T02

Gestión de Frutería

David Calvo García

Instituto Tecnológico de Costa Rica

IC4700 Lenguajes de Programación

Prof. Allan Rodríguez Dávila

II Semestre

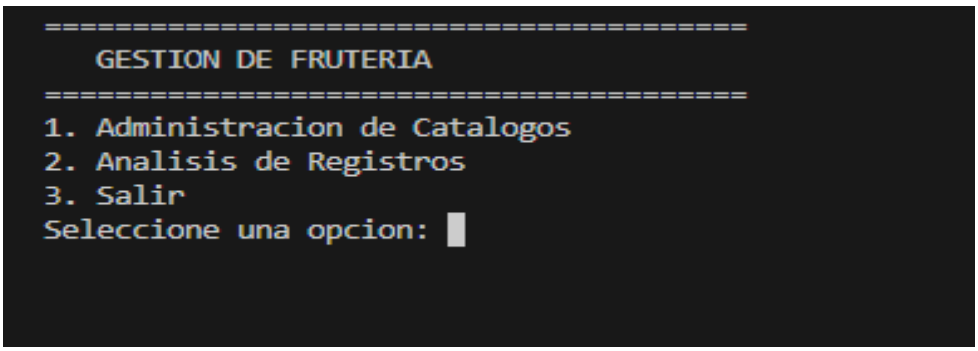
26 de septiembre del 2025

Manual de Usuario

1. Instalar SML of New Jersey
    a. Ingresar a https://www.smlnj.org/dist/working/110.99.8/index.html y descargar de su versión correspondiente, en este caso Windows versión 110.99.8.
    b. Presiona Win + R, escribe sysdm.cpl, ve a Opciones Avanzadas y Variables de entorno.
    c. En la sección Variables del Sistema, edita la variable Path.
    d. Agrega esta ruta C:\Program Files (x86)\SMLNJ\bin.
    e. Acepta todos los cambios.

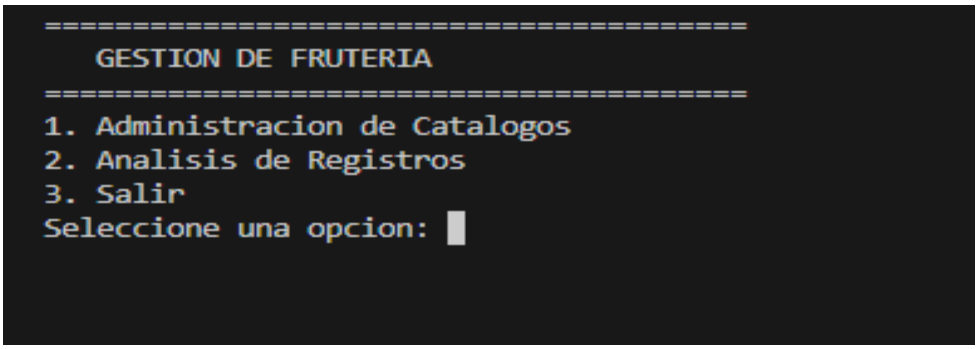
2. Clonar repositorio github
    a. Desde el Simbolo del sistema (cmd), usar el comando "git clone https://github.com/Dcalvog25/Tarea2-Gestion-de-Fruteria"
    b. Escribe cd programa
    c. Escribe este comando: sml main.sml
    d. Se abrirá el menú principal del sistema en la consola.

3. Menu Principal y Uso General}



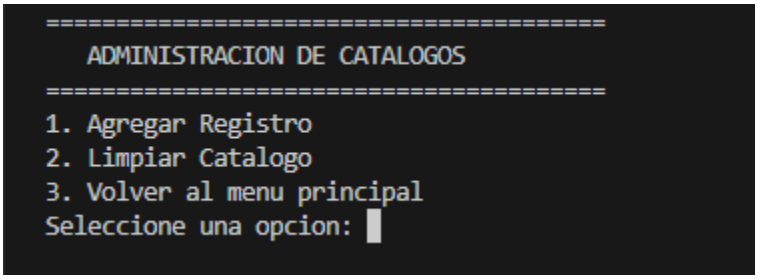
    a. El menú principal muestra las opciones **Administracion de Catalogos** y **Analisis de Registros**.
    b. Para mayor explicación de funcionalidades, observa el siguiente apartado.

