

|  |
| --- |
| RECUPERATORIO 1° PARCIAL  LABORATORIO I |
|  |
| 27 noviembre 2021  Laboratorio de Computacion I  Creado por: Sergio Ledesma |

# Prototipos de las funciones:

Biblioteca “eDireccion.h”:

/// @fn int eDireccion\_IngresarDireccion(eDireccion\*)

/// @brief Le pide una direccion al usuario y lo guarda en una direccion de memoria

///

/// @param direccion -> Direccion de memoria donde guardara el dato

/// @return Retorna 1 (OK) o 0 (Error)

int eDireccion\_IngresarDireccion(eDireccion\* direccion);

/// @fn int eDireccion\_MostrarDireccion(eDireccion)

/// @brief Muestra una direccion por consola

///

/// @param direccion -> direccion a mostrar

/// @return Retorna 1 (OK) o 0 (Error)

int eDireccion\_MostrarDireccion(eDireccion direccion);

Biblioteca “eLocalidad.h”:

/// @fn int eLocalidad\_ValidarLocalidades(eLocalidad[], int)

/// @brief Valida que al menos haya una localidad

///

/// @param lista -> Array de la localidad

/// @param tam -> Tamaño del array

/// @return Retorna 1 (OK) o 0 (ERROR)

int eLocalidad\_ValidarLocalidades(eLocalidad lista[], int tam);

/// @fn int eLocalidad\_MostrarUnaLocalidad(int, eLocalidad[], int)

/// @brief Imprime una localidad

///

/// @param idLocalidad -> ID de la localidad

/// @param lista -> Array de la localidad

/// @param tam -> Tamaño del array

/// @return Retorna 1 (OK) o 0 (ERROR)

int eLocalidad\_MostrarUnaLocalidad(int idLocalidad, eLocalidad lista[], int tam);

/// @fn int eLocalidad\_MostrarLocalidades(eLocalidad[], int)

/// @brief Muestra por consola las localidades disponibles

///

/// @param lista -> Array de la localidad

/// @param tam -> Tamaño del array

/// @return Retorna 1 (OK) o 0 (ERROR)

int eLocalidad\_MostrarLocalidades(eLocalidad lista[], int tam);

/// @fn int eLocalidad\_IngresarLocalidad(int\*, eLocalidad[], int)

/// @brief

///

/// @param idLocalidad -> ID de la localidad elegida

/// @param lista -> Array de localidades

/// @param tam -> Tamaño del array

/// @return Retorna 1 (OK) o 0 (ERROR)

int eLocalidad\_IngresarLocalidad(int\* idLocalidad, eLocalidad lista[], int tam);

Biblioteca “eCliente.h”:

/// @fn int eCliente\_InicializarClientes(eCliente[], int)

/// @brief Inicializa el campo "isEmpty" de los clientes en 0 (EMPTY)

///

/// @param clientes -> Array de clientes

/// @param tam -> Tamaño del array

/// @return Retorna 1 (OK) o 0 (Error)

int eCliente\_InicializarClientes(eCliente clientes[], int tam);

/// @fn int eCliente\_BuscarPrimerEspacioLibre(eCliente[], int)

/// @brief Busca el primer espacio libre del array y retorna el indice

///

/// @param clientes -> Array de clientes

/// @param tam -> Tamaño del array

/// @return Retorna -1 (Error) o el indice del 1er espacio libre del array

int eCliente\_BuscarPrimerEspacioLibre(eCliente clientes[], int tam);

/// @fn int eCliente\_IngresarUnCliente(eCliente\*)

/// @brief Toma por consola los datos de un cliente y los guarda

/// en una direccion de memoria de una estructura

///

/// @param cliente -> Direccion de memoria de la estructura

/// @param localidades -> Array de localidades

/// @param tam -> Tamaño del array

/// @return Retorna 1 (OK) o 0 (Error)

int eCliente\_IngresarUnCliente(eCliente\* cliente, eLocalidad localidades[], int tam);

/// @fn int eCliente\_CargarUnCliente(eCliente[], int, eLocalidad[], int, int\*)

/// @brief Carga un nuevo cliente en el array de clientes, asignando el ultimo id al mismo

///

/// @param clientes -> Array de clientes

/// @param tamClientes -> Tamaño del array

/// @param localidades -> Array de localidades

/// @param tamLocalidades -> Tamaño del array

/// @param ultimoId -> ultimo id que se cargo

/// @return Retorna 1 (OK), 0 (Error)

int eCliente\_CargarUnCliente(eCliente clientes[], int tamClientes, eLocalidad localidades[], int tamLocalidades, int\* ultimoId);

