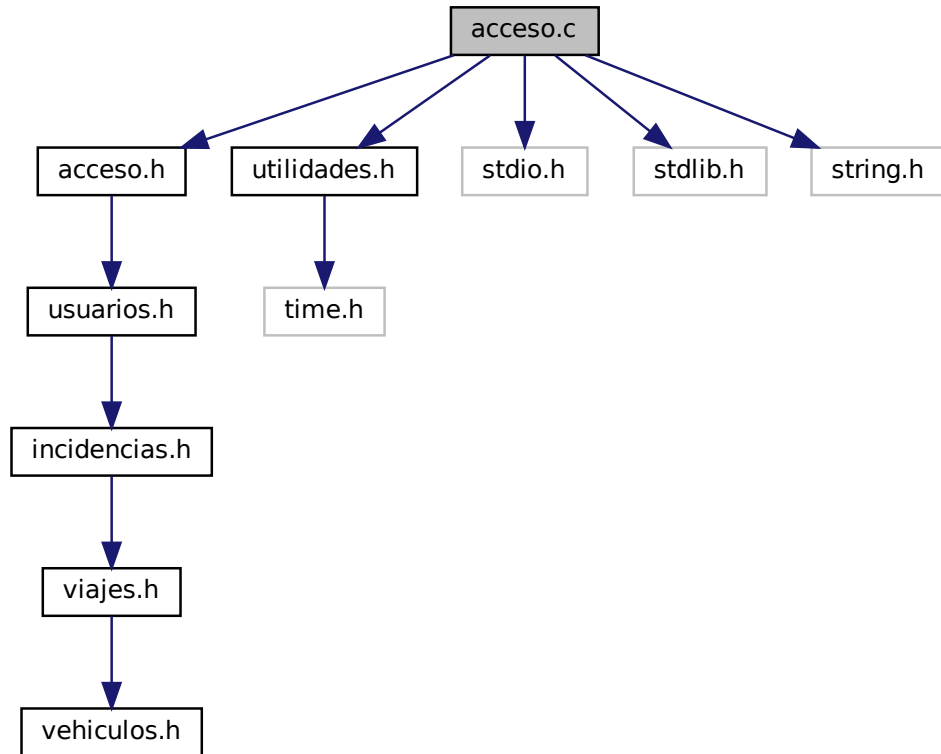
```
Vehiculos* vehi
int       tam
```

1.10. vViajes

```
Pasajeros* pasaj
Pasos*     pasos
Viajes*    viajes
int        tam_pj
int        tam_p
int        tam_v
int        last
```

2. ACCESO

2.1. Gráfico de dependencias



2.2. Funciones

- `int *acceder(vUsuarios *usuarios)`

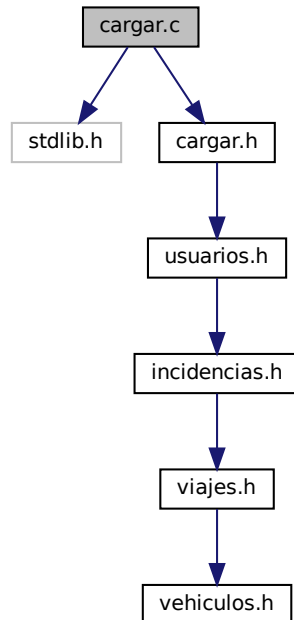
2.3. Definiciones

- `int *acceder(vUsuarios *usuarios)`

- **Descripcion**
 - Comprueba el tipo de usuario (usuario/administrador).
- **Parametros**
 - `usuarios` → Referencia al vector user.
- **Devuelve**
 - Índice posición del usuario / -1 si no lo ha encontrado posición 0.
 - Tipo usuario (0 (admin) / 1 (usuario)) en posición 1.