Pruebas de Funcionalidad



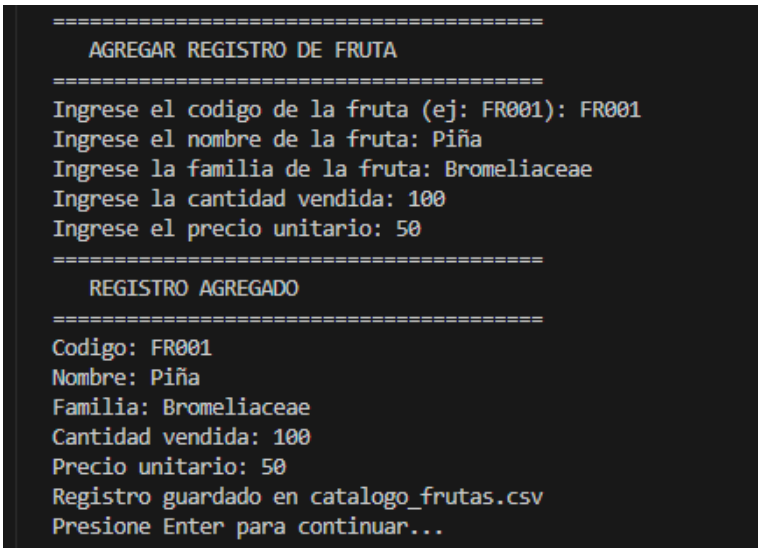
Al iniciar el programa se muestra este menú, con opción Administracion de Catalogos, Analisis de Registros y Salir.

A) Administración de Catálogos



El menú presenta las siguientes tres opciones: Agregar Registro, Limpiar Catálogo y Regresar.

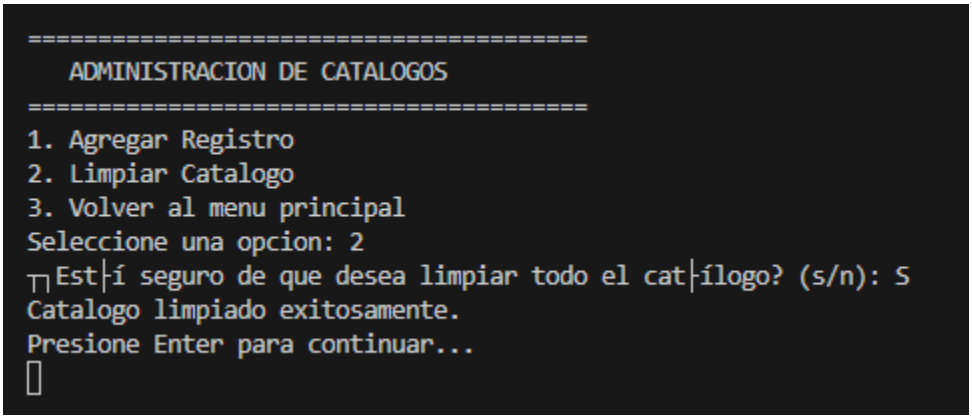
1) Agregar Registro



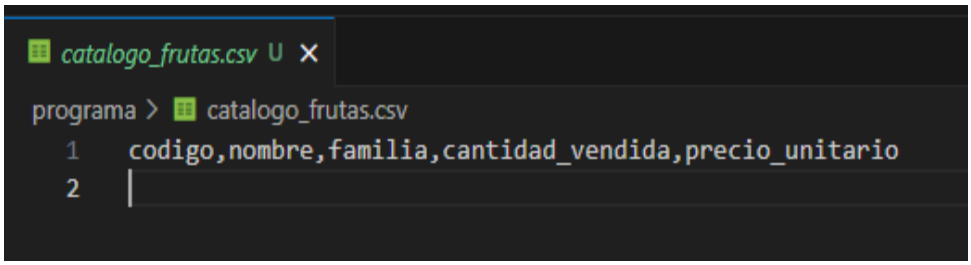
Al agregar un registro de frutas se solicitarán los siguientes datos: Código de fruta, nombre y familia como caracteres, cantidad vendida entero y precio unitario real, ambos positivos.