/// @fn int eCliente\_BuscarClientePorID(eCliente[], int, int)

/// @brief Busca un cliente en un array y si lo encuentra retorna el indice

///

/// @param clientes -> Array de clientes

/// @param tam -> Tamaño del array

/// @param id -> ID del cliente a buscar

/// @return Retorna -1 (Error) o el indice del cliente buscado

int eCliente\_BuscarClientePorID(eCliente clientes[], int tam, int id);

/// @fn int eCliente\_ModificarCliente(eCliente[], int, eLocalidad[], int)

/// @brief Busca en la lista un cliente y modifica los datos segun lo ingresado en consola

///

/// @param clientes -> Lista de clientes

/// @param tamClientes -> Tamaño de la lista

/// @param localidades -> Lista de localidades

/// @param tamLocalidades -> Tamaño de la lista

/// @return Retorna 1 (OK) o 0 (Error)

int eCliente\_ModificarCliente(eCliente clientes[], int tamClientes, eLocalidad localidades[], int tamLocalidades);

/// @fn int eCliente\_EliminarCliente(eCliente[], int, int)

/// @brief Busca un cliente por ID y si lo encuentra da la opcion de eliminar

///

/// @param clientes -> Array de clientes

/// @param tam -> Tamaño del array

/// @param id -> ID del cliente a buscar

/// @return Retorna 1 (OK), 0 (No se encontro el cliente) o -1 (Se cancelo la operacion)

int eCliente\_EliminarCliente(eCliente clientes[], int tam, int id);

/// @fn int eCliente\_VerificarClientes(eCliente[], int)

/// @brief Recorre el array de clientes buscando si hay clientes activos

///

/// @param clientes -> Array de clientes

/// @param tam -> Tamaño del array

/// @return Retorna 1 (Encontro cliente) o 0 (No encontro cliente)

int eCliente\_VerificarClientes(eCliente clientes[], int tam);

/// @fn int eCliente\_MostrarUnCliente(eCliente, eLocalidad[], int)

/// @brief Muestra los datos de un cliente por consola

///

/// @param cliente -> Cliente a mostrar

/// @param localidades -> Lista de localidades

/// @param tam -> Tamaño de la lista

/// @return Retorna 1 (OK) o 0 (Error)

int eCliente\_MostrarUnCliente(eCliente cliente, eLocalidad localidades[], int tam);

/// @fn int eCliente\_MostrarDatosClienteEnPedido(eCliente[], int, eLocalidad[], int, int)

/// @brief Muestra los datos de un cliente por consola (Llamada por Informes para mostrar junto a un pedido)

///

/// @param clientes -> Lista de clientes

/// @param tamClientes -> Tamaño de la lista

/// @param localidades -> Lista de localidades

/// @param tamLocalidades -> Tamaño de la lista

/// @param idCliente -> ID del cliente a mostrar

/// @return Retorna 1 (OK) o 0 (Error)

int eCliente\_MostrarDatosClienteEnPedido(eCliente clientes[], int tamClientes, eLocalidad localidades[], int tamLocalidades, int idCliente);

/// @fn int eCliente\_MostrarClientes(eCliente[], int, eLocalidad[], int)

/// @brief Muestra los clientes que hay cargados en la lista

///

/// @param clientes -> Lista de clientes

/// @param tamClientes -> Tamaño de la lista

/// @param localidades -> Lista de localidades

/// @param tamLocalidades -> Tamaño de la lista

/// @return Retorna 1 (OK) o 0 (Error)

int eCliente\_MostrarClientes(eCliente clientes[], int tamClientes, eLocalidad localidades[], int tamLocalidades);

Biblioteca “ePedido.h”:

/// @fn int ePedido\_InicializarPedidos(ePedido[], int)

/// @brief Inicializa el campo "isEmpty" de los pedidos en 0 (EMPTY)

///

/// @param pedidos -> Array de pedidos

/// @param tam -> Tamaño de pedidos

/// @return Retorna 1 (OK) o 0 (Error)

int ePedido\_InicializarPedidos(ePedido pedidos[], int tam);

/// @fn int ePedido\_BuscarPrimerEspacioLibre(ePedido[], int)

/// @brief Busca el primer espacio libre del array y retorna el indice

///

/// @param pedidos -> Array de pedidos

/// @param tam -> Tamaño del array

/// @return Retorna -1 (Error) o el indice del primer espacio libre

int ePedido\_BuscarPrimerEspacioLibre(ePedido pedidos[], int tam);

/// @fn int ePedido\_BuscarPedidoPendientePorCliente(ePedido[], int, int)

/// @brief Busca si el cliente con el ID ingresado tiene pedidos pendientes,

/// si lo encuentra retorna el indice

///

/// @param pedidos -> Array de pedidos

/// @param tam -> Tamaño del array

/// @param id -> ID del cliente

/// @return Retorna -1 (No encontro pedidos pendientes para ese cliente) o el indice del cliente

int ePedido\_BuscarPedidoPendientePorCliente(ePedido pedidos[], int tam, int id);

/// @fn int ePedido\_CargarUnPedido(ePedido[], int, eCliente[], int, int\*)

/// @brief Pide por consola el ID del cliente y los kg totales a reciclar,

/// luego genera un nuevo pedido.

