Tobias Moretti
Div 1E
DNI 41668744

Video: https://drive.google.com/file/d/1md50jcfFuZPMLLUZh5y4IsqOAnu5kDcf/view?usp=sharing

Cliente.h

```
/// \fn int AltaCliente(eCliente[], eLocalidad[], int, int)
/// \brief Da de alta un cliente utilizando la lista de clientes y de localidades
/// \param listaClientes La lista de clientes
/// \param listaLocalidad La lista de localidades
/// \param tamCliente El tamaño de la lista de clientes
/// \param tamLocalidad El tamaño de la lista de localidades
/// \return retorna -1 en caso de error o 0 si salio bien
int AltaCliente(eCliente listaClientes[], eLocalidad listaLocalidad[],
           int tamCliente, int tamLocalidad);

/// \fn eCliente IngresarCliente(eLocalidad[], int)
/// \brief Ingresa un cliente con su localidad
/// \param listaLocalidad Lista de localidad a mostrar para ingresarla
/// \param tamLocalidad Tamaño de la lista de localidad
/// \return el cliente ingresado
eCliente IngresarCliente(eLocalidad listaLocalidad[], int
tamLocalidad);

/// \fn int BajaCliente(eCliente[], int)
/// \brief Da de baja un cliente logicamente utilizando el espacio isEmpty
/// \param listaClientes La lista de clientes a dar de baja
/// \param tamClientes El tamaño de la lista de clientes
/// \return -1 hubo error 0 si se dio de baja correctamente
int BajaCliente(eCliente listaClientes[], int tamClientes);

/// \fn int ModificarCliente(eCliente[], int, eLocalidad[], int)
/// \brief Modifica la localidad y la direccion de un cliente
/// \param listaClientes La lista de clientes a modificar
/// \param tamClientes El tamaño de la lista de clientes
/// \param listaLocalidad La lista de localidades
/// \param tamLocalidad El tamaño de la lista de localidades
/// \return -1 si hubo error 0 si se modifico correctamente
int ModificarCliente(eCliente listaClientes[], int tamClientes,
           eLocalidad listaLocalidad[], int tamLocalidad);
```

```
/// \fn int ListarCliente(eCliente[], eLocalidad[], int, int)
/// \brief Lista los clientes con su localidad
/// \param listaClientes La lista de clientes a mostrar
/// \param listaLocalidad La lista de localidades a mostrar
/// \param tamClientes El tamaño de la lista de clientes
/// \param tamLocalidad El tamaño de la lista de localidades
/// \return -1 si no hay elementos en la lista 0 si los hay
int ListarCliente(eCliente listaClientes[], eLocalidad
listaLocalidad[],
           int tamClientes, int tamLocalidad);

/// \fn int InicialilarListaCliente(eCliente[], int)
/// \brief Inicializa la lista de clientes usando el patron logico de
isEmpty
/// \param listaClientes La lista de clientes a inicializar
/// \param tamClientes El tamaño de la lista de clientes
/// \return -1 si hubo error 0 si se inicio correctamente
int InicialilarListaCliente(eCliente listaClientes[], int
tamClientes);

/// \fn void MostrarCliente(eCliente, char[])
/// \brief Muestra un cliente con su localidad
/// \param unCliente El cliente a mostrar
/// \param nombreLocalidad El nombre de la localidad a mostrar
void MostrarCliente(eCliente unCliente, char nombreLocalidad[]);

/// \fn int BuscarIsEmpty(eCliente[], int, int*)
/// \brief Busca el espacio en la lista de clientes
/// que este empty y retorna su posicion por puntero
/// \param listaClientes La lista de clientes a buscar
/// \param tamClientes El tamaño de la lista de clientes
/// \param posicion La posicion en donde se encuentre vacio
/// \return -1 si hubo error 0 si se encontro
int BuscarIsEmpty(eCliente listaClientes[], int tamClientes, int
*posicion);

/// \fn int BuscarIdCliente(eCliente[], int, int, int*)
/// \brief Busca el id de un cliente y devuelve su posicion por