El archivo catalogo_frutas.csv se crea y ahí se guardan cada vez que se quiera agregar un nuevo registro.

2) Limpiar Catálogo

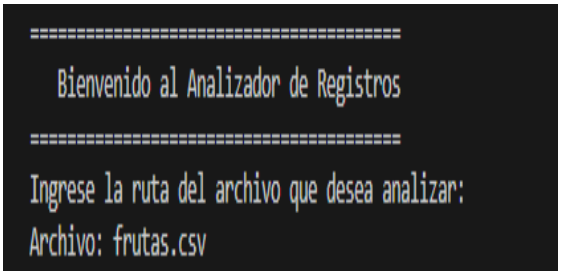


Limpia el catálogo de frutas registrados en el archivo .csv. Solo dejando el encabezado.



3) Volver al menú principal
Regresa al menú principal.

B) Análisis de Registros



Para ingresar a esta opción, primero se le solicita la ruta de un archivo .csv que será analizado, debe existir, si no, no se permite el ingreso al menú. Puede ingresar el creado en agregar registro (ver punto A) o cualquier otro que tenga.

```
Seleccione una opcion de analisis para el archivo frutas.csv:
1. Mostrar frutas populares dentro de un rango de cantidad vendida
2. Identificar familias con mas de 4 frutas diferentes registradas
3. Buscar frutas por codigo o nombre
4. Cantidad y lista de frutas por familia
5. Resumen general de la verduleria
6. Volver al menu principal
Opcion:
```

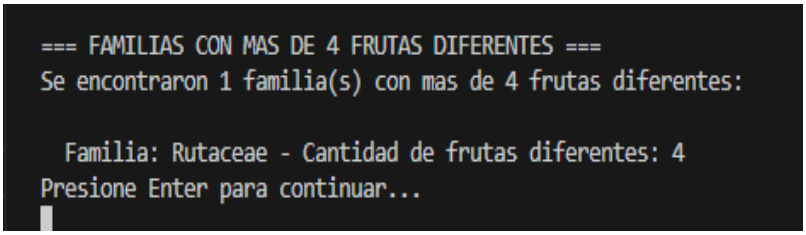
Al tener acceso concedido, se mostrará este menú con esta variedad de opciones:

1) Mostrar frutas populares dentro de un rango de cantidad vendida

```
========================================
   FRUTAS POPULARES DENTRO DE UN RANGO DE MONTO VENDIDO
========================================
Ingrese el monto minimo vendido: $50000
Ingrese el monto maximo vendido: $200000

=== RANKING DE FRUTAS POPULARES EN EL RANGO DE $5E4 A $2E5 ===
Se encontraron 4 tipos de fruta en el rango:

     Nombre: Mel├┤n
     Codigo: FR006
     Familia: Cucurbitaceae
     Total unidades vendidas: 321
     MONTO TOTAL VENDIDO: $176550
     PRECIO UNITARIO: $550
     --------------------------------------------------
     Nombre: Papaya
     Codigo: FR003
     Familia: Caricaceae
     Total unidades vendidas: 120
     MONTO TOTAL VENDIDO: $84000
     PRECIO UNITARIO: $700
     --------------------------------------------------
     Nombre: Pi▒a
     Codigo: FR001
     Familia: Bromeliaceae
     Total unidades vendidas: 150
     MONTO TOTAL VENDIDO: $75000
     PRECIO UNITARIO: $500
     --------------------------------------------------
     Nombre: Guayaba
     Codigo: FR010
     Familia: Myrtaceae
     Total unidades vendidas: 281
     MONTO TOTAL VENDIDO: $56200
     PRECIO UNITARIO: $200
     --------------------------------------------------

=== RESUMEN DEL RANGO ===
Total de tipos de fruta en el rango: 4
Total de unidades vendidas en el rango: 872
Total de ventas en el rango: $391750
Presione Enter para continuar...
```

