FACULTAD DE INGENIERÍA Y CIENCIAS

UNIVERSIDAD ADOLFO IBÁÑEZ

INFORME DE DISEÑO Y JUSTIFICACIÓN DE LA SOLUCIÓN

TICS 200 - Lenguaje y Paradigmas de la Programación

Grupo 3
Carlos Escobar
Jorge Rivas
Diego Subiabre

1. Objetivo: Explicar de manera breve cómo se organizó el código, las estructuras de datos empleadas (struct order, etc.), la justificación de la modularidad y el uso de punteros a funciones.
    a) **Diagrama de flujo general que muestre el flujo principal del programa.**

`main.c` es el controlador principal.

`parser.c` carga los datos desde el archivo CSV.

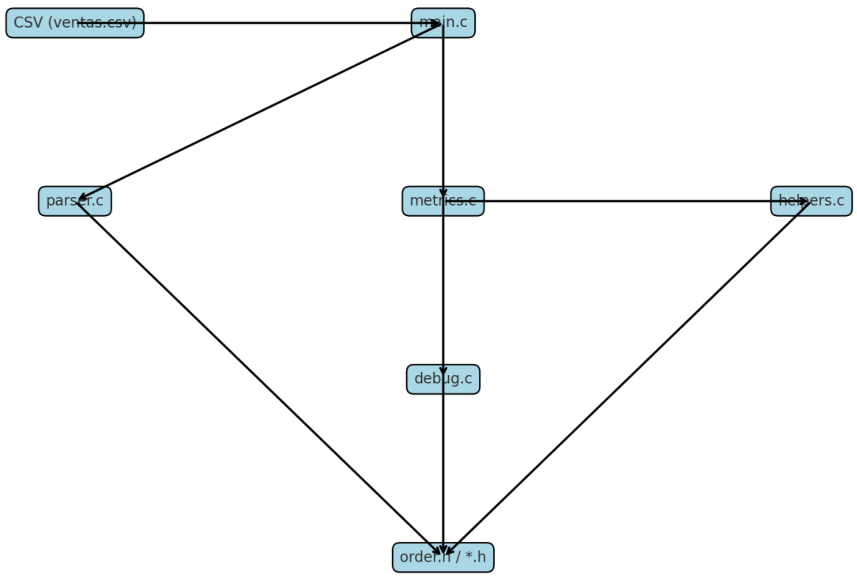
`metrics.c` gestiona las métricas solicitadas por el usuario.

`helpers.c` proporciona funciones auxiliares comunes, como la de fecha.

`debug.c` ofrece herramientas para visualizar internamente los datos.

Todos estos módulos interactúan a través de las definiciones declaradas en los headers (`order.h`, `parser.h`, `metrics.h`, etc.).

Diagrama de Arquitectura App 1 (Análisis de Pizzas en C)



b) Razones de diseño (por qué se eligió un determinado método de parseo del CSV, cómo se almacenan ingredientes, etc.).

1. **Método de parseo del CSV:**

   Se utilizó fgets para leer línea por línea del archivo CSV, lo que permite manejar archivos de tamaño arbitrario sin cargar todo en memoria, evitando overflow de esta y posibles errores.

   sscanf lo empleamos para extraer los campos principales del CSV hasta pizza_category, ya que es eficiente para datos estructurados.

   Para campos complejos como pizza_ingredients, utilizamos una combinación de strchr y manejo manual de cadenas para localizar y extraer datos entre comillas dobles.

   Este enfoque es modular y permite manejar errores en partes específicas sin dañar el resto del código.

2. **Almacenamiento de ingredientes:**

   Los ingredientes se almacenan como una cadena en el campo pizza_ingredients del struct Order. Esto simplifica el almacenamiento inicial y permite procesarlos dinámicamente (por ejemplo, separarlos por comas) cuando sea necesario.

3. **Uso de estructuras de datos:**

   El struct Order encapsula toda la información de una orden, lo que facilitó el manejo de datos relacionados. Para métricas específicas, se utilizaron estructuras auxiliares como Conteo (para pizzas) e Ingrediente (para ingredientes), lo que permite realizar cálculos sin modificar la estructura principal.

4. **Punteros a funciones:**

   Las métricas se implementaron como funciones con la firma char* (int*, Order*) y se almacenan en un arreglo de punteros a funciones (Metricas disponibles[]). Esto permite seleccionar y ejecutar dinámicamente las métricas solicitadas por el usuario, mejorando la extensibilidad del programa. Este diseño permite **agregar nuevas métricas sin modificar** main.c, solo agregando una función y registrándola.

c) Explicación de la interacción entre archivos (main.c, metrics.c, utils.c, etc.). Referencias a recursos externos utilizados.

Main.c: Es el punto de entrada del programa. Maneja la interacción con el usuario, como la carga del archivo CSV y la selección de métricas.Llama a funciones de otros módulos (parser.c, metrics.c, etc.) para realizar tareas específicas.