3. CARGAR

3.1. Gráfico de dependencias



3.2. Funciones

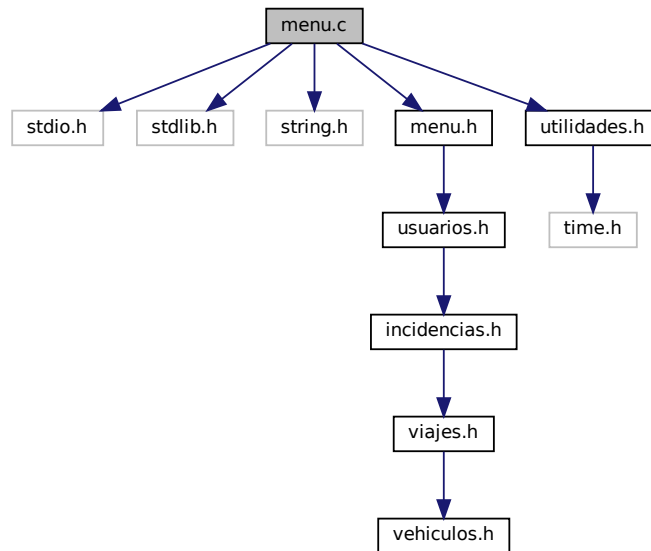
- `void init(vUsuarios* vu, vIncidencias* vi, vViajes* vv, vVehiculos* vve)`
- `void save(vUsuarios* vu, vIncidencias* vi, vViajes* vv, vVehiculos* vve)`

3.3. Definiciones

- `void init(vUsuarios* vu, vIncidencias* vi, vViajes* vv, vVehiculos* vve)`
 - **Descripcion**
 - Inicializa los vectores de usuarios, incidencias, viajes y vehiculos.
 - **Parametros**
 - `v` → Referencia al vector user.
 - `vi` → Referencia al vector inci.
 - `vv` → Referencia al vector viajes.
 - `vve` → Referencia al vector vehi.
- `void save(vUsuarios* vu, vIncidencias* vi, vViajes* vv, vVehiculos* vve)`
 - **Descripcion**
 - Guarda datos de usuarios, incidencias, viajes y vehiculos en ficheros y libera memoria.
 - **Parametros**
 - `v` → Referencia al vector user.
 - `vi` → Referencia al vector inci.
 - `vv` → Referencia al vector viajes.
 - `vve` → Referencia al vector vehi.

4. MENU

4.1. Gráfico de dependencias



4.2. Funciones

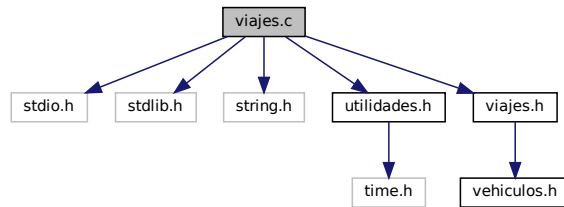
- `void menuUser(vUsuarios *vu, vIncidencias *vi, vViajes *vv, vVehiculos *vve, int indexusuario)`
- `void save(vUsuarios* vu,vIncidencias* vi,vViajes* vv,vVehiculos* vve)`

4.3. Definiciones

- `void init(vUsuarios* vu,vIncidencias* vi,vViajes* vv,vVehiculos* vve)`
 - **Descripcion**
 - Muestra menu principal para usuarios.
 - **Parametros**
 - `v` → Referencia al vector user.
 - `vi` → Referencia al vector inci.
 - `vv` → Referencia al vector viajes.
 - `vve` → Referencia al vector vehi.
 - `indexusuario` → Index del usuario.
- `void save(vUsuarios* vu,vIncidencias* vi,vViajes* vv,vVehiculos* vve)`
 - **Descripcion**
 - Muestra menu principal para administradores.
 - **Parametros**
 - `vu` → Referencia al vector user.
 - `vi` → Referencia al vector inci.
 - `vv` → Referencia al vector viajes.
 - `vve` → Referencia al vector vehi.
 - `indexadmin` → Index del admin.

