```
Buscar medicamento (Codigo, this) {

  Si (Nodo tmp == Arot)
    Mientras (tmp) {
        Si (tmp → lote1) {
            Si (codigo == codigo en lote1) {
                Devuelve medicamento
            }
        }
        Si(tmp → lote1) {
            Si (codigo == codigo en lote 2) {
                Devuelve medicamento
            }
        }

    tmp = tmp → next
    }
  Devuelve NULL
}
```

```
Bool Remove - Producto: (DLL* lote, char* nombre, int cantidad)
{
  si (no existe lote) { devuelve FALSE }
  Medicamento* tmp = buscar_lote(nombre, lote)
  si (tmp -> lote1.cantidad + tmp -> lote2.cantidad >= cantidad) {
    si(Remover_Medicamento_Unidades(tmp, cantidad)) {
      devuelve TRUE
    }
    else {
      devuelve FALSE }
```

Remove_Producto_reducido (this → DLL*) → Bool

{

si (la lista no existe) { return FALSE }

abrir archivo Fpt_Ale = Fopen ("Alerta.csv", rt)

si (no se pudo abrir) { Imprimir "Alerta", return FALSE }

~~[illegible]~~ Crear codigo

Crear buffer

Leer primera linea del archivo

mientras (! Feof (Fpt_Ale)) {

Extraer codigo de la linea del buffer y guardar en ~~[illegible]~~ codigo

~~[illegible]~~ Buscar codigo en ~~[illegible]~~ The registro medicamento

~~[illegible]~~ si (medicamento → lot 1 → codigo == codigo) {

medicamento → lot 1 = ~~[illegible]~~ NULL medicamento → lot 2

medicamento → lot 2 = NULL

}

~~[illegible]~~ sino si (medicamento → lot 2 → codigo == codigo) {

~~[illegible]~~ medicamento → lot 2 = NULL }

Leer siguiente linea del archivo

} // Fin mientras

Se limpia el archivo Alerta

Cerrar archivo

destruir ~~[illegible]~~ buffer

return TRUE

}

```
Buscar_Medicamento (Codigo, this) {
si (Nodo_tmp == first)
  Mientras (tmp) {
    si (tmp -> blet1) {
      si (codigo == codigo de medicamento en blet1) {
        Devuelve medicamento
      }
    }
    si (tmp -> blet2) {
      si (codigo == codig. del medicamento en blet2) {
        Devuelve medicamento
      }
    }
    tmp = tmp -> next
  }
  Devuelve NULL
}
```

Intafas grafica

Bienvenido al Inventario

Cargando - Extrayendo datos... (Estar atas)

Aviso - medicamentos caducados

Almacenados en Alerta.csv

¿Va a remover los medicamentos?

S.

Removiendo...

S. (Medicamentos eliminados del inventario

INVENTARIO

1- Imprimir Inventario

2- Nuevo Lotes

3- Nuevo producto al mercado

4- Eliminar producto del mercado

5- ? Ventas

>: -

1- Imprimir el inventario

2- Nuevos lotes

- ¿Nombre del medicamento?

L

- ¿Cuantos productos contiene?

L

Anexar codigo de barras

L

- Fecha de caducidad - (con numeros)

Dia : L

Mes : L

Año : L

Agregando al Inventario

Listo agregado

Void SetCantidad(Lote *this, int cantidad)
this->cantidad = cantidad;

GetCantidad(Lote *this)
return this->cantidad

Lote2 → Get;

2= Destroy Lote 1
1= apuntador Lote 1 = Lote 2
3= Lote 2 = NULL

Cantidad

Tmp2 = this → medicamento;
tmpcant = GetCantidad(tmp2 → lote1)

Si (this, nombre) != NULL
Si (cantidad <= tmpcant)

cantidad2 = tmpCant - cantidad

VERIFICAR si todavia hay medicamento
en existir?