puntero
/// \param listaClientes La lista de clientes a buscar
/// \param tamClientes El tamaño de la lista de clientes
/// \param valorBuscado El id a buscar
/// \param posicion La posicion a buscar
/// \return -1 si hubo error 0 si lo encontro
int BuscarIdCliente(eCliente listaClientes[], int tamClientes, int
valorBuscado, int *posicion);

/// \fn void MostrarLocalidad(eLocalidad)
/// \brief Muestro una localidad
/// \param unaLocalidad La localidad a mostrar
void MostrarLocalidad(eLocalidad unaLocalidad);
```

```
/// \fn int IngresarLocalidad(eLocalidad[], int)
/// \brief Ingresa una localidad por un id que debe ser ingresado
/// \param listaLocalidad La lista de localidades
/// \param tamLocalidad El tamaño de la lista de localidades
/// \return 0 si hubo error O el idLocalidad ingresado
int IngresarLocalidad(eLocalidad listaLocalidad[], int tamLocalidad);

/// \fn int GenerarIdCliente()
/// \brief Genera un id de cliente auto incremental
/// \return El id generado
int GenerarIdCliente();


/// \fn int BuscarIdLocalidad(eLocalidad[], int, int, int*)
/// \brief Busca el id de la localidad
/// segun un parametro a buscar y la devuelve a modo puntero
/// \param listaLocalidad La lista de localidades a buscar
/// \param tamLocalidad El tamaño de la lista de localidad
/// \param valorBuscado El parametro a buscar
/// \param posicion La posicion donde se encuentra
/// \return -1 si hubo error 0 si logro encontrarla
int BuscarIdLocalidad(eLocalidad listaLocalidad[], int tamLocalidad,
        int valorBuscado, int *posicion);

/// \fn int ClientesTotales(eCliente[], int)
/// \brief Cuenta cuantos clientes hay en la lista
/// \param listaClientes La lista de clientes a contrar
/// \param tamClientes El tamaño de la lista de clientes
/// \return La cantidad de clientes
int ClientesTotales(eCliente listaClientes[], int tamClientes);


/// \fn eCliente ObtenerClientePorId(eCliente[], int, int)
/// \brief Obtiene el cliente por el id
/// \param listaClientes La lista de clientes a obtener
/// \param tamCliente El tamaño de la lista de clientes
/// \param id El id para buscar
/// \return el cliente obtenido
eCliente ObtenerClientePorId(eCliente listaClientes[], int tamCliente,
int id);

/// \fn int NombrePorIdLocalidad(char[], eLocalidad[], int, int)
/// \brief Busca el nombre de la localidad por su id
/// \param nombre El nombre a buscar
/// \param listaLocalidad La lista de localidades a buscar
/// \param tamLocalidad El tamaño de la lista de localidades
/// \param idLocalidad El id de la localidad que se busca
/// \return -1 si hubo error 0 si la encontro
int NombrePorIdLocalidad(char nombre[], eLocalidad listaLocalidad[],
        int tamLocalidad, int idLocalidad);
```

```
/// \fn void MostrarClienteCantidad(eCliente, char[], int)
/// \brief Muestra un cliente con la cantidad de pedidos
/// \param unCliente El cliente a mostrar
/// \param nombreLocalidad El nombre de la localidad a mostrar
/// \param cantidad La cantidad a mostrar
void MostrarClienteCantidad(eCliente unCliente, char
nombreLocalidad[], int cantidad);
```

Pedido.h

```
/// \fn int InicializaPedido(ePedido[], int)
/// \brief Inicializa la lista pedidos utilizando el valor logico
isEmpty
/// \param listaPedido La lista de pedidos a inicializar
/// \param tamPedido El tamaño de la lista de pedidos
/// \return -1 si hubo error 0 si salio bien
int InicializaPedido(ePedido listaPedido[], int tamPedido);

/// \fn int BuscarLugarPedido(ePedido[], int, int*)
/// \brief Busca un lugar en la lista de pedidos y los devuelve a modo
puntero
/// \param listaPedido La lista de pedidos a buscar
/// \param tamPedido El tamaño de la lista de pedidos