5. VIAJES

5.1. Gráfico de dependencias



5.2. Estructura de Datos

- `struct Viajes`
- `struct Pasos`
- `struct Pasajeros`
- `struct vViajes`

5.3. Funciones

- `Viajes* initViajes(int* n)`
- `Pasos* initPasos(int* n)`
- `Pasajeros* initPasajeros(int* n)`
- `void publicarViajeUsuario(vViajes* v, vVehiculos* ve, int userId)`
- `void editarViajesUsuario(vViajes* v, vVehiculos* ve, int userId)`
- `void incorporarseViaje(vViajes* v)`
- `void detalleViaje(vViajes* v)`
- `void cancelarViaje(vViajes *v ,int Id_usuario)`
- `void publicarViajeAdmin(vViajes* v, vVehiculos* ve)`
- `void eliminarViajesAdmin(vViajes* v)`
- `void modificarViajesAdmin(vViajes* v, vVehiculos *vve)`
- `void listarViajesAdmin(vViajes* v)`
- `void saveViajes(int n, Viajes* viajes)`
- `void savePasos(int n, Pasos* pasos)`
- `void savePasajeros(int n, Pasajeros* pasaj)`
- `int buscarIndexViajes(vViajes* v, int id_viaje)`
- `int buscarIndexPasejeros(vViajes *v, int viaje, int viajero)`
- `void listarViajesAbiertos(vViajes* v)`
- `void actualizarViajes(vViajes* v)`

5.4. Definiciones

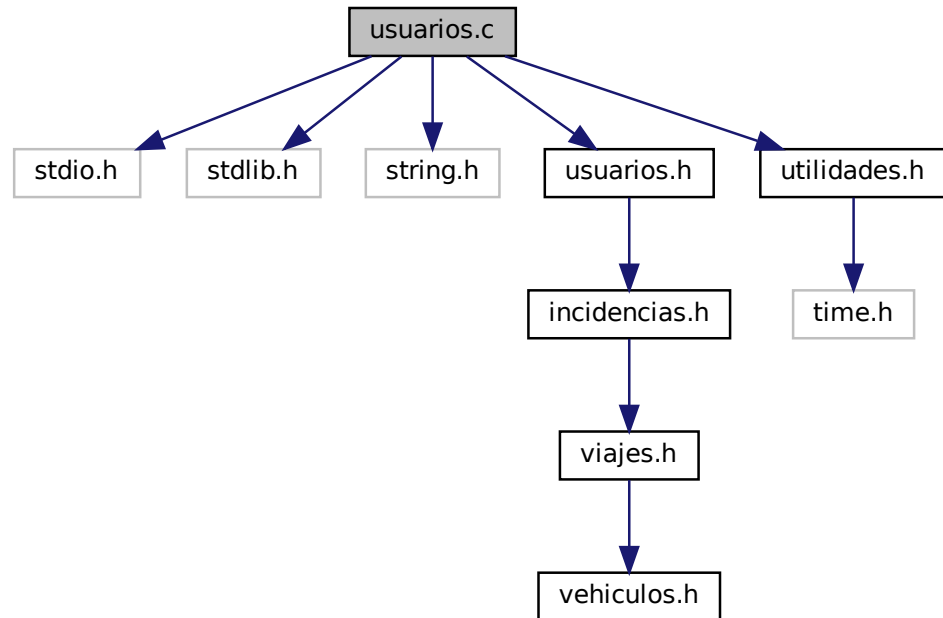
- `Viajes* initViajes(int* n)`
 - **Descripcion**
 - Inicializa una estructura del tipo Viajes.
 - **Parametros**
 - `n` → Referencia al tamaño de la estructura.
 - **Devuelve**
 - Un vector con los datos del fichero Viajes.txt
- `Pasos* initPasos(int* n)`
 - **Descripcion**
 - Inicializa una estructura del tipo Pasos.
 - **Parametros**
 - `n` → Referencia al tamaño de la estructura.
 - **Devuelve**
 - Un vector con los datos del fichero Pasos.txt
- `Pasajeros* initPasajeros(int* n)`
 - **Descripcion**
 - Inicializa una estructura del tipo Pasajeros.
 - **Parametros**
 - `n` → Referencia al tamaño de la estructura.
 - **Devuelve**
 - Un vector con los datos del fichero Pasajeros.txt
- `void publicarViajeUsuario(vViajes* v, vVehiculos* ve, int userId)`
 - **Descripcion**
 - Funcion para publicar un viaje en el sistema de esi-share.
 - **Parametros**
 - `v` → Referencia al vector de viajes.
 - `ve` → Referencia al vector de vehiculos.
 - `userId` → Identificador del usuario que publica el viaje.
- `void editarViajesUsuario(vViajes* v, vVehiculos* ve, int userId)`
 - **Descripcion**
 - Permite la edicion de un viaje en estado abierto.
 - **Parametros**
 - `v` → Referencia al vector de viajes.
 - `ve` → Referencia al vector de vehiculos.
 - `userId` → Identificador del usuario que publico el viaje.
- `void incorporarseViaje(vViajes* v)`
 - **Descripcion**
 - Permite a un usuario incorporarse a un viajes publicado en el sistema.
 - **Parametros**
 - `v` → Referencia al vector de viajes.

