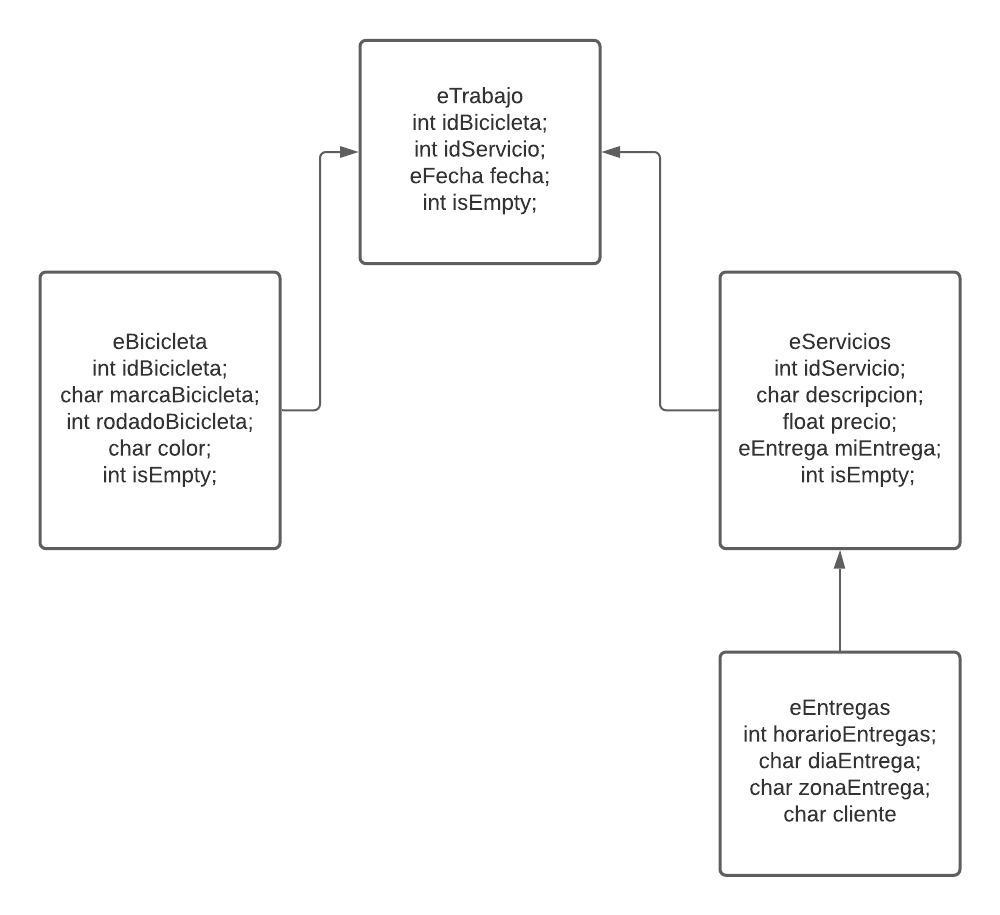
**Domínguez Estefania**

**Recuperatorio Parcial 1-1F**



**Estructura agregada**

**typedef** **struct**

{

**int** horarioEntregas;

**char** diaEntrega[MAX\_CHARS\_CADENAS];

**char** zonaEntrega[MAX\_CHARS\_CADENA];

**char** cliente[MAX\_CHARS\_CADENA];

**int** isEmpty;

}eEntrega;

**Documentacion**

**eTrabajo**

/\*\*

\* @fn int TotalPorServicios(eServicios[], int, eTrabajos[], int)

\* @brief trabajos ordenados por año y ante igualdad de año por marca de la bicicleta.