/// \param posicion La posicion donde hay un lugar
/// \return -1 si hubo error 0 si encontro lugar
int BuscarLugarPedido(ePedido listaPedido[], int tamPedido, int
*posicion);

/// \fn int CrearPedido(ePedido[], int)
/// \brief Crea un pedido solo si hay lugar en la lista de pedidos
/// \param listaPedido La lista de pedidos a crear
/// \param tamPedido El tamaño de la lista de pedidos
/// \return -1 si hubo error 0 si salio bien
int CrearPedido(ePedido listaPedido[], int tamPedido);

/// \fn int ProcesarPedido(ePedido[], int)
/// \brief Procesa un pedido que debe ser ingresado por el id
/// \param listaPedido La lista de pedidos
/// \param tamPedido el tamaño de la lista
/// \return -1 si hubo error 0 si se pudo procesar
int ProcesarPedido(ePedido listaPedido[], int tamPedido);

/// \fn int BuscarPedido(ePedido[], int, int, int*)
/// \brief Busca un pedido por un valor y devuelve su posicion a modo
puntero
/// \param listaPedido La lista de pedidos a buscar
/// \param tamPedido El tamaño de la lista de pedidos
/// \param valorBuscado El valor a buscar
/// \param posicion La posicion donde se encuentra
/// \return -1 si hubo error 0 si se encontro
int BuscarPedido(ePedido listaPedido[], int tamPedido, int
valorBuscado, int *posicion);
```

```
/// \fn int ListarPedidos(ePedido[], int)
/// \brief Lista los pedidos con todos sus datos
/// \param listaPedido Lista de pedidos
/// \param tamPedido Tamaño de la lista de pedidos
/// \return -1 si la lista esta vacia 0 si no lo esta
int ListarPedidos(ePedido listaPedido[], int tamPedido);

/// \fn void MostrarPedidos(ePedido)
/// \brief Muestra un pedido
/// \param unPedido El pedido a mostrar
void MostrarPedidos(ePedido unPedido);

/// \fn int GenerarIdPedido()
/// \brief Genera un id autoincremental para los pedidos
/// \return El id
int GenerarIdPedido();

/// \fn ePedido ObtenerPedidoPorIdPedido(ePedido[], int, int)
/// \brief Obtiene un pedido en base a un id
/// \param listaPedidos La lista de pedidos a obtener
/// \param tamPedido El tamaño de la lista de pedidos
/// \param idPedido El id a obtener
/// \return el pedido obtenido
ePedido ObtenerPedidoPorIdPedido(ePedido listaPedidos[], int
tamPedido, int idPedido);

/// \fn int CantidadPedidosPorIdCliente(ePedido[], int, int)
/// \brief Calcula la cantidad de pedidos que hay por el id del
cliente
/// \param listaPedidos La lista de pedidos
/// \param tamPedidos El tamaño de la lista de pedidos
/// \param idCliente El id del cliente
/// \return La cantidad de pedidos del cliente
int CantidadPedidosPorIdCliente(ePedido listaPedidos[], int
tamPedidos, int idCliente);

/// \fn void MostrarPedidoPendiente(char[], char[], int)
/// \brief Muestra los pedidos pendientes con datos del cliente
/// \param unCuit El cuit del cliente
/// \param unaDireccion La direccion del cliente
/// \param Kilos Los kilos totales del cliente
void MostrarPedidoPendiente(char unCuit[], char unaDireccion[], int
Kilos);
```

```
/// \fn void MostrarPedidoCompletado(char[], char[], int, int, int)
/// \brief Muestra los datos del pedido procesado con los datos de
cliente
/// \param unCuit El cuit del cliente
/// \param unaDireccion La direccion del ciente
/// \param HDPE Los kilos de HDPE
/// \param LDPE Los kilos de LDPE
/// \param PP Los kilos de PP
void MostrarPedidoCompletado(char unCuit[], char unaDireccion[], int
HDPE, int LDPE, int PP);

/// \fn int AcumularKilos(ePedido[], int)
/// \brief Acumula todos los kilos de la lista de pedidos
/// \param listaPedido La lista de pedidos
/// \param tamPedido El tamaño de la lista de pedidos
/// \return La cantidad de kilos totales
