**ALTA DE CLIENTE**

**int AltaCliente(int\* idAuto, eCliente lista[],int tam)**

**{**

**int altaOk;**

**int index;**

**if(lista != NULL && tam >0)**

**{**

**index=BuscarEspacioVacio(lista,tam);**

**if(index!= -1)**

**{**

**lista[index] = CargarCliente(idAuto);**

**mostrarUnCliente(lista[index]);**

**lista[index].isEmpty=OCUPADO;**

**altaOk=1;**

**}**

**}**

**return altaOk;**

**}**

**MODIFICAR CLIENTES**

**int** modificarClientes(eCliente lista[], **int** tam) {

**int** retorno = 0;

**if** (modificarUnCliente(lista, tam) == 1) {

printf("\n La modificacion ha sido exitosa \n");

retorno=1;

}

**return** retorno;

}

MODIFICAR UN CLIENTE

**int** modificarUnCliente(eCliente lista[], **int** tam) {

**int** retorno = 0;

**int** IdCliente;

**int** index = -1;

**int** opcion;

mostrarListaClientes(lista, tam);

IdCliente = getValidInt("Ingrese el ID de Cliente a modificar",

"ERROR. Intente nuevamente", 1000, 2000);

index = buscarClientePorId(lista, tam, IdCliente);

**if** (lista != NULL && tam > 0) {

**if** (index != -1) {

**do** {

opcion = menuModificar();

printf(

"Ingrese una opcion 1)Modificar Direccion - 2) Modificar Localdiad - 3) SALIR");

**switch** (opcion) {

**case** 1:

printf("\n 1) Usted selecciono modificar direccion ");

getStringAlfaNumerico("\n Ingrese la nueva direccion",

"\n Error. Intente nuevamente",

lista[index].direccion, tam, 3);

**break**;

**case** 2:

printf("\n 2) Usted selecciono modificar Localidad");

getStringSoloLetras("\n Ingrese la Nueva Localidad",

"\n Error. Intente nuevamente",

lista[index].localidad, tam, 3);

**break**;

**case** 3:

printf("\n 3) Hasta luego");

retorno = 0;

**break**;

}

} **while** (opcion != 3);

retorno = 1;

}

}

**return** retorno;

}

**BAJA DE CLIENTE**

**int** BajaDeCliente(eCliente lista[], **int** TAM) {

**int** IdCliente;

**int** retorno = -1;

**char** respuesta;

**int** index;

mostrarListaClientes(lista, TAM);

IdCliente = getValidInt("Ingrese el ID de Cliente a Eliminar",

"Error. Ingrese un ID valido", 0, 1000);

printf("Esta seguro de ELIMINAR el Cliente %d ? s/n", IdCliente);

fflush(stdin);

scanf("%c", &respuesta);

**if** (respuesta == 's') {

{

index = buscarClientePorId(lista, TAM, IdCliente);

**if** (index != -1) {

lista[index].isEmpty = VACIO;

retorno = 1;

printf("Se ha eliminado el ID %d correctamente \n", IdCliente);

} **else** {

printf("ID incorrecto, intente nuevamente \n");

}

}

**while** (respuesta != 's' && respuesta != 'n') {

printf(

"\n ERROR. SOLO s/n. Esta seguro de ELIMINAR el Cliente %d ? s/n \n",

IdCliente);

scanf("%c", &respuesta);

}

}

**return** retorno;

}

**ALTA DE PEDIDO**

**int** altaPedido(eCliente lista[], ePedidos pedido[], eTransporte medio[],**int** max, **int** top, **int** tam, **int**\* idAutoPedido)

{

**int** retorno=0;

**int** index;

**int** IdCliente;

mostrarListaClientes(lista, tam);

IdCliente = getValidInt("\n Ingrese el ID de Cliente a Cargar el Pedido \n","\n Error. Ingrese un ID valido \n ", 0, 5000);

**if**(buscarClientePorId(lista, tam, IdCliente)!=-1)

{

index = buscarEspacioPedido(pedido, top);

**if**(index!=-1){

cargaDeUnPedido(pedido, index);

pedido[index].estadoPedido=PENDIENTE;

\*idAutoPedido = \*idAutoPedido + 1;

pedido[index].idCliente=IdCliente;

pedido[index].idPedido=\*idAutoPedido;

pedido[index].isEmpty=FULL;

retorno=1;

**if**(determinarTransporte(medio, max, &(pedido[index]) ) != 1)

{

printf("No se han cargado los kilos, se eliminara el pedido\n");

pedido[index].isEmpty=VACIO;

