

**PY01**

**Gestión de Librería**

David Calvo, Erin Camacho, Johnsy López

Instituto Tecnológico de Costa Rica

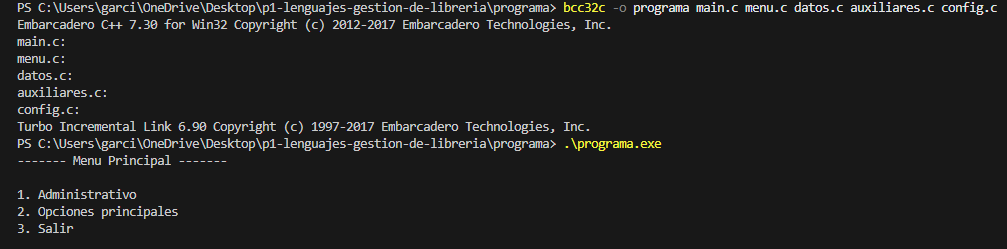
IC4700 Lenguajes de Programación

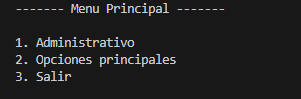
Prof. Allan Rodríguez Dávila

II Semestre

16 de septiembre del 2025

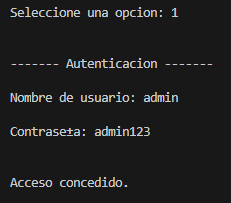
**Manual de Usuario**

1. Instalar Compilador Embarcadero
   1. Ingresar a <https://www.embarcadero.com/es/free-tools/ccompiler/free-download/thank-you> y descargar él .zip.
   2. En la ubicación del archivo, clic derecho sobre el archivo .zip descargado y selecciona “Extraer aquí”.
   3. Presiona Win + R, escribe sysdm.cpl, ve a Opciones Avanzadas y Variables de entorno.
   4. En la sección Variables del Sistema, edita la variable Path.
   5. Agrega la ruta a la carpeta bin del compilador previamente extraído.
   6. Acepta todos los cambios.
2. Clonar repositorio github
   1. Desde el Simbolo del sistema (cmd), usar el comando “git clone [https://github.com/johnsydev/p1-lenguajes-gestion-de-libreria#](https://github.com/johnsydev/p1-lenguajes-gestion-de-libreria)”
   2. Escribe cd programa
   3. Escribe este comando: bcc32c -o programa main.c menu.c datos.c auxiliares.c config.c y luego .\programa.exe
   4. Se abrirá el menú principal del sistema en la consola.
3. Menu Principal y Uso General
   1. Al iniciar, el sistema cargará automáticamente los datos desde los archivos de texto (si existen).
   2. El menú principal muestra las opciones **Administrativas** y **Generales**.
   3. Para acceder a las opciones administrativas se debe ingresar el **usuario y contraseña** almacenados en el archivo correspondiente (config.json).
   4. El sistema guardará los cambios realizados (libros, clientes, pedidos) en los archivos de texto al salir, asegurando la persistencia de la información.
   5. Para mayor explicación de funcionalidades, observa el siguiente apartado.

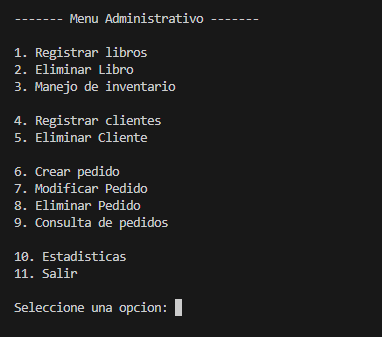
**Pruebas de Funcionalidad**

Al iniciar el programa se muestra este menú, con opción Administrativo, Opciones Generales y Salir.

1. **Administrativo**

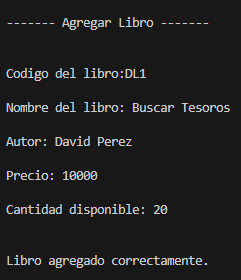


Para ingresar a esta opción, se solicitará usuario y contraseña de la libreria, en este caso definidos como admin y admin123 respectivamente.