- `void detalleViaje(vViajes* v)`
 - **Descripcion**
 - Permite a un usuario ver los datos de un viaje al detalle.
 - **Parametros**
 - `v` → Referencia al vector de viajes.
- `void cancelarViaje(vViajes* v)`
 - **Descripcion**
 - Cancelar la incorporacion a un viaje.
 - **Parametros**
 - `v` → Referencia al vector de viajes.
 - `Id_usuario` → Identificador del que se da de baja.
- `void publicarViajeAdmin(vViajes* v, vVehiculos* ve)`
 - **Descripcion**
 - Permite a un administrador publicar un viaje en nombre de un usuario.
 - **Parametros**
 - `v` → Referencia al vector de viajes.
 - `ve` → Referencia al vector de vehiculos.
- `void eliminarViajesAdmin(vViajes* v)`
 - **Descripcion**
 - Permite a un administrador eliminar un viaje.
 - **Parametros**
 - `v` → Referencia al vector de viajes.
- `void modificarViajesAdmin(vViajes* v, vVehiculos *vve)`
 - **Descripcion**
 - Permite a un administrador eliminar un viaje.
 - **Parametros**
 - `v` → Referencia al vector de viajes.
 - `vve` → Referencia al vector de vehiculos.
- `void listarViajesAdmin(vViajes* v)`
 - **Descripcion**
 - Muestra los viajes al detalle.
 - **Parametros**
 - `v` → Referencia al vector de viajes.
- `void saveViajes(int n, Viajes* viajes)`
 - **Descripcion**
 - Guarda los datos en el fichero Viajes.txt y libera la memoria.
 - **Parametros**
 - `n` → Tamaño del vector user en vViajes.
 - `v` → Referencia al vector de viajes.

- `void savePasos(int n,Pasos* pasos)`
 - **Descripcion**
 - Guarda los datos en el fichero Pasos.txt y libera la memoria.
 - **Parametros**
 - `n` → Tamaño del vector pasos en `vViajes`.
 - `v` → Referencia al vector de pasos.
- `void savePasajeros(int n,Pasajeros* pasaj)`
 - **Descripcion**
 - Guarda los datos en el fichero Pasajeros.txt y libera la memoria.
 - **Parametros**
 - `n` → Tamaño del vector pasos en `vViajes`.
 - `v` → Referencia al vector pasaj.
- `int buscarIndexViajes(vViajes* v,int id_viaje)`
 - **Descripcion**
 - Busca un viaje el vector `vViajes`.
 - **Parametros**
 - `v` → Referencia al vector de viajes.
 - `id_viaje` → Identificador del viaje a buscar.
 - **Devuelve**
 - `iesima` posicion del vector donde se encuentra el viaje.
 - `-1` si no se encuentra.
- `int buscarIndexPasejeros(vViajes *v, int viaje,int viajero)`
 - **Descripcion**
 - Busca si un usuario esta en un viaje el vector `vViajes`.
 - **Parametros**
 - `v` → Referencia al vector de viajes.
 - `id_viaje` → Identificador del viaje a buscar.
 - `id_viajero` → Identificador del usuario a buscar.
 - **Devuelve**
 - `iesima` posicion del vector donde se encuentra.
 - `-1` si no se encuentra.
- `void listarViajesAbiertos(vViajes* v)`
 - **Descripcion**
 - Muestra los viajes en estado abierto.
 - **Parametros**
 - `v` → Referencia al vector de viajes.
- `void actualizarViajes(vViajes* v)`
 - **Descripcion**
 - Actualiza el estado de los viajes.
 - **Parametros**
 - `v` → Referencia al vector de viajes.