///

/// @param pedidos -> Array de pedidos

/// @param tamPedidos ->Tamaño del array pedidos

/// @param clientes -> Array de clientes

/// @param tamClientes -> Tamaño del array clientes

/// @param ultimoPedido -> Ultimo pedido ingresado (0 si no se ingreso ninguno)

/// @return Retorna 1 (OK), 0 (Si no se pudo cargar el pedido) o -1 (Si los parametros no son validos)

int ePedido\_CargarUnPedido(ePedido pedidos[], int tamPedidos, eCliente clientes[], int tamClientes, int\* ultimoPedido);

/// @fn int ePedido\_ContarPedidos(ePedido[], int, int, int)

/// @brief Cuenta los pedidos activos de un cliente segun el modo indicado

///

/// @param pedidos -> Lista de pedidos

/// @param tam -> Tamaño de la lista

/// @param idCliente -> ID del cliente

/// @param modo -> 1 (Pendientes), 2 (Completados), si se ingresa cualquier otro entero contara

/// pendientes y completados

/// @return Retorna la cantidad de pedidos que tiene ese cliente segun el modo

int ePedido\_ContarPedidos(ePedido pedidos[], int tam, int idCliente, int modo);

/// @fn int ePedido\_MostrarUnPedido(ePedido, int)

/// @brief Muestra por consola los datos de un pedido dependiendo el modo indicado

/// (1 - Pendiente o 2 - Completado)

///

/// @param pedido -> Puntero a estructura

/// @param modo -> Entero que indica que campos mostrara (1 - Pendiente o 2 - Completado)

/// @return Retorna 1 (Pudo mostrar al menos un pedido) o 0 (No encontro resultados)

int ePedido\_MostrarUnPedido(ePedido pedido, int modo);

/// @fn int ePedido\_MostrarPedidos(ePedido[], int, int)

/// @brief Muestra por consola los pedidos, dependiendo el modo mostrara pendientes o completados

///

/// @param pedidos -> Array de pedidos

/// @param tam -> Tamaño del array

/// @param modo -> Entero que indica que campos mostrara (1 - Pendiente o 2 - Completado)

/// @return Retorna 0 (No encontro resultados) o 1 (Encontro al menos un resultado)

int ePedido\_MostrarPedidos(ePedido pedidos[], int tam, int modo);

/// @fn int ePedido\_ProcesarResiduos(ePedido[], int)

/// @brief Procesa los pedidos pendientes separando el total de kg en tres tipos que se ingresaran

/// por consola

///

/// @param pedidos -> Array de pedidos

/// @param tam -> Tamaño del array

/// @return Retorna 1 (OK), 0 (Los parametros son incorrectos) o -1 (No se encontraron pedidos pendientes)

int ePedido\_ProcesarResiduos(ePedido pedidos[],int tam);

Biblioteca “Informes.h”:

/// @fn int Informes\_ImprimirClientesConPedidosPendientes(eCliente[], int, ePedido[], int, eLocalidad[], int, int)

/// @brief Imprime los clientes y la cantidad de pedidos pendientes que tiene cada uno

///

/// @param clientes -> Lista de clientes

/// @param tamClientes -> Tamaño de la lista

/// @param pedidos -> Lista de pedidos

/// @param tamPedidos -> Tamaño de la lista

/// @param localidades -> Lista de localidades

/// @param tamLocalidades -> Tamaño de la lista

/// @param ultimoPedido -> ultimo pedido cargado

/// @return -> Retorna 1 (OK), 0 (Error) o -1 (Sin pedidos)

int Informes\_ImprimirClientesConPedidosPendientes(eCliente clientes[], int tamClientes, ePedido pedidos[], int tamPedidos, eLocalidad localidades[], int tamLocalidades, int ultimoPedido);

/// @fn int Informes\_ImprimirPedidosPendientesConDatosCliente(ePedido[], int, eCliente[], int, eLocalidad[], int, int)