\*

\* @param lista

\* @param tam

\* @param listado

\* @param tamT

\* @return

\*/

**int** TotalPorServicios(eServicios lista[], **int** tam, eTrabajos listado[], **int** tamT);

/\*\*

\* @fn void InicializarTrabajo(eTrabajos[], int)

\* @brief inicializa array en espacio libre

\*

\* @param lista

\* @param tam

\*/

**void** InicializarTrabajo(eTrabajos lista[],**int** tam);

/\*\*

\* @fn int ObtenerID(void)

\* @brief genera id autoincremental

\*

\* @return id

\*/

**int** ObtenerID(**void**);

/\*\*

\* @fn int TrabajoBuscarLibre(eTrabajos[], int)

\* @brief busca espacio libre para cargar datos

\*

\* @param array

\* @param TAM

\* @return el index del primer vector libre

\*/

**int** TrabajoBuscarLibre(eTrabajos array[], **int** TAM);

/\*\*

\* @fn void HardcodeoTrabajo(eTrabajos[], int)

\* @brief listado de datos pasados

\*

\* @param lista

\* @param tam

\*/

**void** HardcodeoTrabajo(eTrabajos lista[], **int** tam);

/\*\*

\* @fn void MostrarHardcodeoTrabajo(eTrabajos[], int)

\* @brief muestra el hardcodeo de datos

\*

\* @param lista

\* @param tam

\*/

**void** MostrarHardcodeoTrabajo(eTrabajos lista[], **int** tam);

/\*\*

\* @fn int CargarDatosTrabajo(eTrabajos[], int)

\* @brief busca espacio libre y si encuentra pide carga datos del alta

\*

\* @param array

\* @param TAM

\* @return el auxiliar del espacio que encontro

\*/

**int** CargarDatosTrabajo(eTrabajos array[], **int** TAM);

/\*\*

\* @fn eTrabajos AltaTrabajo()

\* @brief pide cargar datos

\*

\* @return datos cargados

\*/

**int** AltaTrabajo();

/\*\*

\* @fn int Modificacion(eTrabajos[], int, eServicios[], int, eBicicleta[], int)

\* @brief modifica un elemento de la lista

\*

\* @return 0 en caso de error, 1 en caso de modificacion exitosa

\*/

**int** Modificacion(eTrabajos producto[],**int** tam,eServicios servicio[], **int** tamT, eBicicleta miBicicleta[], **int** tamB);

/\*\*

\* @fn int eTrabajosBaja(eTrabajos[], int, eBicicleta[], int)

\* @brief baja logica

\*

\* @return-1 para baja, borra el dato y deja cargar otro

\*/

**int** eTrabajosBaja(eTrabajos array[], **int** TAM,eBicicleta lista[], **int** tam);

/\*\*

\* @fn int eTrabajos\_BuscarPorID(eTrabajos[], int, int)

\* @brief busca id por parametro

\*

\* @return el index del id buscado

\*/

**int** eTrabajos\_BuscarPorID(eTrabajos array[], **int** TAM, **int** ID);

/\*\*

\* @fn void MostrarTrabajo(eTrabajos, int, eBicicleta, int)

\* @brief muestra un trabajo cargado

\*

\* @param listado

\* @param tamT

\* @param lista

\* @param tamb

\*/

**void** MostrarTrabajo(eTrabajos listado, **int** tamT, eBicicleta lista, **int** tamb);

/\*\*

\* @fn int MostrarTodosLosTrabajos(eTrabajos[], int, eBicicleta[], int)

\* @brief muestra todos los trabajos cargados

\*

\* @param listado

\* @param TAM

\* @param lista

\* @param tam

\* @return lista de trabajos cargados

\*/

**int** MostrarTodosLosTrabajos(eTrabajos listado[], **int** TAM, eBicicleta lista[], **int** tam);

/\*\*

\* @fn int OrdenarporCriterio(eTrabajos[], int, eBicicleta[], int)

\* @brief ordena segun parametro pasado

\*

\* @return 0 en caso de error, 1 en caso de sort exitoso

\*/

**int** OrdenarporCriterio(eTrabajos array[], **int** TAM, eBicicleta lista[], **int** tam);

/\*\*

\* @fn void ListarTrabajos(eTrabajos[], int, eServicios[], int)

\* @brief muestra listado segun criterio de ordenamiento

\*

\* @param listaTrabajo

\* @param tam

\* @param listaServicio

\* @param tamM

\*/

**void** ListarTrabajos(eTrabajos listaTrabajo[],**int** tam, eServicios listaServicio[],**int** tamM,eBicicleta miLista[], **int** tamb);

/\*\*

\* @fn void ListadoPorCriterio(eTrabajos[], int, eServicios[], int, eBicicleta[], int)