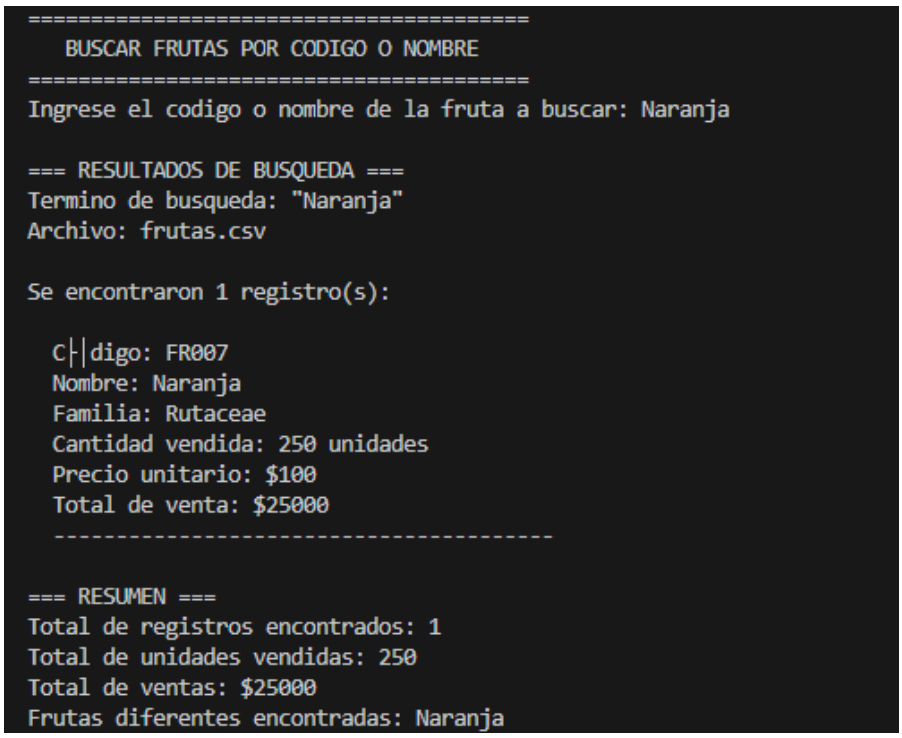
Se solicita al usuario que ingrese un monto mínimo y máximo vendido, el cual será el rango a buscar para el ranking de frutas. Por fruta se muestra el nombre, código, familia, total unidades vendidas, monto total vendido y el precio unitario.

2) Identificar familias con al menos 4 frutas diferentes registradas



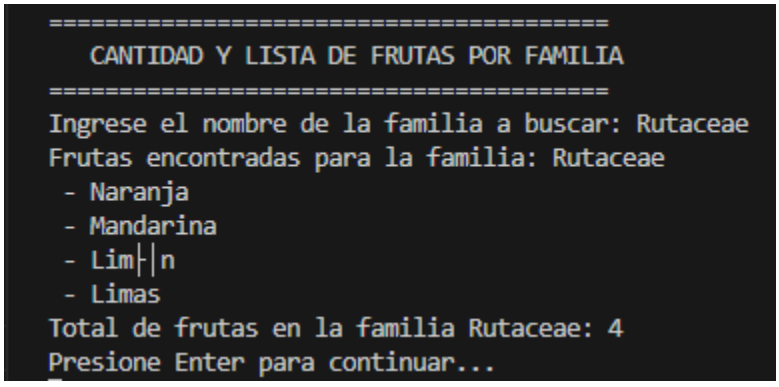
Se muestran las familias con al menos 4 frutas registradas en el catálogo.

