UNIVERSIDAD ADOLFO IBÁÑEZ

Informe App 1

Lenguajes y paradigmas de programación

Profesora: María Loreto Arriagada

Ayudante: Diego Duhalde

Integrantes:

Gonzalo Eyzaguirre

Rebeca Fernández

Tomás López

Agustín Zapata

Fecha: 03/04/2025

Grupo 12

Sección 1

Resumen

El código de la App 1 se encuentra organizado de una manera modular, ya que se encuentra dividido en secciones individuales para su mejor funcionamiento y entendimiento, en primer lugar, definiendo la estructura para almacenar la información de cada pedido, continuando con la creación de las funciones para las distintas métricas que son pedidas y la lectura del archivo .csv, para terminar con la parte principal del código donde primero se almacena la información de la base de datos para después mediante de la función de *if*, reconocer la información que es pedida por el usuario llamando a las funciones ya creadas anteriormente, y mostrando la información requerida, para finalizar liberando la memoria, cerrar el archivo de las ventas y cerrar el programa.

La modularidad que presenta el código es principalmente debida a la facilidad de trabajo que presenta ya que la corrección de errores es mucho menos compleja, por su visualización a la hora trabajar en grupo es mucho más legible entender el paso a paso. Otro punto importante fue el poder probar funciones individualmente a la hora de que se avanzó en el proyecto.

Los punteros en el código se usan principalmente para modificar los datos sin hacer copias de estos, permitiendo trabajar con los datos de manera más eficiente con relación a la memoria utilizada, evitando errores por memoria en el código.

El parseo del archivo .csv en el código se realiza utilizando la función, que lee cada línea del archivo, manejando tanto los campos separados por comas. Esta función toma un puntero que señala la posición actual en la línea, extrayendo los valores hasta encontrar una coma, y luego los asigna a los campos correspondientes de la estructura. De esta manera, se leen correctamente las líneas del archivo, asignando los valores a las variables de cada orden de pizza.

En conclusión, el código de la App 1 está bien organizado de manera modular, lo que facilita su comprensión, mantenimiento y depuración. Al dividirse en secciones específicas,

como la definición de estructuras para almacenar la información de los pedidos, las funciones para calcular diversas métricas, y la lectura y procesamiento del archivo .csv, se logra una mayor claridad en su funcionamiento. El uso de punteros es esencial para la eficiencia en la manipulación de datos, ya que permite modificar directamente la información sin hacer copias innecesarias, optimizando el uso de la memoria y reduciendo posibles errores. El parseo del archivo .csv, gestionado por una función especializada, lee cada línea de manera eficiente, extrayendo y asignando correctamente los valores a las variables correspondientes en la estructura. En resumen, la modularidad, el uso de punteros y la correcta implementación del parseo de datos contribuyen a un código eficiente, legible y fácil de mantener.

Reflexiones

Gonzalo Eyzaguirre: A la hora de realizar esta aplicación lo más complicado fue entender el uso de la memoria y como pedirla y liberarla durante el código, ya que era un concepto nuevo. Los errores que fue presentando el código a la hora de hacerlo los fuimos trabajando con la ayuda de la IA lo cual nos fue muy concluyente debido a los errores que presentaba y las soluciones erróneas que nos daba, pero que terminamos solucionando mediante investigación y sugerencias a la IA. Para la lectura de los archivos en C tuvimos grandes problemas con su formato, para lo que fue clave saber cómo se dividía el archivo por la cantidad de problemas que tuvimos con su división y especialmente con las comas. Para el cálculo de las métricas notamos que fue clave el uso de las funciones para evitar errores y manejarlos de mejor manera.