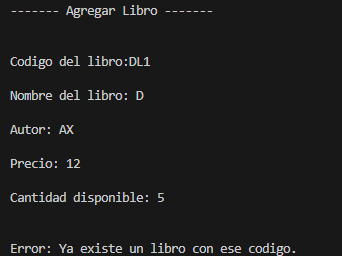


Al tener acceso concedido se mostrará este menú con esta variedad de opciones.

1. Registrar Libro

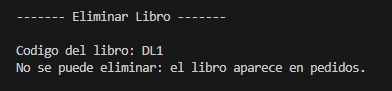


Al agregar un libro se solicitarán los siguientes datos: Código de libro único, titulo y autor como caracteres, precio flotante y cantidad entero, ambos positivos,

****

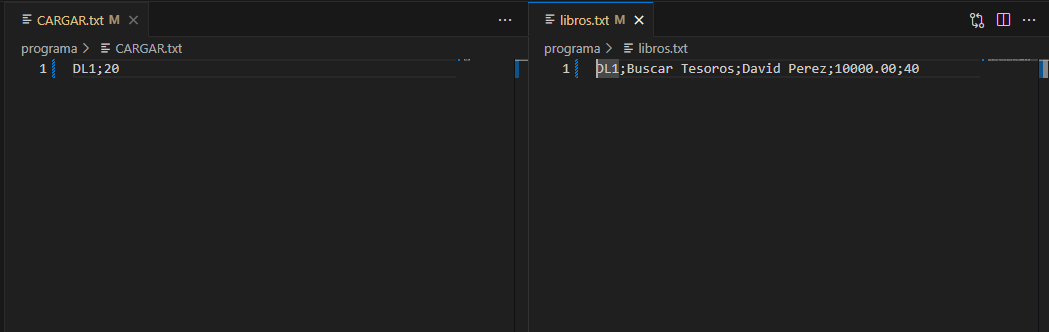
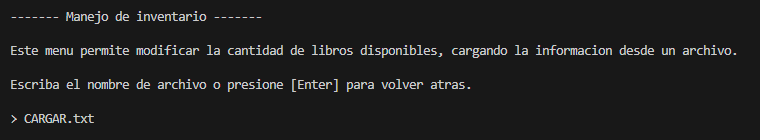
Si se intenta colocar otro libro con el mismo código, lanza este error.

1. Eliminar Libro

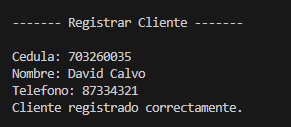


Permite eliminar un libro siempre y cuando este no se encuentre asociado a un pedido.

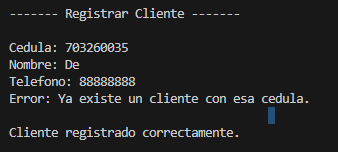
1. Manejo de Inventario

****Este apartado permite dar mantenimiento al stock de libros por medio de una carga de lote, utilizando el archivo CARGAR.txt. Suma o resta el stock existente,

1. Registrar clientes

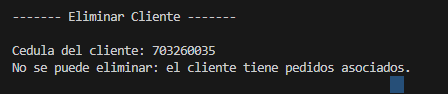


Al registrar un cliente se solicitará su cedula única, y nombre y teléfono como cadena de caracteres.

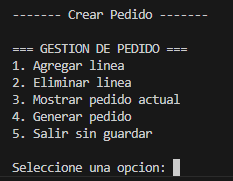
****

Si se intenta registrar otro cliente con la misma cedula lanza este error.

1. Eliminar Cliente

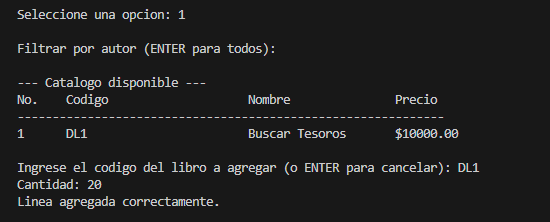
Permite eliminar un cliente registrado, siempre y cuando no esté asociado a un pedido.