3) Buscar frutas por código o nombre



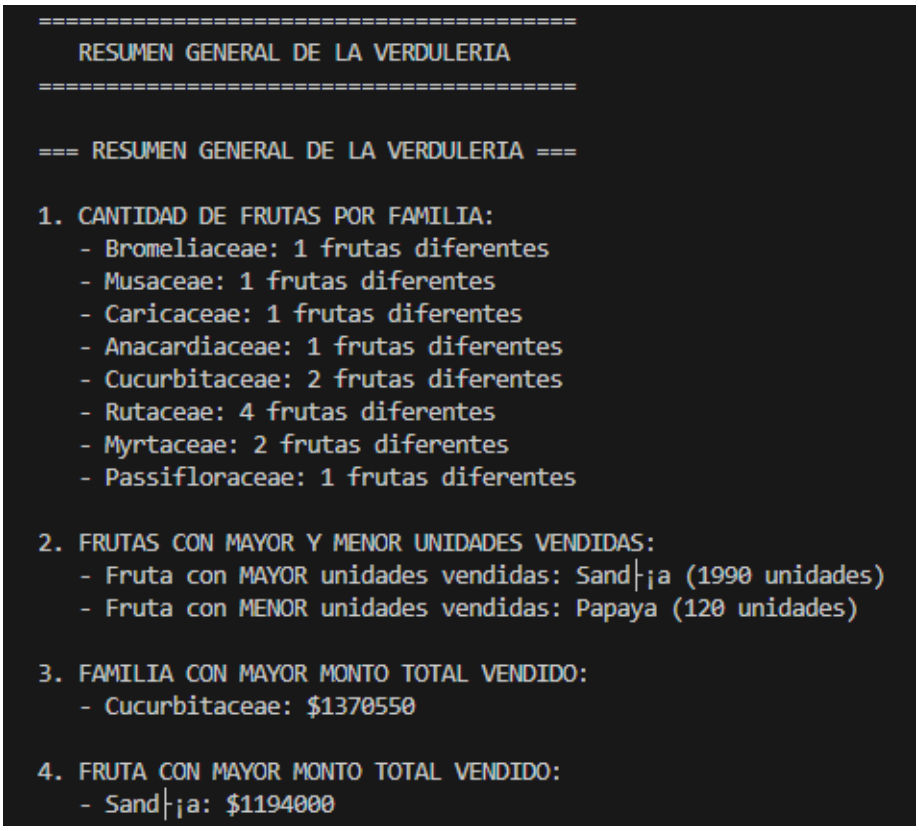
Se solicita al usuario código o nombre de una fruta y se muestra todas las ventas asociadas.

4) Cantidad y lista de frutas por familia



Se solicita al usuario que ingrese el nombre de una familia a buscar, y se muestra las frutas asociadas a ella.

5) Resumen general de la verdulería



Se muestra un resumen general de verdulería con cantidad de frutas por familia, frutas con mayor y menor unidades vendidas y familia y fruta con mayor monto total vendido,

6) Volver al menú principal

Regresa al menú principal

C) Salir

El programa finaliza.

Descripción del Problema

El problema consistía en implementar un sistema de gestión para una frutería utilizando el lenguaje Standard ML (SML), aplicando el paradigma funcional. El sistema debía estar dividido en dos partes principales:

1. Creador: encargado de administrar el catálogo de frutas, permitiendo agregar nuevos registros con código, nombre, familia, cantidad vendida y precio unitario, así como limpiar el catálogo para iniciar desde cero.
2. Analizador: encargado de procesar el archivo CSV con la información de frutas y ventas, mostrando distintos análisis como frutas más populares según ventas, familias con más de cuatro frutas registradas, búsqueda por código o nombre, cantidad de frutas por familia y un resumen general con métricas específicas (como fruta con más/menos ventas y familia con mayor monto vendido).

El reto principal fue diseñar y programar todo este sistema de manera funcional, utilizando listas como estructuras de datos, funciones puras y funciones de orden superior (map, filter, foldl).