}

}**else**{

printf("\n No Existe ese ID de Cliente \n");

}

}

**return** retorno;

}

**5 PROCESAR RESIDUOS**

**int** procesarResiduos(ePedidos pedido[], **int** top) {

**int** retorno = 0;

**int** index;

**int** opcion;

**int** acumPeso = 0;

**int** IdPedido;

mostrarListaPedidos(pedido, top);

IdPedido = getValidInt("Ingrese el ID de Pedido a Procesar",

"Error. Ingrese un ID de Pedido valido", 0, 1000);

index = validarEstadoPedido(pedido, top, IdPedido);

**if** (validarEstadoPedido(pedido, top, IdPedido) != -1) {

printf("La cantidad de KG TOTALES es de %f", pedido[index].kilos);

printf("\n Elija: 1- Cargar HDPE \t 2- Cargar LDPE \t 3- Cargar PP \t 4- SALIR ");

**do** {

opcion = subMenu();

**switch** (opcion) {

**case** 1:

printf("\n 1- Ingrese SOLO los kilos de HDPE");

scanf("%f", &pedido[index].HDPE);

acumPeso += pedido[index].HDPE;

retorno = 1;

**break**;

**case** 2:

printf("\n 2- Ingrese SOLO los kilos de LDPE");

scanf("%f", &pedido[index].LDPE);

acumPeso += pedido[index].LDPE;

retorno = 1;

**break**;

**case** 3:

printf("\n 3- Ingrese SOLO los kilos de PP");

scanf("%f", &pedido[index].PP);

acumPeso += pedido[index].PP;

retorno = 1;

**break**;

**case** 4:

printf("\n 4- SALIR");

}

} **while** (opcion != 4);

}

**if** (acumPeso > pedido[index].kilos) {

retorno = 0;

pedido[index].estadoPedido = PENDIENTE;

} **else** {

retorno = 1;

pedido[index].estadoPedido = COMPLETADO;

}

**return** retorno;

}

**6 MOSTRAR CLIENTES CANT PENDIENTE**

**int** mostrarClientes\_CantidadPendiente(eCliente lista[], **int** tam, ePedidos pedido[], **int** top){

**int** retorno=-1;

**int** i;

**int** acumP;

acumP=0;

**if**(lista!=NULL && pedido !=NULL && tam>0 && top>0){

printf("Paso el NULL \n");

**for**(i=0;i<tam;i++){

**if**(lista[i].isEmpty==OCUPADO){

acumP=contarPendiente\_PorCliente(pedido, top, lista[i].idCliente);

printf("%d", acumP);

**if**(acumP>0){

mostrarUnClientePendiente(lista[i], pedido, top, acumP);

retorno=1;

}

}

}

}

**return** retorno;

}

**7 CLIENTE CANT KG PENDIENTE**

**int** imprimirKg\_Pendientes(eCliente lista[], **int** tam, ePedidos pedido[], **int** top){

**int** retorno=-1;

**int** i;

**float** acumKg;

acumKg=0;

**if**(lista!=NULL && pedido !=NULL && tam>0 && top>0){

printf("Paso el NULL \n");

**for**(i=0;i<tam;i++){

**if**(lista[i].isEmpty==OCUPADO){

acumKg=contarKgPendiente\_PorCliente(pedido, top, lista[i].idCliente);

**if**(acumKg>0){

imprimirPedidosPendientes(lista[i], pedido, top, acumKg);

retorno=1;

}

}

}

}

**return** retorno;

}

**8 LISTA PEDIDOS COMPLETADO CON KG**

**int** imprimirKg\_Completados(eCliente lista[], **int** tam, ePedidos pedido[], **int** top){

**int** retorno=-1;

**int** i;

**float** acumHDPE=0;

**float** acumLDPE=0;

**float** acumPP=0;

**if**(lista!=NULL && pedido !=NULL && tam>0 && top>0){

**for**(i=0;i<tam;i++){

**if**(lista[i].isEmpty==OCUPADO){

acumHDPE=contarHDPE\_PorCliente(pedido, top, lista[i].idCliente);

acumLDPE=contarLDPE\_PorCliente(pedido, top, lista[i].idCliente);

acumPP=contarPP\_PorCliente(pedido, top, lista[i].idCliente);

**if**(acumHDPE>=0 && acumLDPE>=0 && acumPP>=0){

imprimirPedidosCompletados(lista[i], pedido, top, acumHDPE, acumLDPE, acumPP);

retorno=1;