1. Crear pedido



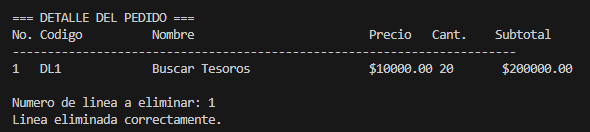
Para la creación del pedido se muestra el siguiente menú con dichas opciones como si fuera un bucle. Es el mismo pedido hasta que se genere o se salga sin guardar.

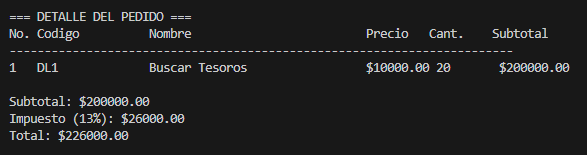
6.1 Agregar Línea



Básicamente se muestra el catálogo de libros, se escoge el libro y la cantidad comprada para el respectivo pedido,

6.2 Eliminar línea

Se escoge el número de línea y se elimina dicha línea agregada del pedido.

 6.3 Mostrar pedido actual

Se muestra el detalle del pedido, en el caso de no haber líneas se muestra el mensaje “No hay líneas en el pedido”.

 6.4 Generar pedido

Se solicita la cedula (existente) del cliente, y la fecha del pedido. Posteriomente, se muestra el detalle del pedido completo.

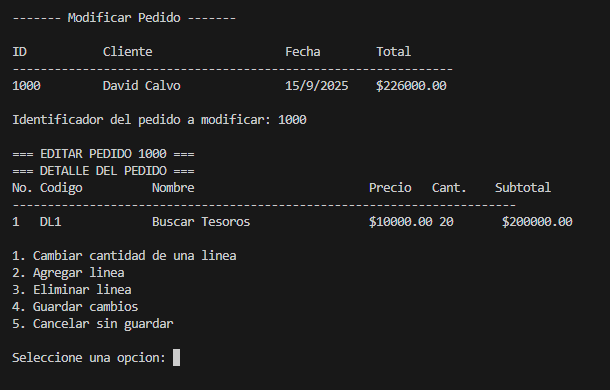
Si el pedido no tiene líneas, no es posible generar el pedido,

Se actualiza el stock del libro.

6.5 Salir sin guardar

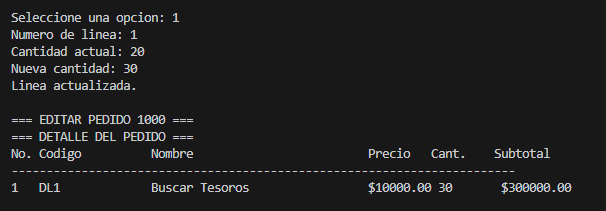
Básicamente, no se genera ni guarda ninguna línea del pedido.

1. Modificar Pedido

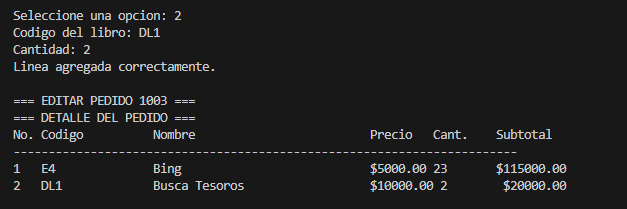


Para la modificacion del pedido se muestra los pedidos existentes, se solicita que se escoja uno y luego un menú con dichas opciones como si fuera un bucle. Es el mismo pedido hasta que se guarden los cambios o se cancele sin guardar.

7.1 Cambiar cantidad de una línea

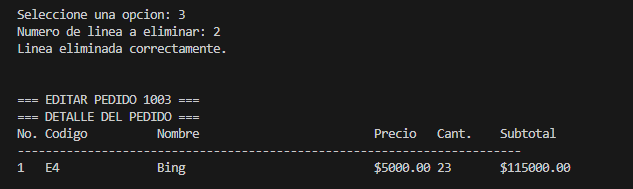


Se muestra la cantidad actual y se solicita una nueva cantidad. Esta nueva cantidad se suma o resta la diferencia con la actual, en el stock del libro. Si el pedido tenía 20 y se actualiza a 30, el stock del libro se resta 10 y etc.