Diseño del Programa

A. Decisiones de diseño

- Persistencia en archivos CSV (.csv):
  - Los datos de frutas se leen/escriben desde/hacia un archivo .csv.
  - Las funciones leerRegistrosCSV y agregarRegistro manejan la persistencia de datos usando archivos de texto plano con formato CSV.
- Estructuras de tuplas:
  - Registros de frutas: tupla (codigo, nombre, familia, cantidad, precio) manejada implícitamente en todas las funciones de análisis.
  - Se usan tuplas de SML para representar los datos.
- Separación en capas/archivos:
  - Interfaz / navegación: main.sml (menú principal) y funciones de menú en creador.sml y analizador.sml.
  - Lógica de datos: funciones de lectura/escritura CSV en auxiliares.sml.
  - Utilidades de entrada/validación: auxiliares.sml (validación de enteros/reales, entrada de usuario).
  - Gestión de archivos: funciones como archivoExiste y crearArchivoSiNoExiste en auxiliares.sml.
- Gestión de memoria:
  - Uso de estructuras de datos propias de SML (listas).
  - Manipulación de strings mediante funciones nativas como String.substring, String.explode y String.implode.
  - Conversiones de tipos con funciones como Int.fromString y Real.fromString con manejo de opciones (SOME/NONE).

B. Algoritmos usados

- Lectura de archivos y parseo:
  - Lectura línea por línea en leerRegistrosCSV y separación por delimitador coma en dividirCSV.
  - Conversión de líneas CSV a tuplas de registros con lineaARegistro.
- Búsquedas:
  - Búsquedas por código/nombre se realizan por recorrido lineal con List.filter (ej. en buscarFrutasPorCodigoONombre).
  - Búsqueda de substring insensible a mayúsculas/minúsculas usando contieneCadena y aMinusculas.
- Agrupación y eliminación de duplicados:
  - Agrupación por nombre/familia usando eliminarDuplicados.

- Ordenamiento:
    - Utilizado en analizarFrutasPopulares para ordenar frutas por monto total vendido.
- Resúmenes:
    - Construcción de totales por fruta/familia usando List.foldl para sumar cantidades y montos.
    - Funciones como frutaconmayorunidades, frutaconmayormonto y familiaconmayormonto para encontrar máximos/mínimos.
- Persistencia:
    - Escritura por adición al final del archivo usando TextIO.openAppend para agregar nuevos registros.
    - Reescritura completa de archivo con encabezado en limpiarCatalogo usando TextIO.openOut.
    - Validación de existencia de archivos con archivoExiste antes de operaciones de lectura.

Librerías Usadas

- **TextIO (Entrada/Salida de texto):**
  - TextIO.inputLine: Para leer líneas completas desde archivos y entrada estándar.
  - TextIO.openIn / TextIO.closeIn: Para abrir y cerrar archivos en modo lectura.
  - TextIO.openOut / TextIO.openAppend / TextIO.closeOut: Para crear, abrir en modo escritura/adición y cerrar archivos.
  - TextIO.output: Para escribir contenido a archivos.
  - TextIO.stdIn: Para capturar entrada del usuario desde consola.
- **String (Manipulación de cadenas):**
  - String.substring: Para extraer subcadenas (eliminar saltos de línea).
  - String.size: Para obtener la longitud de cadenas.
  - String.explode / String.implode: Para convertir entre strings y listas de caracteres.
  - String.isPrefix: Para verificar prefijos en cadenas (detectar encabezados CSV).
  - String.isSubstring: Para búsquedas de subcadenas dentro de texto.
- **Int (Operaciones con enteros):**
  - Int.fromString: Para convertir strings a enteros con validación
  - Int.toString: Para convertir enteros a representación string.
- **Real (Operaciones con números reales):**
  - Real.fromString: Para convertir strings a números reales con validación.
  - Real.toString: Para convertir reales a representación string.
  - Real.fromInt: Para convertir enteros a reales en cálculos de totales.
- **List (Operaciones con listas):**
  - List.filter: Para filtrar elementos que cumplan condiciones específicas.
  - List.map: Para transformar elementos de listas (extracción de campos).
  - List.foldl: Para operaciones de agregación (sumas, totales, acumuladores).
  - List.app: Para aplicar efectos secundarios sobre elementos de listas (mostrar registros).
  - List.length: Para obtener el tamaño de listas.
  - List.tabulate: Para generar listas de elementos repetidos (líneas decorativas).
- **Tipos de datos opcionales:**
  - OPTION (SOME/NONE): Para manejar valores que pueden fallar (conversiones, lectura de archivos).
  - Patrón utilizado extensivamente para validación de entrada y parseo seguro de datos.