}

}

}

}

**return** retorno;

}

**9 PEDIDOS PROCESADOS POR LOCALIDAD**

**int** pendientesPorLocalidad(eCliente lista[], **int** tam, ePedidos pedido[],**int** top) {

**int** retorno = 0;

**int** i;

**int** contadorPendientes = 0;

**char** auxLocalidad[51];

getValidString("\n Ingrese la localidad de la Empresa \n",

"\n Error, ingrese SOLO letras \n", auxLocalidad, 51);

**if** (lista != NULL && pedido != NULL && tam > 0 && top > 0) {

**for** (i = 0; i < tam; i++) {

**if** (lista[i].isEmpty == OCUPADO

&& strcmp(auxLocalidad, lista[i].localidad) == 0) {

**if** (lista[i].idPedido == pedido[i].idPedido) {

contadorPendientes++;

retorno = 1;

}

}

}

}

**if** (contadorPendientes > 0) {

printf("\n Los pedidos pendientes para la localidad %s son %d \n",

auxLocalidad, contadorPendientes);

} **else** {

printf("\n No hay pedidos pendientes para la localidad %s \n",

auxLocalidad);

}

**return** retorno;

}

**10 KG TOTAL PP/CLIENTES**

**int** promedio\_Clientes(eCliente lista[], **int** tam, ePedidos pedido[], **int** top) {

**int** retorno = -1;

**int** acumPP;

**int** contC;

**float** promedio = 0;

**int** i;

acumPP = 0;

contC = 0;

**if** (lista != NULL && tam > 0 && pedido != NULL && top > 0) {

**for** (i = 0; i < tam; i++) {

**if** (lista[i].isEmpty == OCUPADO && lista[i].idPedido==pedido[i].idPedido) {

**if**(pedido[i].estadoPedido==COMPLETADO){

contC++;

acumPP= contarPP\_Clientes( pedido, top, lista[i].idCliente);

}

}

}

}

**if** (contC > 0 && acumPP > 0) {

promedio = acumPP / contC;

printf("El promedio de kg de polipropileno reciclado por cliente es %f \n", promedio);

retorno=1;

}**else**{

printf("No hubo PP reciclado \n");

}

**return** retorno;

}

**11 DETERMINAR MAYOR TRANSPORTE UTILIZADO X EMPRESA**

**int** mayorMedio\_Transporte(ePedidos pedido[], **int** top, eTransporte medio[], **int** max, eCliente lista[], **int** tam){

**int** retorno;

**int** i;

**int** idCliente;

**int** indexIdCliente=-1;

**int** acumT;

acumT=0;

**if**(pedido!=NULL && medio!=NULL && top>0 && max>0)

{

**for**(i=0;i<max;i++)

{

acumT=contador\_TransportePorId(medio[i].idTransporte, pedido, top, lista, tam,&idCliente);

indexIdCliente=buscarClientePorId(lista, tam, idCliente);

**if**(acumT>0)

{

printf("\n|ID CL %4d |EMPRESA %-18s |TRANSPORTE %-18s|VECES UTILIZADO %4d| \n",lista[indexIdCliente].idCliente, lista[indexIdCliente].empresa, medio[i].medioTransporte, acumT);

retorno=1;

}

}

}

**return** retorno;

}

**int** determinarTransporte(eTransporte medio[],**int** max,ePedidos\* unPedido)

{

**int** retorno=-1;

**int** i;

**if**(medio!=NULL)

{

**if**(unPedido->estadoPedido==PENDIENTE)

{

**for**(i=0;i<max;i++)

{

**if**(unPedido->kilos>0)

{

**if**(unPedido->kilos<500)

{

unPedido->idTransporte=medio[i].idTransporte;

}**else** **if**(unPedido->kilos<1100)

{

unPedido->idTransporte=medio[i].idTransporte;

}**else**

{

unPedido->idTransporte=medio[i].idTransporte;

}

retorno =1;

}

}

}

}

**return** retorno;

}

**int** cant\_PedidosPorCliente(ePedidos lsPedidos[], **int** tam, **int** idCliente){

**int** retorno=-1;

**int** cont;

**int** i;

cont=0;

**if**(lsPedidos!=NULL && tam>0)

{

**for**(i=0;i<tam;i++){

**if**(lsPedidos[i].idCliente==idCliente){

cont++;

}

}

retorno=cont;

}

**return** retorno;

}

RECUPERATORIO PARTE I

<https://drive.google.com/file/d/1LbnzkJunn6w-MgERxOVLvR9zjjWgsxB4/view?usp=sharing>

RECUPERATORIO PARTE II <https://drive.google.com/file/d/1VWfov9zvMh7eEptb2cy6Iv6MjCeXsGt-/view?usp=sharing>