REMOVER Medicamento(CLAVE*, char nombre)
*tmp = bst_search

Buscar si existe el medicamento
if (Imp == TRUE)
    ↳ Buscar si lote 1 y lote 2 tienen medica.
    if (lote 1 y lote 2 == existe Medicamento)
    {
    — CONTADOR REMOVER 2 = REMOVER.
    Si (REMOVER == TRUE)
    {
        REMOVEAT_(Imp)
        RETURN TRUE;
    }
    ↳ RETURN FALSE;

→ RETURN FALSE;

→ Sino: REMOVEAT (Imp)
    RETURN TRUE;
}

CONT 1 & CONT 2

toker

int dia = atoi(strtok("s", ,/)

1 solo medicamento

(DAL this)

1) ~~Caducidad (Medicamento this)~~ → Int (FILE * Alerta)

~~{~~ if (! Alerta) {

~~Si (Comparar_Fechas (this → lote 1)~~ "Error al abrir el archivo"

int dias = comparar_fechas (this → lote 1) return FALSE;

}

~~Si (dias < 16) { Imprimir ("Alerta de caducidad");~~

~~Insertar Formato (this → nombre, dias,~~ this → lote 1 → codigo, Alerta)

~~}~~

si (~~dias > 5~~) { this → lote 1 → bloqueo = TRUE }

return dias

}

2) Toda la lista

Caducidad (DAL * this)

{

Si (!(fpt_ALE = Abrir Archivo ("Inv_Alerta.csv", "w"))) {Devolver FALSE}

Tipo_tiempo ta = tiempo_local ()

it := this → first;

Mientras (it) {

Comparar_Fechas (it → medicamento, ta, dias 1, dias 2)

Si (dias 1 < 16)

Imp_Forato (fpt_Ale, "it → medicamento, it → medicamento → lote1, dias 1)

Si (dias1 < 6) {

Lote_setBloqueo(it → medicamento → lote1, TRUE) }

Alert = TRUE; }

Si (dias2 < 16)

Imp_Formato (fpt_Ale, "it → medicamento, it → medicamento → lote1, dias2)

Si (dias2 < 6)

Lote_setBloqueo (it → medicamento → lote2, TRUE) }

}

it = it → next;

}

fclose (fpt_Ale);

return alerta

}

struct Fecha_Type {

int dias;

int mes;

int año;

} Fecha

Fecha guardada cod: dd/mm/año

Tipotiempo Fecha(char *cad_fecha) {

fm Fecha; //estructura de tiempo

strcpy (buffer[10], cad-fecha);

dias = Subcadena (cad_Fecha, "/");

mes = Subcadena (cad-Fecha, "/");

año = Subcadena (cad_Fecha, "/");

Fecha.dias = Entero (dias)

Fecha_mes = Entero (mes) - 1 // Enero=0 diciembre=11

Fecha_Year = Entero (año) - 160 // de desde 1900

Devuelve Convertir Est_a_tim_t (tm)

}

Remover medicamento (DLL *this, char nom_medica)

//if(this->medicamento->nombre == nom_medica) →

//Entro al lote

If(this->medic->lote1->cantidad != 0)

lote1->cantidad -= 1

if(this->medic->lote1->cantidad == 0)

Remove lote

¿Cambiar lote?

Return TRUE;

else

//Remover lote

cambiar lote1 → x el 2

Return TRUE?

else

Return FALSE

Nodo → Medicamento → Nombre
Precio
cantidad
lote

Cantidad
caducidad
código
precio

Medicamento = Color1 (this*)

this → lote1;

If(tmp → medicamento → lote → cantidad

Medicamento *tmp = Medicamento - create()

tmp → lote1 → cantidad = 0;

Insertar medicamento (this, medicamento) → bool

{

si (!this) { devuelve false }

insertar lista (this, medicamento)

devuelve false;

}

~~bool remove (this, medicamento) → bool~~

{

si (first != last) {

Asigna el valor de first a medicamento

Nodo tmp = first → next

borrar Node first

}

else if (first) {

asigna el valor de last a medicamento

first = last = NULL

empty = true

Devuelve true;

}

else { devuelve false }

}

Comparar Fechas (Medicamento *this, time_t fa, int *dias, int *dias2){

// para dias 1

time_t f_cad = Fecha (Lote_GetCaducidad (this->lote1));

*dias1 = (int)(difftime (fcad, fa) / 86400);

difftime → (los regreso a segundos)

Segundos por día

f_cad = Fecha (Lote_GetCaducidad (this->lote2));

*dias2 = (int)(difftime (fa, fcad) / (86400));

