**Instituto Tecnológico de Costa Rica**

**Proyecto 1: Sistema de Análisis de Datos de Ventas**

Curso:

IC4700 – Lenguajes de programación

Profesor:

Allan Rodríguez Dávila

Estudiante:

**Mary Paz Álvarez Navarrete**

Carnet:  
2023138604

II Semestre 2024

**Manual de Usuario**

**Compilación y Ejecución**

1. Abra la terminal y entre al subdirectorio programa en la carpeta del proyecto.
2. Para compilar el proyecto use el siguiente comando:  
   gcc -o p DataAnalysis.c cJSONLibary/cJSON.c
3. Para ejecutar el programa una vez compilado use el siguiente comando:  
   ./p

**Uso**

Al ejecutar el programa, se mostrará un menú principal con cinco opciones para el usuario. A continuación, el uso de cada opción:

1. **A: Importar datos**

Cuando se selecciona esta opción, el usuario es llevado a un submenú con 2 opciones:

* **Importar archivo:** Se le pedirá al usuario ingresar una ruta del archivo “.json” que desee importar.
* **Volver:** Le permite al usuario volver al menú principal.

1. **B: Procesar datos**

Cuando se selecciona esta opción, el usuario es llevado a un submenú con 3 opciones:

* **Completar datos faltantes:** En esta opción se modificarán los datos de cantidad y precio unitario. Esta es una función automática por lo que el usuario no tendrá que proporcionar ninguna información.
* **Eliminar duplicados:** En esta opción se eliminarán las ventas duplicadas. Al igual que la opción anterior es una función automática por lo que el usuario no deberá de proporcionar ninguna información**.**
* **Volver:** Le permite al usuario volver al menú principal.

1. **C: Análisis de datos**

Cuando se selecciona esta opción, el usuario es llevado a un submenú con 3 opciones:

* **Ver total de ventas:** Esta opción le enseña al usuario el total general de ventas en memoria.
* **Ver total mensual y anual:** Muestra al usuario un reporte del total de ventas por año y por mes.
* **Volver:** Le permite al usuario volver al menú principal.

1. **D: Análisis temporal**

Cuando se selecciona esta opción, el usuario es llevado a un submenú con 4 opciones:

* **Mes con mayor venta:** Permite al usuario seleccionar un año y luego muestra el mes con más ventas de ese año y su total.
* **Dia de la semana más activo:** Al elegir esta opción, se pedirá primero el año y luego el mes. Después de esto se le mostrara el día de la semana con la cantidad de transacciones hechas.
* **Tasa de crecimiento o decrecimiento trimestral:** Solicita el año para mostrar la tasa de crecimiento trimestral para ese año.
* **Volver:** Le permite al usuario volver al menú principal.

1. **E: Estadística**

Cuando se selecciona esta opción, el usuario es llevado a un submenú con 2 opciones:

* **Top 5 categorías:** Se le mostrara al usuario las 5 categoría con más ventas , además de que también se mostrara el total hecho por cada una.
* **Volver:** Le permite al usuario volver al menú principal.

1. **S: Salir**

El programa le mostrar un mensaje indicando que se están guardando los datos y que está saliendo, después terminara la ejecución del programa.

El menú y los submenús se mostrarán siempre que una función termine de hacer una tarea. Si ocurre algún error el programa le mostrara un mensaje de error junto con unas instrucciones a seguir.

**Descripción del Problema**

El proyecto consiste en desarrollar un sistema de gestión de datos de ventas que permita importar, procesar, y analizar información almacenada en archivos JSON. A través de un menú interactivo por medio de la consola, el usuario podrá cargar datos, limpiar y procesar la información para corregir inconsistencias, realizar análisis estadísticos y temporales, y generar informes sobre las ventas. El programa deberá asegurar la persistencia de la información al guardar los datos en archivos adicionales. Además, el programa deberá hacer uso de estructuras en C para organizar los datos y utilizar librerías especializadas para manipular archivos JSON.

**Diseño del programa**

**Decisiones del diseño**

Para este proyecto, se ha optado por utilizar la consola de comandos como interfaz de usuario. Se han implementado mensajes claros y concisos para evitar saturar al usuario con demasiada información. Dado que el programa incluye un menú principal y submenús, se ha decidido ofrecer siempre la opción de volver y mostrar el menú o submenú actual al finalizar cada tarea.

Al iniciar el programa, la información almacenada en el archivo JSON "ventas.json" se carga en memoria. De manera similar, al salir del programa, la información en memoria se guarda de nuevo en el archivo JSON "ventas.json". Esto asegura que no se pierda información ni los cambios realizados durante el uso del programa.