Rebeca Fernández: Lo más interesante dentro de la tarea fue resolver la lectura del archivo CSV, ya que en un comienzo teníamos la separación por cada coma que detectara al leer linea por linea, sin embargo, recibimos como resultado que el segundo ingrediente hacia delante lo detectaba como nombre de pizza, por lo que tuvimos que realizar un cambio al leer el archivo de comillas dobles (“”) a slash (/) para contener todo lo que está dentro de ellas como ingredientes y poder tener una correcta separación del archivo. Una vez

resuelto, presentamos un problema de manejo de espacio de memoria, por lo que tuvimos que aprender más sobre estructuras y liberación de espacio (malloc), para ello, nos apoyamos de la IA para un manejo correcto y no presentar ese tipo de problemas. Luego de muchas iteraciones conseguimos tener una buena lectura del csv con el manejo de memoria controlado. Al implementar C, nos pudimos percatar que saber usar la memoria de manera eficiente simplifica el código en temas de complejidades espaciales y temporales, además del momento de realizar las métricas que requiere comprender la lógica de los algoritmos y tener claridad de lo que se buscaba.

Tomás López: De las cosas más complicadas de la tarea fue entender el funcionamiento de un lenguaje de programación nuevo y poco amigable, los errores que presentamos a la hora de probar el código se enfrentan a base de prueba y error, donde uno de los problemas principales fue el uso de la memoria que se usa en c, y entenderlo fue muy complejo. Para la lectura de los archivos fue clave entender el contenido del mismo, debido que presentamos muchos problemas con la lectura y el guardado de información del mismo, para el cálculo de métricas fue clave entender cómo se almacenaban los datos, y cómo trabajar con la memoria para liberarla al final.

Agustín Zapata: Para mí lo más interesante fue conocer más a fondo C. Previamente ya tenía un poco de experiencia con C# pero ahora vi las diferencias entre los 2 lenguajes y así añadir otra opción de lenguaje para usar en proyectos, ya que, C es bastante eficiente con el manejo de memoria lo que resulta útil para aplicaciones que requieran un manejo más estricto de esta. Para enfrentar problemas como errores y debugging simplemente seguí los mismos pasos que con otros lenguajes, que es ver dónde está el error y comprobar si la sintaxis está correcta, si no consulto con la documentación o busco el problema en foros como Stack Overflow, aunque también cuando uso un lenguaje con el que no soy muy familiarizado le hago consultas a Copilot. En mi caso aprendí la estructura y la manera de usar C con WSL, ya que, nunca antes pensé en usar Linux. También aprendí la manera en que C gestiona los argumentos desde el terminal directamente en el main().

IA

La inteligencia artificial fue utilizada en varios aspectos relacionados con la app1 desarrollada. Uno de los aspectos más importantes fue su uso para comprender el lenguaje de programación, ya que gracias a esto fue posible entender cómo se maneja este idioma al programar.

Otro aspecto en el que resultó útil fue en la identificación y corrección de errores. Al ser nuevos en este lenguaje, no se tenía claridad sobre los problemas que surgían al ejecutar el programa, lo que dificultaba el avance en la app1. Sin embargo, aunque Chat GPT cometía errores en ocasiones, sí proporcionaba ciertos tipos de referencias útiles. Cuando se solicitaba una lista de posibles errores en un programa, la IA generaba un listado basado en el problema detectado. No obstante, en algunas situaciones, omitía el tipo de error específico, lo que obligaba a buscar ayuda adicional.

Asimismo, la inteligencia artificial contribuyó en la generación de ideas para agregar elementos al código o en la elección de estructuras adecuadas para organizarlo mejor. De esta manera, proporcionaba pasos detallados a seguir para lograr que el código resultara más ordenado y comprensible. De lo contrario, aunque el código funcionara, su desorden dificultaría su interpretación.

Estas fueron las principales formas en las que la IA brindó ayuda, aunque también apoyó en otros aspectos menos relevantes. Para verificar la información proporcionada por la IA, se recurrió principalmente a la lógica, descartando códigos que carecían de variables y funciones necesarias. Otra estrategia fue poner a prueba el código generado: si presentaba múltiples errores, se revisaba para intentar corregirlos, ya que su estructura podía ser aprovechada. Sin embargo, en varias ocasiones, a pesar de señalar un tipo específico de error, la IA generaba un código diferente que contenía el mismo problema, lo que suponía un obstáculo para avanzar en el trabajo.