\* @brief muestra listado, segun ordenamiento

\*

\* @param listaTrabajo

\* @param tam

\* @param listaServicio

\* @param tamM

\* @param miLista

\* @param tamb

\*/

**void** ListadoPorCriterio(eTrabajos listaTrabajo[],**int** tam, eServicios listaServicio[],**int** tamM, eBicicleta miLista[], **int** tamb);

/\*\*

\* @fn int BiciRojaConServicio(eServicios[], int, eBicicleta[], int, eTrabajos[], int)

\* @brief funcion que suma la cant de bicicletas rojas que tengan un servicio

\*

\* @return 1 si el contador encontro bicicleta con el parametro pasado, 0 en caso de error

\*/

**int** BiciRojaConServicio(eServicios listaServicios[],**int** tamS,eBicicleta listaBicicletas[], **int** tamB, eTrabajos lista[], **int** tam);

**eServicios**

/\*\*

\* @fn void InicializarServicios(eServicios[], int)

\* @brief inicializa array en espacio libre

\*

\* @param lista

\* @param tam

\*/

**void** InicializarServicios(eServicios lista[],**int** tam);

/\*\*

\* @fn int ObtenerIDservicio(void)

\* @brief genera id autoincremental

\*

\* @return id

\*/

**int** ObtenerIDservicio(**void**);

/\*\*

\* @fn int ServicioBuscarLibre(eServicios[], int)

\* @brief busca espacio para carga de datos

\*

\* @param array

\* @param TAM

\* @return el index del primer vector libre

\*/

**int** ServicioBuscarLibre(eServicios array[], **int** TAM);

/\*\*

\* @fn void HardcodeoServicio(eServicios[], int)

\* @brief listado de datos pasados

\*

\* @param lista

\* @param tam

\*/

**void** HardcodeoServicio(eServicios lista[], **int** tam);

/\*\*

\* @fn void MostrarHardcodeo(eServicios[], int)

\* @brief muestra la lista de datos hardcodeados

\*

\* @param lista

\* @param tam

\*/

**void** MostrarHardcodeo(eServicios lista[], **int** tam);

/\*\*

\* @fn int Servicio\_CargarDatos(eServicios[], int)

\* @brief busca espacio libre y si encuentra pide carga datos del alta

\*

\* @param array

\* @param TAM

\* @return datos cargados

\*/

**int** Servicio\_CargarDatos(eServicios array[], **int** TAM);

/\*\*

\* @fn eServicios AltaServicio(eServicios[], int)

\* @brief pide cargar datos

\*

\* @param lista

\* @param tam

\* @return datos cargados

\*/

eServicios AltaServicio(eServicios lista[], **int** tam);

/\*\*

\* @fn void MostrarServicio(eServicios)

\* @brief print de listado de servicios

\*

\* @param listado

\* @param tamT

\*/

**void** MostrarServicio(eServicios listado);

/\*\*

\* @fn void ListaServicios(eServicios[], int)

\* @brief lista de servicios hardcodeados

\*

\* @param listado

\* @param tam

\*/

**void** ListaServicios(eServicios listado[],**int** tam);

/\*\*

\* @fn int eServicios\_BuscarPorID(eServicios[], int, int)

\* @brief busco id

\*

\* @param array

\* @param TAM

\* @param ID

\* @return devuelve el index del id encontrado

\*/

**int** eServicios\_BuscarPorID(eServicios array[], **int** TAM, **int** ID);

/\*\*

\* @fn int eServiciosBaja(eServicios[], int)

\* @brief borrado logico

\*

\* @param array

\* @param TAM

\* @return1 si se accedio a la baja, 0 sino se accedio

\*/

**int** eServiciosBaja(eServicios array[], **int** TAM);

/\*\*

\* @fn int TrabajosOrdenadosPorMarca(eServicios[], int, eBicicleta[], int)

\* @brief ordena segun parametro

\*

\* @return 1 si ordeno, 0 en caso de error

\*/

**int** TrabajosOrdenadosPorMarca(eServicios array[], **int** TAM, eBicicleta lista[], **int** tam);

/\*\*

\* @fn void MostrarTrabajosOrdenadosPorMarca(eServicios[], int, eBicicleta[], int)

\* @brief

\*

\* @param listaServicio

\* @param tamM

\* @param miLista

\* @param tam

\*/

**void** MostrarTrabajosOrdenadosPorMarca(eServicios listaServicio[],**int** tamM, eBicicleta miLista[], **int** tam);

/\*\*

\* @fn void ListadoDeServiciosConBicicletas(eServicios[], int, eBicicleta[], int)