El programa maneja archivos JSON y realiza asignación dinámica de memoria. En caso de que ocurra algún error relacionado con estos procesos, se proporcionarán mensajes de error claros al usuario para informarles sobre el problema y facilitar su resolución.

Para mantener la simplicidad y legibilidad del código, se ha decidido utilizar diversas funciones auxiliares en lugar de concentrar toda la lógica en una única función.

**Algoritmos usados**

Las funciones cleanMemory son usadas cuando se sale del programa, por si ocurre algún error o para limpiar el array para que vuela a ser utilizado.

**int main()**

Inicia el programa cargando todos los datos guardados en “ventas.js si existe algún problema mostrara un mensaje de error. Si no ocurren errores iniciara el programa mostrando el menú.

**void cleanMemorySales()**

Libera la memoria usada para el array de sales.

**void cleanMemoryReports()**Libera la memoria usada para el array years, month y days.

**void cleanMemoryCategories()**

Libera la memoria usada para el array categories.

**void cleanMemoryJson(cJSON \*json, char \*contentFile)**

Libera la memoria usada para guardar los objetos json y su contenido. Es usada cuanto se termina de cargar la información a la memoria y cuando ocurre algún error.

**void cleanBuffer()**

Es usada para limpiar el buffer después de usar getChar() para que el carácter \n no cause errores.

**void menu()**

Le muestra a los usuarios las diferentes funciones y los envia a la función elegida. Antes de usar una función que no sea importData() se asegurara que exista información en memoria para evitar errores.

**void verifyInfoInMemory()**

Verifica que el array de sales no sea nulo. Si lo es muestra el mensaje de error correspondiente y vuelve a menú.

**int loadMemory(char \*path, int mode)**Recibe como parámetro una ruta de archivo y un modo. En donde 0 es cargando programa y 1 que es una importación. Devuelve un valor entero. Siendo 0 que todo salió bien y -1 que hubo un error. Carga los datos del archivo JSON especificado en la ruta a un array de la estructura Sale. Si es una importación y hace falta un dato en cantidad, precio unitario o total se le hará saber al usuario y se guardará el dato con un 0.

**void importData()**

Le pedirá al usuario una ruta para poder importar los datos en memoria. Si la ruta no es válida u ocurre algún error se le mostrará un mensaje de error y se volverá a mostrar el submenú.

**void deleteDuplicates()**

Esta función elimina cualquier venta que esté duplicada, es decir, aquellas cuyo id de venta aparezca más de una vez. La función opera de manera automática: si se encuentra algún duplicado, se notificará al usuario. Del mismo modo, si no se detectan duplicados, se informará al usuario que no existen ventas duplicadas.

**int mode()**

A través de un array de estructuras, esta función calcula la moda de las cantidades de ventas diferentes de cero. Para ello, recorre el array sales y agrega al array quant el número de ventas y la frecuencia con la que se repite cada cantidad. Al finalizar la iteración, la función identifica el número que aparece con mayor frecuencia en el array y lo devuelve como la moda.

**int average()**

Calcula el promedio del precio unitario recorriendo el array y sumando los precios unitarios de todas las ventas que no sean cero. La función utiliza un contador para llevar la cuenta de cuántas ventas cumplen esta condición. Una vez finalizado el recorrido, se calcula el promedio del precio unitario dividiendo la suma total entre el número de ventas consideradas. Finalmente, la función devuelve el promedio calculado.

**void missingData()**

Esta función identifica las ventas que tienen un valor de 0 en la cantidad, el precio unitario o el total. Cuando se detecta una venta con la cantidad en 0, se calcula utilizando la moda; si el precio unitario es 0, se calcula utilizando el promedio. Después de calcular estos valores, se actualizan los datos en el array sales, y el total se establece temporalmente en 0 para recalcularlo correctamente como el producto de la cantidad por el precio unitario. Si se encuentran ventas con estas características, se mostrará un mensaje al usuario, y si no se encuentran, también se notificará al usuario.

**void processData()**

Le mostrara al usuario un submenú con las opciones de completar los datos faltantes, eliminar duplicados o volver al menú. Si la opción no es válida se le informara al usuario. De lo contrario se conducirá a la función de la opción ingresada.

**int getDay(const char \*date)**

Recibe una cadena de texto con la fecha. Calcula el día de la semana de la fecha dada y devuelve un índice como indicador del día. Empezando con 1 como Domingo y terminando con 7 como sábado.