6. USUARIOS

6.1. Gráfico de dependencias



6.2. Estructura de Datos

- `struct Usuarios`
- `struct vUsuarios`

6.3. Funciones

- `Usuarios* initUsuarios(int * n)`
- `void saveUsuarios(int n, Usuarios *usuarios)`
- `void listarUsuarios(vUsuarios* u,vIncidencias* vi)`
- `void altaUsuario(vUsuarios* v)`
- `void modificarUsuario(vUsuarios* v,int userId)`
- `void perfilUsuario(vUsuarios* v,int userId)`
- `void preguntarIdBaja(vUsuarios* v)`
- `int printPerfil(vUsuarios* v,int userIndex)`
- `void reguntarIdModificar(void)`

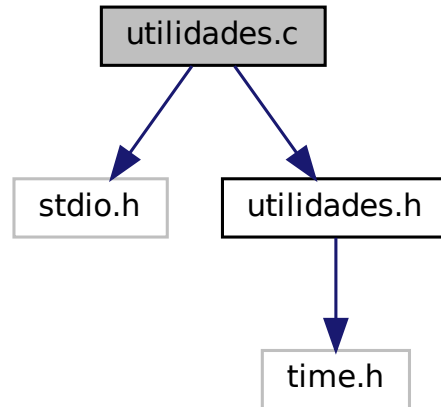
6.4. Definiciones

- `Usuarios* initUsuarios(int * n)`
 - **Descripcion**
 - Inicializa una estructura del tipo Usuarios.
 - **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**.
- `void saveUsuarios(int n, Usuarios *usuarios)`
 - **Descripcion**
 - Guarda el contenido actual de una estructura del tipo Usuarios en ficheros.
 - **Parametros**
 - `n` → Referencia a la posición del vector que almacena el número de usuarios.
 - `usuarios` → Puntero a la estructura de usuarios.
- `void listarUsuarios(vUsuarios* u,vIncidencias* vi)`
 - **Descripcion**
 - Lista el contenido actual de la estructura del tipo Usuarios.
 - **Parametros**
 - `u` → Referencia al vector de Usuarios
 - `vi` → Referencia al vector de Incidencias.
- `void altaUsuario(vUsuarios* v)`
 - **Descripcion**
 - Añade una nueva línea a la estructura del tipo Usuarios.
 - **Parametros**
 - `v` → Referencia al vector de Usuarios.
- `void modificarUsuario(vUsuarios* v,int userId)`
 - **Descripcion**
 - Modificar una línea concreta de la estructura del tipo Usuarios.
 - **Parametros**
 - `v` → Referencia al vector de Usuarios.
 - `userId` → Referencia al entero con el índice del usuario seleccionado.
- `void perfilUsuario(vUsuarios* v,int userId)`
 - **Descripcion**
 - Permite editar al usuario sus datos personales en el sistema y los modifica en la estructura del tipo Usuarios.
 - **Parametros**
 - `v` → Referencia al vector de Usuarios.
 - `userId` → Referencia al entero con el índice del usuario seleccionado.

- `void preguntarIdBaja(vUsuarios* v)`
 - **Descripcion**
 - Pregunta al administrador qué usuario quiere dar de baja.
 - **Parametros**
 - `v` → Referencia al vector de Usuarios.
- `void printPerfil(vUsuarios* v, int userIndex)`
 - **Descripcion**
 - Imprime por pantalla el perfil de un usuario en concreto.
 - **Parametros**
 - `v` → Referencia al vector de Usuarios.
 - `userIndex` → Referencia al entero con el indice del usuario seleccionado.
- `int preguntarIdModificar(void)`
 - **Descripcion**
 - Pregunta al administrador qué usuario desea modificar.
 - **Devuelve**
 - `tmp` → Entero con el id seleccionado.

