



Recuperatorio Laboratorio

26/11/2021

Fermín Gonzalez

División B

eMail : fermingonzalez011@gmail.com

DNI : 43322850

[Enlace Google drive](#)

Prototipo de cada función del núcleo del programa con la documentación correspondiente :

Alta Clientes.

fn int cliente_alta(eCliente[], int, int*, eLocalidad[], int, eZona[], int)

Brief Busca un espacio libre y da de alta un empleado pidiéndole los datos necesarios, además se relaciona con la estructura localidad y zonas , donde se muestra una lista con las localidades disponibles y da la opción al usuario para elegir cada una de ellas.

param listaClientes eCliente array de clientes.

param lenClientes int tamaño del array de clientes.

param refId int* Id auto incremental que se le asigna al cliente.

param listaLocalidades eLocalidad array de localidades.

param lenLocalidades int tamaño del array de localidades.

param listaZonas eZona array de zonas.

param lenZonas int tamaño del array de zonas.

return ERROR(-1) si hubo un error o FUNCIONÓ(0) si todo estuvo bien.

```
//int cliente_alta(eCliente listaClientes[], int lenClientes, int* refId, eLocalidad  
listaLocalidades[], int lenLocalidades);
```

Alta Pedidos.

```
fn int pedido_alta(ePedido[], int, eCliente[], int, eLocalidad[], int, ePago[], int, int*, int*)
```

Brief Muestra el listado de clientes y el usuario puede elegir uno para crearle un pedido, luego rellena los campos del pedido y también da de alta al pago.

param listaPedidos ePedido array de pedidos.

param lenPedidos int tamaño del array de pedidos.

param listaClientes eCliente array de clientes.

param lenClientes int tamaño del array de clientes.

param listaLocalidades eLocalidades array de localidades.

param lenLocalidades int tamaño del array de localidades.

param listaPagos ePago array de pagos.

param lenPagos int tamaño del array de pagos.

param refIdPagos int* ID autoincremental que se asigna al pago.

param refIdPedido int* ID autoincremental que se asigna al pedido.

return ERROR(-1) si hubo un error o FUNCIONÓ(0) si todo estuvo bien.

```
//int pedido_alta(ePedido listaPedidos[], int lenPedidos, eCliente listaClientes[], int
lenClientes, eLocalidad listaLocalidades[], int lenLocalidades, ePago listaPagos[], int
lenPagos, int* refldPagos , int* refldPedido);
```

Procesar Residuos.

```
fn int pedido_procesarResiduos(ePedido[], int, eCliente[], int, ePago[], int)
```

Brief Imprime los pedidos pendientes y el usuario tiene la opción de elegir cual procesar, luego se recorre la lista de pagos en busca del pedido que le corresponde y por último modifica los campos de ambos respectivamente. param listaPedidos ePedido array de pedidos.

param lenPedidos int tamaño del array de pedidos.

param listaClientes eCliente array de clientes.

param lenClientes int tamaño del array de clientes.

param listaPagos ePago array de pagos.

param lenPagos int tamaño del array de pagos.

return ERROR(-1) si hubo un error o FUNCIONÓ(0) si todo estuvo bien.

```
//int pedido_procesarResiduos(ePedido listaPedidos[], int lenPedidos, eCliente
listaClientes[], int lenClientes, ePago listaPagos[], int lenPagos);
```

Alta de Pagos.

```
fn int pago_alta(ePago[], int, int, float, int)
```

brief Busca un espacio libre en el array de pagos y rellena los campos de ese pago.

param listaPagos ePago array de pagos.

param lenPagos int tamaño del array de pagos.

param auxIdPedido int ID del pedido que se le va a asignar al pago.
param auxKilosBasura float Kilos de basura del pedido que se van a multiplicar por el precio por kg de basura.
param auxIdPago ID autoincremental que se le asigna al pago.
 return ERROR(-1) si hubo un error o FUNCIONÓ(0) si todo estuvo bien.
 //int pago_alta(ePago listaPagos[], int lenPagos, int auxIdPedido, float auxKilosBasura, int auxIdPago).

Enunciado del informe anexo :

13) Cantidad de dinero liquidado promedio en clientes por zona.

fn int promedio_pagosLiquidadosPorZona(eCliente[], int, ePedido[], int, eLocalidad[], int, ePago[], int)

brief Calcula el promedio de pagos liquidado de la zona que el usuario elija. Primero recorre la lista de localidades, contando la cantidad de localidades que cumplen con la condición. Luego recorre la lista de pedidos , en busca del index, de los que les pertenece los barrios. Y por último recorre la lista de pagos, acumulando la cantidad de dinero liquidado, e imprime el promedio por pantalla.

param listaClientes eCliente array de clientes.

param lenClientes int tamaño del array de clientes.

param listaPedidos ePedido array de pedidos.

param lenPedidos int tamaño del array de pedidos.

param listaLocalidades eLocalidad array de localidades.


param lenLocalidades int tamaño del array de localidades.

param listaPagos ePago array de pagos.

param lenPagos int tamaño del array de pagos.

param listaZonas eZona array de zonas.

param lenZonas int tamaño del array de pagos.



```
return ERROR(-1) si hubo un error o FUNCIONÓ(0) si todo estuvo bien.  
int promedio_pagosLiquidadosZonaSur(eCliente listaClientes[], int lenClientes,  
ePedido listaPedidos[], int lenPedidos, eLocalidad listaLocalidades[], int  
lenLocalidades, ePago listaPagos[], int lenPagos);
```