Parser.c: Contiene la función parse_csv, que se encarga de leer y procesar el archivo CSV. Devuelve un arreglo dinámico de structs Order que se utiliza en todo el programa.

Metrics.c: Implementa las funciones de cálculo de métricas, como pms (pizza más vendida) y dms (fecha con más ventas).Utiliza funciones auxiliares de helpers.c para tareas comunes, como agrupar fechas únicas.

Helpers.c: Proporciona funciones de utilidad, como agrupar_fechas_unicas, que se utilizan en varias métricas para procesar datos.

Debug.c: Contiene funciones para depuración, como debug_pizzas, que imprime información sobre las pizzas vendidas.

metrics.h, parser.h, helpers.h, debug.h: Declaran las funciones y estructuras utilizadas en sus respectivos módulos, promoviendo la modularidad y la separación de responsabilidades.

2. .**¿Qué fue lo más complejo o interesante de la tarea?, ¿Cómo enfrentaron los errores, pruebas y debugging?, ¿Qué lecciones aprendieron al implementar en C este tipo de lectura de archivos y cálculos de métricas?**

Una de las cosas interesantes de la tarea a la hora de construir la App 1 en C fue programar bajo el paradigma procedural. Al modularizar el código fue más eficiente la división de tareas. Además, cuando se nos presentaban errores, pruebas o debugging fue más fácil entender el error y solucionarlo en el módulo en específico.

Otro de los aspectos complejos pero interesantes fue tener que trabajar de forma remota y simultánea a través de github, una herramienta nueva que tuvimos que aprender desde cero su funcionamiento y diversas técnicas de colaboración, teniendo siempre en consideración a los otros integrantes para así no afectar su progreso. Aunque haya sido difícil, nos quedó la habilidad que luego será muy importante en nuestra vida laboral.

Por otra parte, cuando fuimos probando el código, hicimos una función debug_pizzas para

inspeccionar el estado de los datos. También realizamos pruebas en el main para ver si la función parser estaba realizando bien su trabajo. Para esto, utilizamos una función auxiliar para imprimir lo que el programa estaba leyendo mediante las funciones fgets.
Los archivos además, fueron probados por separados, y las funciones se verificaron con datos controlados, haciendo ciertas modificaciones en el archivo ventas.csv para comprobar si estaba calculando correctamente las métricas.

Una de las cosas que pudimos aplicar en la realización de este trabajo es el manejo de string en C, que es radicalmente distinto al de python. Los String en C al ser un array de caracteres, hay que tratar con sumo cuidado cuando buscamos por ejemplo, igualdad entre cadenas de texto, ya que no se comparan de la misma forma que en Python. Otra cosa a tener en cuenta siempre, es que los string en C tienen un carácter de término \0, que ocupa la última posición del array. Esto en gran parte del trabajo significó revisar cómo estaba nuestro programa tratando los string.

3. **Si se utilizó ChatGPT u otra herramienta, describir qué tipo de ayuda brindó la IA (e.g., sugerencia de estructuras, debug, explicación de errores) y cómo validaron dicha información.**

Se utilizó Chat GPT para explicación de errores y para la creación de código basado en la lógica definida previamente en nuestra app. Luego de la creación del código se utilizó para entender y evaluar la mejor opción en base a nuestro programa. La ayuda y recomendación fue validada empíricamente, fijándonos que cumpliera con los requisitos de la tarea. En momentos de bloqueo mental nos ayudó a darle otra perspectiva al problema, permitiéndonos finalmente superar la mayoría de obstáculos que se fueron presentando.
Además, nos sirvió para validar ciertas decisiones, como el hecho de mantener dos estructuras auxiliares para las fechas y para los ingredientes, así como también una función en particular que sería utilizada por métricas relacionadas a la fecha.
Por otra parte, nos ayudó a comprobar si nuestro código estaba bien en sintaxis, bien modularizado y poder hacer el Makefile de manera correcta.

Ejemplo de consulta en ChatGPT:

Para crear la métrica pizza más vendida:

“Estoy haciendo un programa en C para una pizzería. El programa maneja las órdenes de los clientes y cada orden tiene el nombre de la pizza y cuántas pidieron. Necesito una función que me diga cuál fue la pizza más vendida. La idea es que la función recorra todas las órdenes, agrupe las pizzas por nombre, sume cuántas se han vendido en total de cada una, y luego me diga cuál fue la más popular. Si no hay existen ordene devuelve el mensaje "Pizza más vendida: Sin datos". Además, necesito que la función me devuelva ese resultado como un

string, con el formato "Pizza más vendida: <nombre de la pizza>". También me gustaría que me explicaras cómo funciona el código para que discutamos la estructuras de las funciones."
Para crear otras métricas siguiendo la lógica y estructura de la primera:

"Ahora, necesito que con la misma lógica y parámetros me ayudes a construir la métrica que me entregue la mayor cantidad de ingredientes. Estos, se encuentran en un string separados por comas. La función debe separar los ingredientes e ir agregándolos en un array para que luego me sume el que más aparece en el array"