7. UTILIDADES

7.1. Gráfico de dependencias



7.2. Funciones

- `void flush_in(void)`
- `void system_pause(void)`
- `int validarFecha(char *cadena)`
- `int validarHora(char *cadena, int hoy)`
- `int fechaMenor(struct tm* fecha)`
- `int fechaIgual(struct tm* fecha)`
- `int horaMenor(struct tm* hora)`

7.3. Definiciones

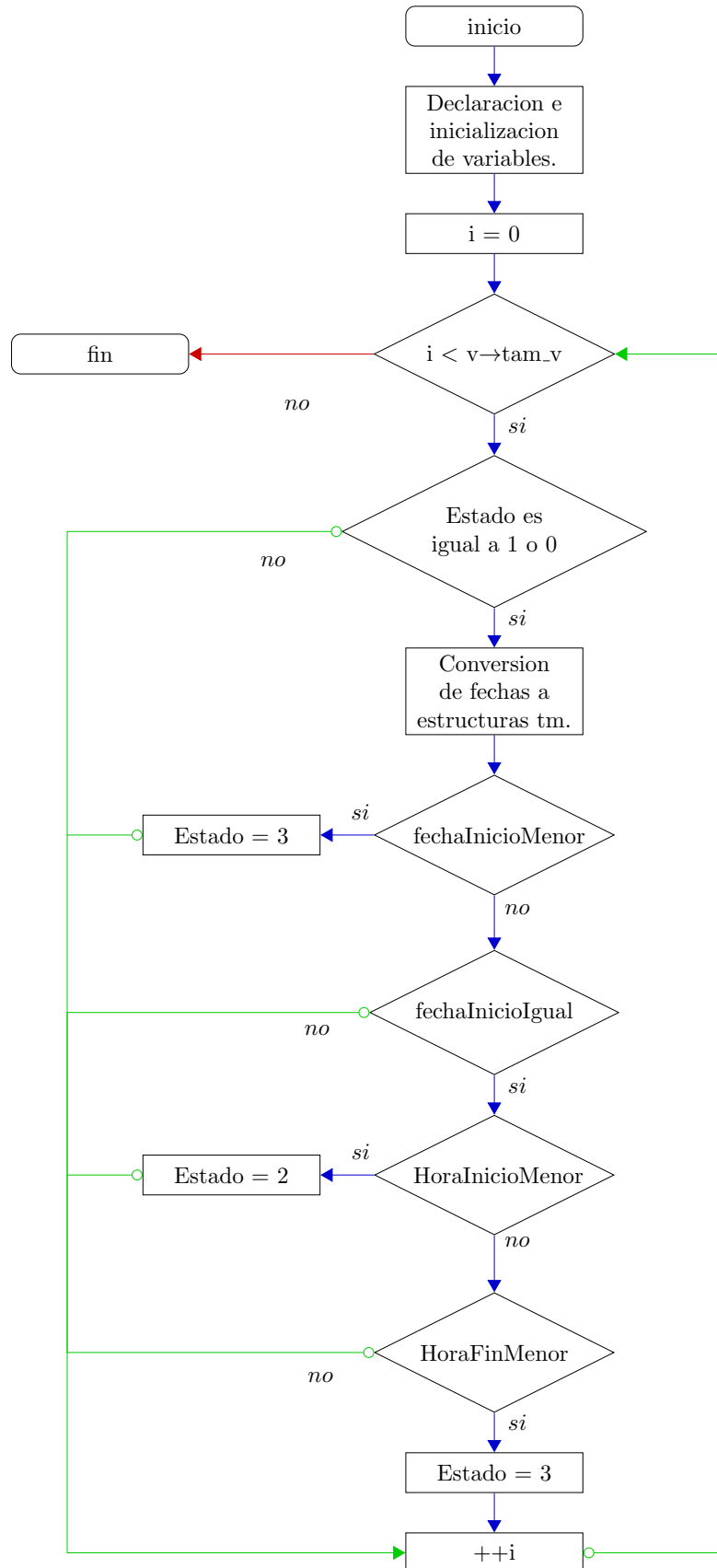
- `void flush_in(void)`
 - **Descripcion**
 - Vacía el flujo de la entrada estandar.
- `void system_pause(void)`
 - **Descripcion**
 - Agrega una pausa en la ejecucion.