}

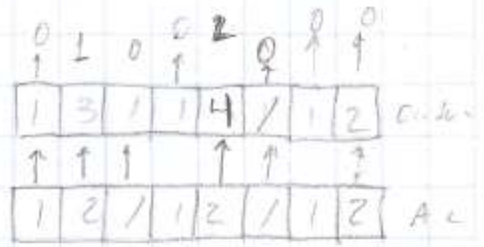
```
Archivo: Medicamento.txt (medicamento, buffer) => Medicamento
    Crear Medicamento();
    it = 0
    Lee (todos los caracteres de la cadena){
      Si (caracter = ';'){
          it = it + 1
      }
      Si (caracter = '\n'){
          break
      }
    }

    Extraer nombre
    Extraer precio
    Extraer control
    Si (it es mayor a 3){
    Crear lote()
      Extrae campos de lote 1
        Si (it es mayor a 6){
          Extrae campos de lote 2
        }
    }

  Regresa medicamento
}
```

LLP
DLL

```
Extraccion(DLL* this){                    -> bool
  si (!this) {return FALSE; }
      FILE  *fpt_Inv = NULL;
  si (!(fpt_Inv=(fopen ("inventorio.csv", "r")))) {
        (error al abrir el archivo \n");
        return FALSE;
    }
   Medicamento
  Producto  tmp = Medicamento_Create ();
   char buffer [1024];
   Fgets (buffer, 1024, fpt_Inv);
 mientras (! feof (fpt_Inv)){    destino        fuente
     Medicamento_Set Nombre (tmp, strtok (buffer, ",")));
     Medicamento_Set Precio (tmp, (float)(atof (strtok (NULL, ","))));
    si (! strcmp (strtok (NULL, ","), "Controlado")) {
        Medicamento_set Control (tmp, TRUE);
    }
   Lote_Set Cantidad (tmp->lote1, atoi (strtok (NULL, ",")));
   Lote_Set Caducidad (tmp->lote1, strtok (NULL, ","));
   Lote_Set Codigo (tmp->lote1, strtok (NULL, ","));
   Lote_Set Cantidad
   Lote_Set Caducidad
   Lote_Set Codigo
   si (!DLL_Insert (this, tmp)) {
      return FALSE;
   }
 fgets (buffer, 1024, fpt_Inv);
 }
Medicamento_Destroy (tmp);
 Fclose (fpt_Inv);
 return TRUE;
}
```

27 / 02   S

05 / 03   C



1	2	1	2	1	2

chus_rud= 8

$$\begin{array}{r} 30 \\ +\ 12 \\ \hline 42 \end{array} - 13$$

```
Imprimir_Inventario (this){
  Nodo *tmp = this->first;
  Mientras (tmp){
    Imprimir medicamento
    Imprimir lote1 y lote2.
    tmp = tmp->next
  }
```

```
Bool Modificar ( DLL *this) {
    si ( ! this ) { Devuelve FALSE ; }
    si ( ! fpt_new ) { Devuelve FALSE ; }
          cadena control
    Node *it = this -> first;
    while (it) {                      formato
    Imprimiendo ("Extrayendo Datos" );
     it = it -> next;
    }
    ~~cerrar (fpt_inv)~~
    cerrar (fpt_new)
    Borrar ("inventario.csv")
    Renombrar (fpt_new)
```

→ si (medicamento controlado) {
control == Controlado }

si no (control ==
No_Controlado) {

Eliminar Medicamento -

Eliminar medicamento
↳ ¿Que medicamento desea eliminar?
Buscar if (medicamento existe)
↳ if (Existe algo en lote1 y lote2)
↳ return true;
↳ else (si no existe)
↳ return false;
return false;

DRIVER PROGRAM

Que medicamento desea remover
if (confir-remove == true)
↳ Existen productos en los lotes del medi
¿Seguro que desea removerlo? SI/NO
↳ if (SI)

10/10/2018

No ⟶ Enter para continuar

```
Remover_MedicamentoUnidosis(Medicamento* this, int cant) Bool
{
    si (!existe this) { devuelve FALSE }
    si (existe this -> lote1) {                    si (cant == 0) { devuelve TRUE }
        this -> lote1 -> cantidad := this -> lote1 -> cantidad - 1
        si (this -> lote1 -> cantidad == 0) {                cant - 1
            si (existe this -> lote2) {
                Elimina this -> lote1
                this -> lote1 = this -> lote2
                this -> lote2 = NULL
            }
        else { this -> lote1 = NULL }
        }
        return TRUE Remover_MedicamentoUnidosis(this, cant)
    }
    devuelve FALSE
}
```