**** 7.2 Agregar línea

Se agrega una nueva línea al pedido, lo mismo que con crear pedido.

7.3 Eliminar Línea



Eliminar línea de un pedido básicamente elimina esa línea del pedido.

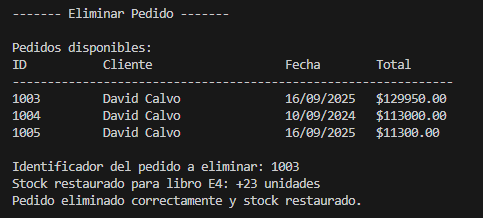
7.4 Guardar Cambios

Pedido actualizado correctamente.

7.5 Cancelar sin guardar

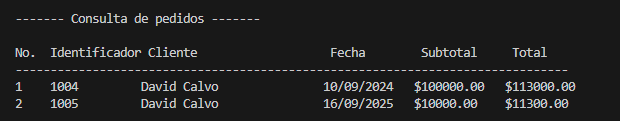
No se efectúa ningún cambio.

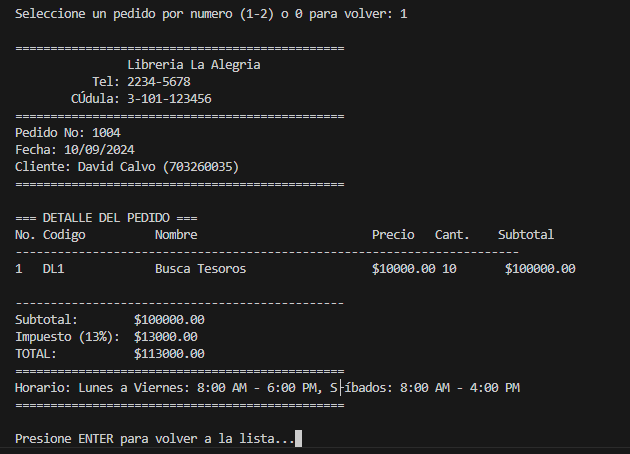
1. Eliminar Pedido



Se elimina un pedido creado, y se restaura el respectivo stock del libro (aumentar cantidad).

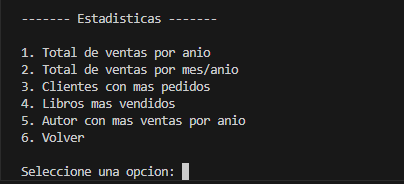
1. Consulta de pedidos

Se realiza la consulta de pedidos existentes. 



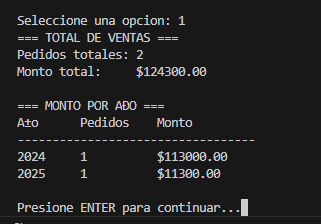
Si se desea, se puede seleccionar un pedido por número de línea para conocer el detalle completo del pedido.

1. Estadisticas



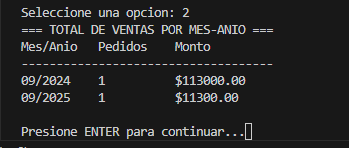
El menú de estadísticas tiene esta variedad de opciones.

10.1 Total de Ventas por Año

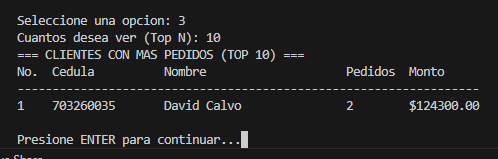


Se muestra el total generado por cada pedido por año.

10.2 Total de ventas por mes/año

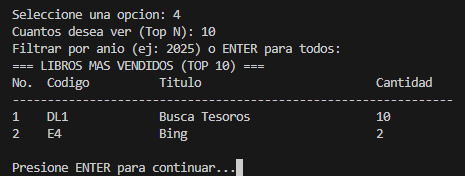


Se muestra el total agrupado por mes/año.