/// @brief Imprime la lista de pedidos pendientes con los datos del cliente

///

/// @param pedidos -> Lista de pedidos

/// @param tamPedidos -> Tamaño de la lista

/// @param clientes -> Lista de clientes

/// @param tamClientes -> Tamaño de la lista

/// @param localidades -> Lista de localidades

/// @param tamLocalidades -> Tamaño de la lista

/// @param ultimoPedido -> ultimo pedido cargado

/// @return -> Retorna 1 (OK), 0 (Error) o -1 (Sin pedidos)

int Informes\_ImprimirPedidosPendientesConDatosCliente(ePedido pedidos[], int tamPedidos, eCliente clientes[], int tamClientes, eLocalidad localidades[], int tamLocalidades, int ultimoPedido);

/// @fn int Informes\_ImprimirPedidosCompletadosConDatosCliente(ePedido[], int, eCliente[], int, eLocalidad[], int, int)

/// @brief Imprime la lista de pedidos completados con los datos del cliente (si no fue eliminado)

///

/// @param pedidos -> Lista de pedidos

/// @param tamPedidos -> Tamaño de la lista

/// @param clientes -> Lista de clientes

/// @param tamClientes -> Tamaño de la lista

/// @param localidades -> Lista de localidades

/// @param tamLocalidades -> Tamaño de la lista

/// @param ultimoPedido -> ultimo pedido cargado

/// @return -> Retorna 1 (OK), 0 (Error) o -1 (Sin pedidos)

int Informes\_ImprimirPedidosCompletadosConDatosCliente(ePedido pedidos[], int tamPedidos, eCliente clientes[], int tamClientes, eLocalidad localidades[], int tamLocalidades, int ultimoPedido);

/// @fn int Informes\_ImprimirPedidosPendientesSegunLocalidad(ePedido[], int, eCliente[], int, eLocalidad[], int, int)

/// @brief Le pide al usuario que ingrese una localidad y si hay pedidos pendientes los muestra

///

/// @param pedidos -> Lista de pedidos

/// @param tamPedidos -> Tamaño de la lista

/// @param clientes -> Lista de clientes

/// @param tamClientes -> Tamaño de la lista

/// @param localidades -> Lista de localidades

/// @param tamLocalidades -> Tamaño de la lista

/// @param ultimoPedido -> ultimo pedido cargado

/// @return -> Retorna 1 (OK), 0 (Error) o -1 (Sin pedidos)

int Informes\_ImprimirPedidosPendientesSegunLocalidad(ePedido pedidos[], int tamPedidos, eCliente clientes[], int tamClientes, eLocalidad localidades[], int tamLocalidades, int ultimoPedido);

/// @fn int Informes\_ImprimirPromedioKgPolipropilenoPorCliente(ePedido[], int, eCliente[], int, int, int)

/// @brief Calcula el promedio de kilos de polipropileno por cliente (si es que hay)

///

/// @param pedidos -> Array de pedidos

/// @param tamPedidos -> Tamaño del array pedidos

/// @param clientes -> Array de Clientes

/// @param tamClientes -> Tamaño del array Clientes

/// @param ultimoPedido -> Ultimo pedido realizado

/// @param ultimoCliente -> Ultimo cliente registrado

/// @return Retorna 1 (OK), 0 (Parametros incorrectos) o -1(No encontro pedidos)

int Informes\_ImprimirPromedioKgPolipropilenoPorCliente(ePedido pedidos[], int tamPedidos, eCliente clientes[], int tamClientes, int ultimoPedido, int ultimoCliente);

/// @fn int Informes\_ClientesConMasPedidos(ePedido[], int, eCliente[], int, int, int, int)

/// @brief

///

/// @param pedidos -> Array de pedidos

/// @param tamPedidos -> Tamaño del array pedidos

/// @param clientes -> Array de Clientes

/// @param tamClientes -> Tamaño del array Clientes

/// @param ultimoPedido -> Ultimo pedido realizado

/// @param ultimoCliente -> Ultimo cliente registrado

/// @param modo -> 1 (Pendientes), 2 (Completados), si se ingresa cualquier otro entero contara

/// pendientes y completados

/// @return Retorna 1 (OK), 0 (Parametros incorrectos) o -1(No encontro pedidos)

int Informes\_ClientesConMasPedidos(ePedido pedidos[], int tamPedidos, eCliente clientes[], int tamClientes, int ultimoPedido, int ultimoCliente, int modo);

Link del video explicativo:   
https://drive.google.com/file/d/1zV-WHEwnBR0OpNWBVpdM6R8UsV01GZnF/view?usp=sharing