**int getMonth(const char \*date)**

Recibe una cadena de texto con la fecha, la escanea y devuelve el número del mes.

**int getYear(const char \*date)**

Recibe una cadena de texto con la fecha, la escanea y devuelve el año.

**bool dayMonthYearSales()**

Llena el array year. Este contiene el año, cuanto se ganó en total ese año y el numero de meses. También contiene un array month en donde se guardan todos los meses de enero a diciembre en donde se guarda cuanto fue el total por mes. Además, el contiene un array con los días de la semana y la cantidad de transacciones. Esta función es usada en las opciones de análisis de datos y análisis temporal.

**char \*intCharMonth(int month)**

Recibe el número del mes y devuelve la cadena de texto para este. Por ejemplo, si recibe 2 devolverá “Febrero”.

**char \*intCharDay(int day)**

Recibe el índice indicado para el día de la semana y devuelve el día de la semana asociado a este. Por ejemplo, si se recibe 3 devolverá “Martes”.

**int totalSales()**

Recorre el array de sales y va sumando e total de cada venta. Este devuelve el total calculado.

**void printReport()**

Imprime el reporte de ventas hechas por anual y mensualmente. Esta función recorre el array de years y imprime el año, sus meses asociados, el total por mes y el total por año. Antes de imprimir el reporte se asegura de llenar el array con los datos de las importaciones.

**void analyzeData()**

Muestra al usuario un submenú con las opciones de ver total de ventas, ver ganancia anuales y mensuales y volver al menú.

**void mostSalesMonth()**

Le pregunta al usuario un año para que mostrar cual fue el mes en ese año en el que se ganó más. Si el año es invalido o no existe dentro del array years. Le notificara al usuario. Una vez encontrado el total con más ganancias se imprimirá en pantalla.

**void mostActiveDay()**

Le preguntara al usuario el año y el mes para mostrar cual fue el día de la semana con más transacciones. Se recorrerás el array de years para encontrar el día de la semana conmas transacciones y mostrárselo al usuario. Si el año o el mes son inválidos o no se encuentran en el array entonces se le indicara al usuario.

**void growthDeclineRate()**

La función solicitará al usuario que ingrese el año deseado y recorrerá el array years en intervalos de tres meses para calcular la tasa de crecimiento o decrecimiento. Existen tres posibles escenarios: si ambos meses son iguales a 0, se indica que no hubo ningún cambio durante el trimestre; si uno de los meses es 0, significa que hubo un crecimiento o decrecimiento del 100%; y si ambos meses son diferentes de 0, se aplicará la fórmula estándar para calcular la tasa de crecimiento o decrecimiento. Una vez realizados estos cálculos, se imprimirá la información correspondiente a los cuatro trimestres del año.

**void temporalAnalysis()**

Mostrar al usuario el submenú correspondiente a esta opción.

**void printCategories()**

Imprimirá el top 5 de las categorías recorriendo el array de categories. Si el número de categorías es menor a 5 imprimirá todas las categorías encontradas en el array.

**void swap(Category \*a, Category \*b)**

Recibe dos punteros a elementos de tipo Category. Intercambia las posiciones de ambos elementos en la memoria.

**int partition(int low, int high)**

Recibe los índices que delimitan una porción del array para realizar el algoritmo de ordenación. Se encarga de reorganizar los elementos de tal manera que los elementos menores que el pivote queden a su izquierda y los mayores a su derecha. Devuelve el índice del pivote, que es el punto de partición para las siguientes iteraciones de ordenación.

**void quickSort(int low, int high)**

Recibe el índice inicial y el índice final del array a ordenar. Divide el array en dos partes utilizando la función partition, y luego aplica recursivamente el algoritmo de ordenación rápida a cada subarray, hasta que toda la estructura esté ordenada.

**bool salesXCategory()**

Llena el array de categories buscando cada una de las categorías y agregándoles el total hecho por cada una de ellas. Esto lo hace aplicando la misma estrategia que con los reportes anuales.

**void estadistic()**

Muestra al usuario el submenú correspondiente a esta opción.

**void exitProgram()**

Esta función guarda todos los datos en memoria en 2l archivo “ventas.js” para que no se pierda ningún dato guarda y pueda ser cargado al iniciar el programa. Además de eso limpia la memoria usada durante el programa para evitar fugas de memoria. Por último, termina la ejecución del programa.