- `int validarFecha(char *cadena)`
 - **Descripcion**
 - Valida una cadena con el formato dd/mm/aaaa.
 - **Parametros**
 - `cadena` → Contiene una fecha.
 - **Devuelve**
 - 1 → Formato correcto y fecha igual a la del sistema.
 - 0 → Formato correcto.
 - -1 → Formato incorrecto.
- `int validarHora(char *cadena, int hoy)`
 - **Descripcion**
 - Valida una cadena con el formato hh:mm.
 - **Parametros**
 - `cadena` → Contiene una hora.
 - `hoy` → Contiene el dia de hoy.
 - **Devuelve**
 - 1 → Formato correcto.
 - 0 → Formato incorrecto.
- `int fechaMenor(struct tm* fecha)`
 - **Descripcion**
 - Comprueba si la fecha es menor a la de hoy.
 - **Parametros**
 - `fecha` → Contiene tm que contiene la fecha a comprobar.
 - **Devuelve**
 - 1 → Si la fecha es menor.
 - 0 → Si la fecha es mayor o igual.
- `int fechaIgual(struct tm* fecha)`
 - **Descripcion**
 - Comprueba si la fecha es igual a la de hoy.
 - **Parametros**
 - `fecha` → Contiene tm que contiene la fecha a comprobar.
 - **Devuelve**
 - 1 → Si la fecha es igual.
 - 0 → Si la fecha es mayor o menor.
- `int horaMenor(struct tm* hora)`
 - **Descripcion**
 - Comprueba si una hora es menor a la hora actual del sistema.
 - **Parametros**
 - `hora` → Contiene tm que contiene la hora a comprobar.
 - **Devuelve**
 - 1 → Si la hora es menor.
 - 0 → Si la hora es mayor o igual.

8. Test

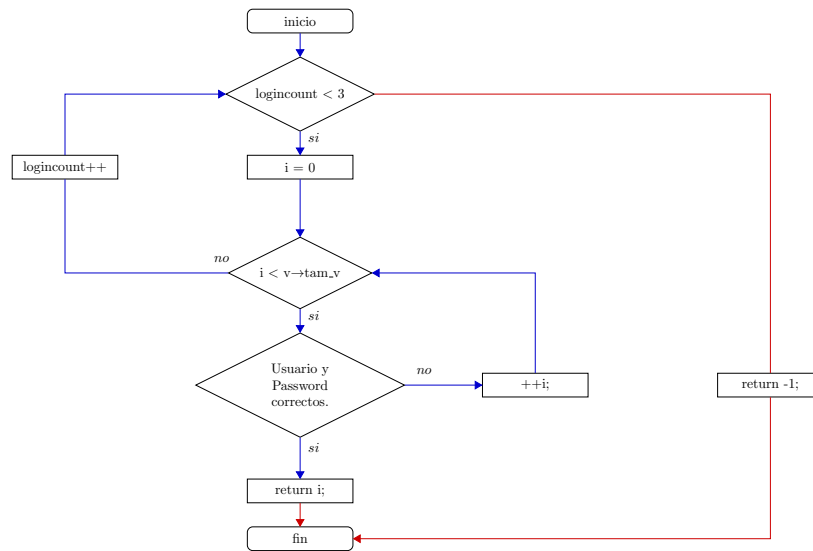
8.1. Test Viajes

8.1.1. Diagrama de flujo



8.2. Test Viajes

8.2.1. Diagrama de flujo



8.3. Test Viajes

8.3.1. Diagrama de flujo