int AcumularKilos(ePedido listaPedido[], int tamPedido);
```

Informes.h

```
/// \fn int ListarPedidosClientes(ePedido[], eCliente[], eLocalidad[],
int, int, int)
/// \brief Lista los clientes por pedidos pendientes y su cantidad
/// \param listaPedido La lista de pedidos
/// \param listaCliente La lista de clientes
/// \param listaLocalidad La lista de localidades
/// \param tamPedido El tamaño de la lista de pedidos
/// \param tamCliente El tamaño de la lista de clientes
/// \param tamLocalidad El tamaño de la lista de localidades
/// \return -1 si la lista esta vacia 0 si no lo esta
int ListarPedidosClientes(ePedido listaPedido[], eCliente
listaCliente[],eLocalidad listaLocalidad[], int tamPedido, int
tamCliente, int tamLocalidad);

/// \fn int ListarPedidosClientesPendientes(ePedido[], eCliente[],
int, int)
/// \brief Muestra los pedidos pendientes
/// \param listaPedido La lista de pedidos
/// \param listaCliente La lista de clientes
/// \param tamPedido El tamaño de la lista de pedidos
/// \param tamCliente El tamaño de la lista de clientes
/// \return -1 si la lista esta vacia 0 si no lo esta
int ListarPedidosClientesPendientes(ePedido listaPedido[],
            eCliente listaCliente[], int tamPedido, int tamCliente);
```

```
/// \fn int ListarPedidosClientesProcesados(ePedido[], eCliente[],
int, int)
/// \brief Muestra los pedidos procesados
/// \param listaPedido La lista de pedidos
/// \param listaCliente La lista de clientes
/// \param tamPedido El tamaño de la lista de pedidos
/// \param tamCliente El tamaño de la lista de clientes
/// \return -1 si la lista esta vacia 0 si no lo esta
int ListarPedidosClientesProcesados(ePedido listaPedido[],eCliente
listaCliente[], int tamPedido, int tamCliente);

/// \fn int CantidadPedidosPendientes(ePedido[], int, int)
/// \brief Calcula la cantidad de pedidos pendientes que tiene el
cliente
/// \param listaPedido La lista de pedidos
/// \param tamPedido El tamaño de la lista de pedidos
/// \param idCliente El id del cliente a calcular
/// \return la cantidad de pedidos pendientes
int CantidadPedidosPendientes(ePedido listaPedido[], int tamPedido,
            int idCliente);

/// \fn int CantidadPedidosPorLocalidad(ePedido[], eCliente[],
eLocalidad[], int, int, int)
/// \brief Ingresa una localidad y calcula
/// la cantidad de pedidos pendientes de la misma
/// \param listaPedido La lista de pedidos
/// \param listaCliente La lista de clientes
/// \param listaLocalidad La lista de localidades
/// \param tamPedido El tamaño de la lista de pedidos
/// \param tamCliente El tamaño de la lista de clientes
/// \param tamLocalidad El tamaño de la lista de localidades
/// \return la cantidad de esa localidad
int CantidadPedidosPorLocalidad(ePedido listaPedido[], eCliente
listaCliente[],eLocalidad listaLocalidad[], int tamPedido, int
tamCliente, int tamLocalidad);

/// \fn int KilosPromedioPorCliente(ePedido[], eCliente[], int, int)
/// \brief Calcula el promedio de kilos de todos los clientes
/// \param listaPedido La lista de pedidos
/// \param listaCliente La lista de clientes
/// \param tamPedido El tamaño de la lista de pedidos
/// \param tamCliente El tamaño de la lista de clientes
/// \return -1 si hubo error 0 si se calculo correctamente
int KilosPromedioPorCliente(ePedido listaPedido[], eCliente
listaCliente[],int tamPedido, int tamCliente);
```

```
/// \fn int MostrarClienteMasPedientes(ePedido[], eCliente[],
eLocalidad[], int, int, int)
/// \brief Muestra el cliente o los clientes con mas pedidos
pendientes
/// \param listaPedido La lista de pedidos
/// \param listaCliente La lista de clientes
/// \param listaLocalidad La lista de localidades