**Diagrama de archivos**

Para este proyecto, se utilizó la librería cJSON para el manejo de archivos JSON. Se incluyeron los archivos cJSON.h y cJSON.c necesarios para trabajar con JSON. En lugar de crear archivos adicionales por separado debido a problemas de con el compilador, se optó por consolidar el código en un archivo: DataAnalysis.h y DataAnalysis.c. Esta decisión simplificó el proceso de compilación y facilitó la gestión del proyecto.

**Librerías Usadas**

**<stdio.h>**

Se usó para el manejo de entrada y salida. Se usaron las funciones fprintf() y scanf().

**<stdlib.h>**

Es usada para manejar la memoria dinámica a los arreglos y a las cadenas de texto. Además del control de procesos. Se uso exit(), malloc(), realloc() y free().

**<string.h>**

Es usada para la manipulación de cadenas de texto. Las funciones usadas fueron strlen(), strcpy() y strcmp().

**<ctype.h>**

Es usada para convertir o validar caracteres. Se usó la función toupper().

**<stdbool.h>**

Maneja el uso de operaciones booleanas. Se añadió para poder usar variables booleanas.

**<time.h>**

Ayuda con la manipulación de la fecha y el tiempo. Se uso el objeto tm, time\_t, mktime() y localtime.

**<math.h>**

Contiene funciones matemáticas básicas. Se usó fabs().

**<sys/types.h>**

Proporciona definiciones de datos usados en operaciones de sistemas.

**cJSONLibary/cJSON.h**

Una biblioteca de terceros usada para la manipulación de archivos JSON. Se usó para leer y modificar archivos JSON.

**Análisis de resultados**

**Objetivos alcanzados**

* **Implementación del menú**: Se ha desarrollado un menú funcional que permite la navegación a través de las diferentes opciones del programa.
* **Carga de datos al inicio**: Los datos se cargan en memoria automáticamente al iniciar el programa, asegurando que toda la información necesaria esté disponible para su procesamiento.
* **Guardado de datos al salir**: La información en memoria se guarda en un archivo JSON al cerrar el programa, evitando la pérdida de datos y preservando los cambios realizados.
* **Importación de datos desde JSON**: Los datos se importan desde un archivo JSON a una estructura en memoria, facilitando el manejo y análisis de la información.}
* **Notificación de datos faltantes**: El programa informa al usuario cuando faltan datos necesarios, mejorando la experiencia de usuario al garantizar que todos los campos requeridos estén completos.
* **Modificación y notificación de datos faltantes**: Permite al usuario modificar los datos faltantes e informa cuáles fueron los datos que se corrigieron.
* **Eliminación de ventas duplicadas**: Identifica y elimina ventas duplicadas, informando al usuario sobre las ventas que fueron eliminadas.
* **Cálculo del total general de ventas**: Calcula el total de ventas en memoria, proporcionando un resumen financiero general.
* **Reporte mensual y anual**: Genera y muestra un reporte de ventas por mes y por año, ofreciendo una visión detallada del desempeño de ventas.
* **Análisis del mes con más ventas**: Identifica el mes con el mayor volumen de ventas, proporcionando información sobre los períodos de mayor actividad.
* **Análisis del día con más transacciones**: Determina el día con la mayor cantidad de transacciones, ayudando a identificar patrones de actividad.
* **Cálculo de tasas de crecimiento y decrecimiento trimestral**: Calcula las tasas de crecimiento y decrecimiento de ventas por trimestre, proporcionando una perspectiva sobre el rendimiento a lo largo del tiempo.
* **Estadísticas del top 5 categorías más vendidas**: Proporciona una estadística detallada de las cinco categorías más vendidas, destacando las áreas de mayor éxito.

**Objetivos no alcanzados**

* **Exportación a PDF del top 5 categorías**: No se logró implementar la funcionalidad para exportar la estadística del top 5 categorías a un archivo PDF. Esto debido a problemas con la librería que se estaba usando.

**Enlace al Repositorio de GitHub**

<https://github.com/Mary-Paz-AN/PY1_AnalisisVentas.git>

**Referencias**

<https://www.youtube.com/watch?v=0YVrLNhKVc8>

<https://www.geeksforgeeks.org/cjson-json-file-write-read-modify-in-c/>

<https://www.geeksforgeeks.org/quick-sort-algorithm/>

<https://nextscenario.com/es/formula-del-porcentaje-de-crecimiento-un-analisis-detallado/#:~:text=La%20f%C3%B3rmula%20para%20calcular%20el,100%20para%20obtener%20el%20porcentaje>