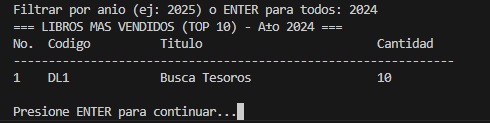
10.3 Clientes con más pedidos****

Se muestran los clientes con más pedidos, mostrando su cedula, nombre, pedidos y monto.

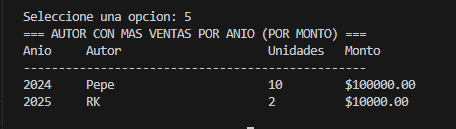
10.4 Libros más vendidos



Se muestran los libros más vendidos, mostrando el código, titulo y cantidad vendida.

 Existe el filtro opcional que permite filtrar esos libros por año del pedido.

10.5 Autor con más ventas por año

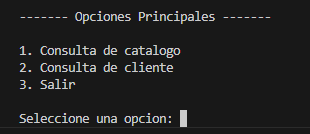


Se muestran los autores con mas ventas por alo, mostrando el año, autor, unidades y monto.

1. Salir

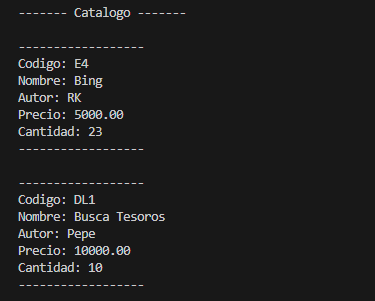
Regresa al menú principal.

1. **Opciones Generales**



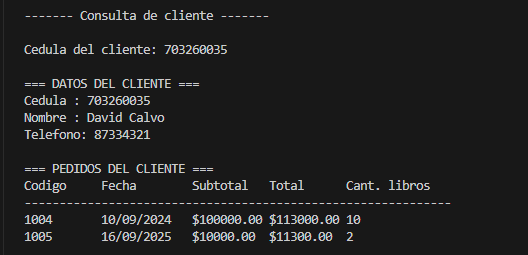
Este apartado es de acceso público, para cualquier usuario, no requiere de ningún acceso con usuario/contraseña. Contiene la consulta de catálogo, consulta de cliente y Salir.

1. Consulta de Catalogo



Se muestra el catálogo actual de la librería, los libros en stock.

1. Consulta de cliente



En este menú, se solicita la cedula del cliente y posteriormente se muestran los datos del cliente como su cédula, nombre y teléfono, también sus pedidos.

1. Salir

Regresa al menú principal.

1. **Salir**

El programa finaliza.

**Descripción del Problema**

El proyecto consiste en el desarrollo de un sistema de gestión de inventario y pedidos para una librería, implementado en el lenguaje de programación C. El propósito principal es simular una situación real donde un negocio necesita administrar de forma organizada y eficiente la información de los productos que ofrece (libros), los clientes que compran y los pedidos que se realizan diariamente.

El problema que se busca resolver es que, si este proceso se lleva de manera manual, es muy fácil cometer errores: por ejemplo, vender un libro sin tener existencias, registrar dos veces al mismo cliente, o no tener un historial claro de los pedidos que ya se generaron. Además, sin un sistema que persista la información, es difícil consultar rápidamente datos importantes como: cuáles son los libros más vendidos, cuáles clientes compran con más frecuencia o cuánto ha vendido la librería en un periodo de tiempo.

Por esa razón, el sistema debe incluir varias funcionalidades: registrar y actualizar libros en inventario, registrar clientes, crear y consultar pedidos, así como generar estadísticas de ventas. Todo esto se debe manejar con archivos de texto para asegurar la persistencia de los datos, de manera que al volver a ejecutar el programa la información registrada siga estando disponible.

A nivel técnico, este proyecto permite aplicar de forma práctica muchos de los conceptos vistos en el curso, como el uso de punteros, memoria dinámica, estructuras (struct), modularidad, paso de parámetros y manejo de archivos. También se exige implementar un sistema de menús en consola, con opciones tanto para el administrador (con acceso mediante usuario y contraseña) como para consultas generales accesibles a cualquier usuario.