/// \param tamPedido El tamaño de la lista de pedidos
/// \param tamCliente El tamaño de la lista de clientes
/// \param tamLocalidad El tamaño de la lista de localidades
/// \return -1 si la lista esta vacia 0 si no lo esta
int MostrarClienteMasPedientes(ePedido listaPedido[], eCliente
listaCliente[],eLocalidad listaLocalidad[], int tamPedido, int
tamCliente, int tamLocalidad);



/// \fn int MostrarClienteMasProcesados(ePedido[], eCliente[],
eLocalidad[], int, int, int)
/// \brief Muestra el cliente o los clientes con mas pedidos
procesados
/// \param listaPedido La lista de pedidos
/// \param listaCliente La lista de clientes
/// \param listaLocalidad La lista de localidades
/// \param tamPedido El tamaño de la lista de pedidos
/// \param tamCliente El tamaño de la lista de clientes
/// \param tamLocalidad El tamaño de la lista de localidades
/// \return -1 si la lista esta vacia 0 si no lo esta
int MostrarClienteMasProcesados(ePedido listaPedido[], eCliente
listaCliente[],eLocalidad listaLocalidad[], int tamPedido, int
tamCliente,int tamLocalidad);

/// \fn int MostrarClienteMasPedidos(ePedido[], eCliente[],
eLocalidad[], int, int, int)
/// \brief Muestra el cliente o los clientes con mas pedidos
/// \param listaPedido La lista de pedidos
/// \param listaCliente La lista de clientes
/// \param listaLocalidad La lista de localidades
/// \param tamPedido El tamaño de la lista de pedidos
/// \param tamCliente El tamaño de la lista de clientes
/// \param tamLocalidad El tamaño de la lista de localidades
/// \return -1 si la lista esta vacia 0 si no lo esta
int MostrarClienteMasPedidos(ePedido listaPedido[], eCliente
listaCliente[],eLocalidad listaLocalidad[], int tamPedido, int
tamCliente, int tamLocalidad);
```

Menu.h

```
/// \fn void MenuOpciones(ePedido[], eCliente[], eLocalidad[], int,
int, int)
/// \brief Menu principal donde se encuentran las opciones principales
/// \param listaPedido La lista de pedidos
/// \param listaCliente La lista de clientes
/// \param listaLocalidad La lista de localidades
/// \param tamPedido El tamaño de la lista de pedidos
/// \param tamCliente El tamaño de la lista de clientes
/// \param tamLocalidad El tamaño de la lista de localidades
void MenuOpciones(ePedido listaPedido[], eCliente listaCliente[],
eLocalidad listaLocalidad[], int tamPedido, int tamCliente,
int tamLocalidad);

/// \fn int MenuImprimir(ePedido[], eCliente[], eLocalidad[], int,
int, int)
/// \brief Menu donde se encuentran las opciones para imprimir
/// \param listaPedido La lista de pedidos
/// \param listaCliente La lista de clientes
/// \param listaLocalidad La lista de localidades
/// \param tamPedido El tamaño de la lista de pedidos
/// \param tamCliente El tamaño de la lista de clientes
/// \param tamLocalidad El tamaño de la lista de localidades
/// \return -1 si hubo error 0 si salio bien
int MenuImprimir(ePedido listaPedido[], eCliente listaCliente[],
      eLocalidad listaLocalidad[], int tamPedido, int tamCliente,
int tamLocalidad);

/// \fn int MenuInformes(ePedido[], eCliente[], eLocalidad[], int,
int, int)
/// \brief Menu con las opciones para los informes
/// \param listaPedido La lista de pedidos
/// \param listaCliente La lista de clientes
/// \param listaLocalidad La lista de localidades
/// \param tamPedido El tamaño de la lista de pedidos
/// \param tamCliente El tamaño de la lista de clientes
/// \param tamLocalidad El tamaño de la lista de localidades
/// \return -1 si hubo error 0 si salio bien
int MenuInformes(ePedido listaPedido[], eCliente listaCliente[],
      eLocalidad listaLocalidad[], int tamPedido, int tamCliente,
int tamLocalidad);
```