\* @brief imprime listado de servicios de cada bicicleta

\*

\* @param listado

\* @param tamT

\* @param lista

\* @param tam

\*/

**void** ListadoDeServiciosConBicicletas(eServicios listado[], **int** tamT, eBicicleta lista[], **int** tam);

//void ServiciosConMasTrabajos(eServicios lista[],int tam);

//void MostrarServiciosConMasTrabajos(eServicios listado[], int tam);

/\*\*

\* @fn int ListadoEntregas(eServicios[], int, eBicicleta[], int)

\* @brief listado de entrega, segun parametros

\*

\* @return

\*/

**int** ListadoEntregas(eServicios listaServicio[],**int** tamM,eBicicleta miLista[], **int** tam);

/\*\*

\* @fn void HardcodeoEntregas(eEntrega[], int)

\* @brief hardcodeo de la estructura de entregas

\*

\* @param lista

\* @param tam

\*/

**void** HardcodeoEntregas(eEntrega lista[], **int** tam);

/\*\*

\* @fn void MostrarHardcodeoEntregas(eEntrega[], int, eBicicleta[])

\* @brief muestra la estrucrura con los datos pasados

\*

\* @param lista

\* @param tam

\* @param miMarca

\*/

**void** MostrarHardcodeoEntregas(eEntrega lista[], **int** tam, eBicicleta miMarca[]);

/\*\*

\* @fn int ServicioOrdenadoZona(eEntrega[], int, eTrabajos[], int, eServicios[], int)

\* @brief ordena el servicio con horario, segun al zona

\*

\* @return

\*/

**int** ServicioOrdenadoZona(eEntrega listado[], **int** TAM, eTrabajos lista[], **int** tam, eServicios listaS[], **int** tamS);

/\*\*}

\* @fn void MostrarTrabajosOrdenadosPorZona(eEntrega[], int, eServicios[], int, eTrabajos[], int)

\* @brief muestra los servicios ordenados por zona y horario

\*

\* @param miEntrega

\* @param tamE

\* @param listaServicio

\* @param tamM

\* @param lista

\* @param tam

\*/

**void** MostrarTrabajosOrdenadosPorZona(eEntrega miEntrega[], **int** tamE, eServicios listaServicio[],**int** tamM, eTrabajos lista[], **int** tam);

**eBicicleta**

/\*\*

\* @fn void InicializarBicicletas(eBicicleta[], int)

\* @brief inicializa array en espacio libre

\*

\* @param lista

\* @param tam

\*/

**void** InicializarBicicletas(eBicicleta lista[],**int** tam);

/\*\*

\* @fn int ObtenerIDbicicleta(void)

\* @brief busca id

\*

\* @return en index del id encontrado

\*/

**int** ObtenerIDbicicleta(**void**);

/\*\*

\* @fn int BicicletaBuscarLibre(eBicicleta[], int)

\* @brief id AUTOINCREMENTAL

\*

\* @param array

\* @param TAM

\* @return id

\*/

**int** BicicletaBuscarLibre(eBicicleta array[], **int** TAM);

/\*\*

\* @fn void HardcodeoBicicleta(eBicicleta[], int)

\* @brief hardcodeo de datos de la estructura

\*

\* @param lista

\* @param tam

\*/

**void** HardcodeoBicicleta(eBicicleta lista[], **int** tam);

/\*\*

\* @fn void MostrarHardcodeoBicicleta(eBicicleta[], int)

\* @brief muestra los datos hardcodeados

\*

\* @param lista

\* @param tam

\*/

**void** MostrarHardcodeoBicicleta(eBicicleta lista[], **int** tam);

/\*\*

\* @fn int eBicicleta\_BuscarPorID(eBicicleta[], int, int)

\* @brief busqueda por id

\*

\* @param array

\* @param TAM

\* @param ID

\* @return el index del id encontrado

\*/

**int** eBicicleta\_BuscarPorID(eBicicleta array[], **int** TAM, **int** ID);

**LINK VIDEO**

https://drive.google.com/file/d/1w-WqfCgnFd7aTP1gQym4jRAhlDfy2o4c/view?usp=sharing