En resumen, el problema que aborda este proyecto es cómo diseñar e implementar, desde cero y en lenguaje C, un sistema que organice y administre de manera adecuada la información de una librería, evitando errores, mejorando la consulta de datos y permitiendo obtener estadísticas útiles para la gestión del negocio.

**Diseño del Programa**

1. **Decisiones de diseño**

* Persistencia simple en archivos de texto planos (.txt / .json).
  + La configuración se lee/escribe con [cargarConfiguracion](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") y [guardarConfiguracion](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") desde/ hacia [config.json](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html).
* Estructuras de dominio en headers:
  + Libros: [struct Libro](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") en [libro.h](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html).
  + Clientes: [struct Cliente](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") en [cliente.h](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ").
  + Pedidos y detalles: [struct Pedido](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") y DetallePedido en [pedido.h](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ").
  + Registros para estadísticas: [Registro](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) en [registro.h](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ").
* Separación en capas/archivos:
* Interfaz / navegación: [menu.c](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") + [menu.h](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") (entrada y menús).
* Lógica de datos y persistencia: [datos.c](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") + [datos.h](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html).
* Utilidades de strings/memoria: [auxiliares.c](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") + [auxiliares.h](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ").
* Configuración: [config.c](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") + [config.h](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ").
* Gestión de memoria:
  + Uso explícito de malloc/realloc/free. Funciones auxiliares para clonar cadenas: [asignarString](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o "). Comparación propia: [compararString](vscode-file://vscode-app/c:/Users/garci/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html).

1. **Algoritmos usados**

* Lectura de archivos y parseo:
  + Lectura línea por línea en leerArchivo y separación por delimitador en separarTexto. Operan en tiempo lineal respecto a las líneas por archivo.
* Búsquedas:
  + Búsquedas por código/cedula se hacen por recorrido lineal (p. ej. buscarLibroPorCodigo, buscarClientePorCedula),
* Actualizaciones de arrays dinámicos:
  + Inserciones/elim. usan realloc y desplazamiento.
* Ordenamiento:
  + Utilizandose para clientes con más pedidos, ordenarPorCantidadDesc en [programa/datos.c] utiliza un bubble-sort.
* Estadísticas / agregaciones:
  + Construcción de agrupaciones (por año, por cliente, por libro, por autor) se realiza acumulando en arreglos dinámicos y buscando entradas existentes con búsqueda lineal.
* Persistencia:
  + Reescritura completa de archivos para operaciones como actualizar inventario o reescribir pedidos (Funciones: actualizarTodosLibros, actualizarTodosClientes, reescribirArchivosPedidos).

1. **Diagrama de Archivos**

* programa/main.c — entrada y arranque del menú.
* programa/menu.c / menu.h — interacción con el usuario y flujos.
* programa/datos.c / datos.h — carga, búsqueda, actualización de libros, clientes y pedidos; generación de estadísticas.
* programa/config.c / config.h — carga/guardado de config JSON y gestión de IDs.
* programa/auxiliares.c / auxiliares.h — utilidades para strings y memoria.
* Headers de tipos:
  + programa/libro.h (struct Libro)
  + programa/cliente.h (struct Cliente)
  + programa/pedido.h (struct Pedido y DetallePedido)
  + programa/registro.h (tipos para estadísticas)
* Archivos de datos:
  + programa/libros.txt, clientes.txt, pedidos.txt, detalles.txt, CARGAR.txt, config.json

**Librerías Usadas**

* **stdio.h** — manejo de archivos y entrada/salida (fopen, fclose, fprintf, fscanf, fgets, printf). Usado en: config.c, datos.c, menu.c, main.c.
* **stdlib.h** — gestión de memoria dinámica, conversiones y utilidades (malloc, realloc, free, atoi, exit). Usado en: auxiliares.c, datos.c, config.c, menu.c.
* **string.h** — operaciones con cadenas (strcpy, strcmp, strlen, strtok). Usado en: auxiliares.c, datos.c, config.c, menu.c.
* **stdbool.h** — tipo booleano (bool, true, false). Usado en: archivos de lógica (datos.c, menu.c).
* **time.h** (si hay manejo de fechas/tiempos) — formato/validación de fechas o timestamps (presente solo si se usa en registros/estadísticas).
* **Headers locales:** auxiliares.h, datos.h, menu.h, config.h, libro.h, cliente.h, pedido.h, registro.h — definen tipos y prototipos usados por los .c.