Análisis de resultados

Objetivo	Porcentaje de Cumplimiento	Justificación
Implementar el creador para agregar registros de frutas al catálogo	100%	Se desarrolló la función que recibe los datos de una fruta y los agrega al archivo CSV siguiendo el formato requerido.
Implementar la limpieza total del catálogo	100%	Se programó correctamente la función para limpiar el archivo y reiniciar el catálogo de frutas.
Mostrar ranking de frutas populares en un rango de ventas	100%	La función filtra y ordena por monto vendido, devolviendo los resultados en orden descendente.
Identificar familias con más de 4 frutas diferentes	100%	Se implementó un conteo por familia y se listaron aquellas con al menos cuatro frutas distintas.
Búsqueda de frutas por código o nombre	100%	Se logró aplicar filtros para mostrar todas las ventas asociadas a la fruta buscada.
Cantidad de frutas por familia	100%	La función devuelve el total y la lista de frutas que pertenecen a la familia ingresada.
Resumen general (fruta con más/menos ventas, familia con más ventas, etc.)	100%	Se implementó el resumen solicitado mostrando toda la información de forma clara.

Justificación de toma de decisiones

a. Uso de funciones de orden superior en cada implementación

- List.filter - Filtrado de datos:
  - Qué hace: Filtra elementos de listas que cumplan una condición específica.
  - Por qué se utilizó:
    - En buscarFrutasPorCodigoONombre: Filtra registros que coincidan con código o nombre de búsqueda.
    - En analizarFrutasPopulares: Filtra frutas dentro de un rango de monto específico.
    - Justificación: Permite aplicar criterios de filtrado de manera funcional, evitando bucles imperativos explícitos y haciendo el código más legible y mantenible.
- List.map - Transformación de datos:
  - Qué hace: Aplica una función a cada elemento de una lista.
  - Por qué se utilizó:
    - En aMinusculas: Transforma cada carácter a minúscula para búsquedas insensibles a mayúsculas.
    - Para extraer campos específicos como nombres o familias de registros de frutas.
    - Justificación: Evita mutabilidad y permite transformaciones puras de datos, manteniendo la inmutabilidad característica de la programación funcional.
- List.foldl - Agregación y acumulación:
  - Qué hace: Reduce una lista a un valor único mediante una función acumuladora.
  - Por qué se utilizó:
    - Para calcular totales de unidades vendidas y montos totales en análisis estadísticos.
    - En funciones como resumen de ventas por fruta y por familia.
    - Justificación: Proporciona una forma elegante y eficiente de realizar operaciones de agregación.
- List.app:
  - Qué hace: Aplica una función con efectos secundarios a cada elemento de una lista.
  - Por qué se utilizó:
    - Para imprimir listas de resultados de búsquedas y análisis.
    - Justificación: Permite manejar efectos secundarios (como imprimir) de manera controlada y funcional, separando la lógica de presentación de la lógica de datos.

- List.tabulate - Generación de datos:
    - Qué hace: Genera listas aplicando una función a índices secuenciales.
    - Por qué se utilizó:
        - Para crear líneas con caracteres repetidos en la interfaz de usuario.
        - Justificación: Proporciona una forma funcional de generar secuencias regulares sin bucles imperativos.