**Análisis de resultados**

| **Objetivo** | **Porcentaje de realización** | **Justificación** |
| --- | --- | --- |
| Registrar libros en inventario | 100% | Se implementó correctamente el registro de libros con validación de código único y almacenamiento en estructuras dinámicas. |
| Actualizar inventario (carga en lote) | 100% | Se desarrolló el módulo para modificar cantidades desde archivo externo, validando que no existan valores negativos. |
| Registrar clientes | 100% | Se implementó el registro con validación de cédula única y validaciones de formato en los datos ingresados. |
| Crear pedidos | 100% | El sistema permite agregar, eliminar y modificar líneas de pedido, validando el stock disponible y descontando al confirmar. |
| Consultar pedidos | 100% | Se listan todos los pedidos generados y es posible ver el detalle completo de cada uno. |
| Generar estadísticas (ventas, clientes frecuentes, libros más vendidos) | 100% | Se implementaron todas las estadísticas requeridas, incluyendo filtros por año. |
| Consulta de catálogo | 100% | Se despliega correctamente la lista de libros, con opción de filtrar por autor. |
| Consulta de cliente | 100% | Se muestran los datos del cliente y su historial de pedidos con totales. |
| Seguridad de acceso (usuario y contraseña administrador) | 100% | El sistema restringe correctamente el acceso a las funciones administrativas mediante validación de credenciales. |
| Persistencia de datos en archivos de texto | 100% | Toda la información se guarda y carga desde archivos de texto. |
| Modularidad y uso de punteros, memoria dinámica, paso de parámetros | 100% | El programa fue diseñado de forma modular, usando struct, punteros y memoria dinámica con control de malloc y free. |
| Documentación interna y externa | 100% | Se incluyó documentación en el código y en el externo según lo solicitado. |

**Justificación de toma de decisiones**

1. **Punteros en secciones de código. Ejemplos concretos de paso por valor y por referencia**

**Punteros:**

* Strings en C: las cadenas son char\*; usar punteros es obligatorio para manejar nombres, códigos y textos.
* Datos dinámicos y vida útil: objetos creados en heap (libros, clientes, pedidos, configuración) deben sobrevivir al retorno de funciones, por eso se devuelven punteros; el llamador es responsable de liberar.
* Arrays dinámicos (Libro\* libros, Cliente\* clientes, Pedido\* pedidos) requieren punteros para realloc, y compartir la misma región de memoria entre funciones.
* Eficiencia / evitar copias: structs pueden contener varios campos (cadenas, arrays). Pasar punteros evita copiar estructuras grandes y reduce coste.
* Funciones que modifican objetos (stock, secuencial, detalles del pedido) reciben punteros para cambiar el estado sin devolver copias.
* Relaciones y referencias: un Pedido puede referenciar Cliente o DetallePedido por puntero o por índice; punteros permiten modelar relaciones directas sin duplicar datos.
* Punteros permiten definir quién posee la memoria

**Paso por valor y por referencia:**

* Buscar y devolver una referencia en memoria:
  + Las funciones de búsqueda (por ejemplo, buscar un libro por código) devuelven un puntero al elemento dentro del array cuando lo encuentran. Esto evita copiar el libro y permite que se lea o modifique ese libro directamente. Si no existe, devuelven NULL para indicar ausencia.
* Modificación por referencia:
  + Para operaciones que cambian estado (por ejemplo, reducir o aumentar stock de un libro, agregar un detalle a un pedido) las funciones reciben punteros a las estructuras para aplicar el cambio sobre el objeto original en memoria sin tener que devolver una copia modificada.
* Gestión de listas dinámicas:
  + Al insertar un nuevo cliente o pedido se utiliza la combinación de un puntero al array (y en algunos casos puntero a puntero) junto con el tamaño, para llamar a realloc y ampliar la colección; la función modifica el array compartido y actualiza el contador, por eso opera sobre referencias y no sobre copias locales.
* Creación y retorno de estructuras al heap:
  + Las funciones que crean entidades (por ejemplo, crear un pedido nuevo) asignan la estructura en el heap y retornan un puntero. Esto garantiza que el objeto siga existiendo después de que la función retorne y define claramente quién debe liberar esa memoria.
* Paso por valor cuando no se modifica:
  + En operaciones meramente de mostrar y con estructuras pequeñas (por ejemplo, una función que imprime datos de un cliente sin modificarlos), puede usarse paso por valor para indicar que la función no altera el original y que una copia temporal es suficiente.

1. **¿Cómo se asegura la modularidad y el ocultamiento de la información?**

* Separación por responsabilidad (cada .c tiene una capa):
  + menu.c: interfaz y flujo del usuario.
  + datos.c: lógica de dominio y persistencia.
  + config.c: carga/guardado de configuración.
  + auxiliares.c: utilidades comunes.
* Encapsulación mediante headers y funciones públicas
  + Solo los prototipos necesarios se exponen en los \*.h (datos.h, menu.h, config.h, libro.h, cliente.h, pedido.h). El resto son funciones internas.
* Ocultamiento con funciones estáticas
  + Funciones internas que no deben exportarse (p. ej. extraerValorJSON en config.c) se declaran static para restringir su visibilidad.
* Minimizar acoplamiento
  + Tipos compartidos (structs en libro.h / cliente.h / pedido.h) permiten comunicación controlada; la lógica de persistencia está en datos.c, no dispersa por la UI.

**Bitácora**

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Usuario** | **Comentario** |
| 2025-08-30 | johnsydev | Inicio de carpetas |
| 2025-08-30 | johnsydev | Inicio de menús |
| 2025-08-30 | Johnsydev | Lectura de archivos, separador de texto, autenticación de administradores |
| 2025-08-31 | Johnsydev | Lectura de archivos, separador de texto, autenticación de administradores |
| 2025-09-01 | Johnsydev | Registrar libro |
| 2025-09-01 | Johnsydev | Modificar inventario y stock, auxiliares |
| 2025-09-01 | Dcalvog25 | Registrar Cliente  Funcionalidad de opciones administrativas: Registrar Clientes. |
| 2025-09-04 | Dcalvog25 | Creacion de Pedidos  Se realizo la funcionalidad de crear pedidos, con el menú de gestión del mismo. Además, de otros cambios y mejoras de errores y warnings. |
| 2025-09-04 | Dcalvog25 | Separacion Pedidos y Detalles  Se realizo la separacion en dos archivos pedidos (encabezado) y detalles del pedido. |
| 2025-09-07 | Erincam09 | Creacion de Estadisticas |
| 2025-09-07 | Erincam09 | Cambios en los prints |
| 2025-09-07 | Erincam09 | Consulta por Cliente y Consulta de Pedidos |
| 2025-09-07 | Erincam09 | Funcionalidades |
| 2025-09-13 | johnsydev | Memoria dinámica en manejo de strings |
| 2025-09-14 | Johnsydev | Avances |
| 2025-09-14 | Dcalvog25 | Valores iniciales en json y Eliminar Pedido |
| 2025-09-14 | Erincam09 | Avances Finales |
| 2025-09-15 | Dcalvog25 | Modificaciones en Generar Pedido |
| 2025-09-15 | Dcalvog25 | Mejoras en errores |
| 2025-09-16 | Erincam09 | Validacion de Stock |
| 2025-09-16 | Dcalvog25 | Mejoras en ModificarPedido |
| 2025-09-16 | Johnsydev | Validaciones |
| 2025-09-16 | Dcalvog25 | DocumentacionExt, Txt y Carpetas |