UNIVERSIDAD ADOLFO IBÁÑEZ

Tarea 1: Pizzería en C.

Integrantes: Claudia Sofía Meneses - Isidora Henríquez - Javier Torres - Benjamín Ventura

Profesora: Loreto Arriagada

Entrega: Miércoles, 2 de abril, 22:00 hrs.

Introducción:

El objetivo principal de esta tarea fue desarrollar una aplicación en C que pudiera leer un archivo CSV con datos de ventas de una pizzería y calcular diversas métricas solicitadas desde la consola. Entre las métricas solicitadas se incluyen la pizza más vendida, el promedio de pizzas por orden, el ingrediente más frecuente y otras estadísticas clave para el negocio. Para lograr este objetivo, se utilizó una estructura modular en el código, donde se organizaron diferentes funciones para cada métrica, optimizando el mantenimiento y la extensibilidad del programa.

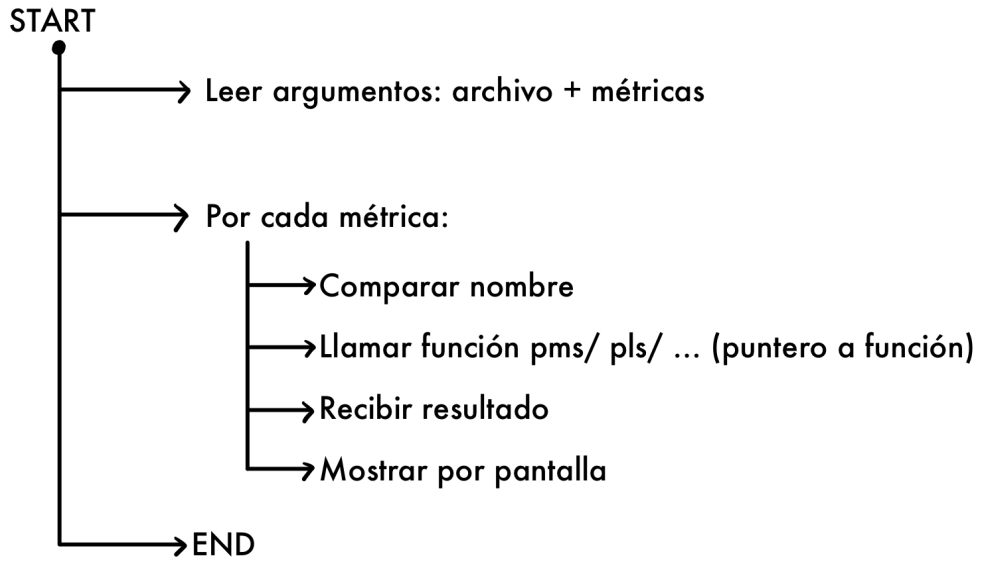
El proyecto implicó un enfoque en el manejo de archivos, trabajando con funciones como fopen(), fgets() y strtok()para leer y procesar los datos de ventas. Además, se trabajó con estructuras de datos (como struct) para organizar la información, lo que facilitó el cálculo de las métricas. Se hizo uso de punteros a funciones para mejorar la flexibilidad del código y permitir la ejecución dinámica de las métricas solicitadas. A lo largo del desarrollo, se enfrentaron retos técnicos que permitieron a los integrantes del grupo aprender sobre la gestión manual de la memoria, la manipulación de cadenas de texto y la optimización del rendimiento en C.

Informe de diseño y justificación de la solución

Objetivo:

El objetivo de la tarea consiste en desarrollar una aplicación en C que leyera un archivo CSV con datos de ventas de pizzas y que pueda calcular métricas solicitadas desde la consola a partir de los datos, como por ejemplo: la pizza más vendida, promedio de pizzas por orden, ingrediente más frecuente, etc.
Para lograr esto se desarrolló un único archivo *.c* que se encuentra organizado de forma clara y ordenada. Se utilizaron estructuras (struct), lectura de archivos, manejo de strings y punteros, aplicando principios básicos de programación modular en C.

Diagrama de flujo:



Razones de diseño:

Durante el desarrollo del programa se tomaron diversas decisiones que permitieron abordar el problema de forma eficiente y clara:

- Lectura del archivo CSV: Se optó por leer el archivo CSV línea por línea utilizando *fgets()* en lugar de cargar todo el archivo en memoria para así evitar errores debido al tamaño de este.
- Uso de estructuras (*structs)*: En vez de utilizar arrays separados, se usaron estructuras para agrupar la información relacionada, lo cual facilitó el manejo y organización de los datos.
- Modularidad en el diseño: Aunque todo el código se encuentra en un único archivo .c, se organizó en bloques: funciones para cada métrica, funciones auxiliares y la función *main()*. Esto permite cierta facilidad en la lectura y mantenimiento del programa.

Cada métrica (*pms, pls, apo,* etc) se implementó como un función independiente, lo que mejora la organización interna del código, permitiendo utilizarlas fácilmente según se necesite.

- Manejo de strings: Se trabajó con cadenas de texto (strings) con el fin de procesar fechas, nombres e ingredientes. Se utilizaron funciones como *strock, strcpy, strcmp* y *strcat* para manejar de manera precisa y eficiente el texto.
- Flexibilidad gracias a punteros: Este diseño permite calcular cualquier combinación de métricas sin tener que modificar el programa. La función *main()* compara los nombres de las métricas a través del uso de *strcmp* y llama a la función correspondiente. Esto facilita el agregar nuevas métricas sin alterar el funcionamiento del código.

Explicación de la interacción entre archivos:

Tal como se mencionó anteriormente, en este proyecto se utilizó un único archivo fuente (*main.c*) que contiene todo el código necesario para ejecutar el programa. Aunque no se modularizó en archivos separados como *metrics.c* o *utils.c,* el código se encuentra organizado internamente en secciones claras.

Estas secciones corresponden a ser las siguientes:

- Definiciones de estructuras (*structs*): Utilizadas para agrupar datos como nombres, ingredientes y fecha.
- Funciones de métricas: Como *pms(), pls(), dms(),* entre otras, cada una implementadas por separado para facilitar su uso.
- Funciones auxiliares: Como *trimString()* para limpiar texto, utilizada en varias métricas y así garantizar consistencia en los datos.
- Función principal (*main*): Recibe los argumentos desde la consola, determina las métricas a calcular y muestra los resultados en pantalla.

Referencias a recursos externos utilizados:

- ChatGPT: Esta herramienta se utilizó para resolver ciertos errores al momento de ejecutar el programa, interpretar mensajes de compilación, proponer mejoras en el manejo de las cadenas de texto. Asimismo, fue utilizado para aprender a como conectar GitHub con Visual Studio, y a cómo utilizar la plataforma GitHub, ya que para la mayoría de nosotros era primera vez que utilizabamos esto. Todas las sugerencias fueron comprobadas manualmente y adaptadas al contexto del proyecto.
- Stack Overflow: Se consultaron respuestas específicas relacionadas con el uso de ciertas funciones estándar de C, como *fgets, strtok, strcpy, etc,* lo cual nos ayudó a resolver dudas técnicas durante el desarrollo del proyecto.

Sección de reflexiones finales o autoevaluación

¿Qué fue lo más complejo o interesante de la tarea?

Claudia Sofía Meneses: Lo más desafiante fue retomar la programación ya que es mi primer minor y hacer paralelos con Python. Me costó especialmente entender punteros a funciones y

su uso en la aplicación de métricas al CSV, ya que al principio tuve problemas al pasarlas como parámetros y ejecutarlas correctamente.

Benjamín Ventura: Lo más complejo fue integrar correctamente el código con los archivos necesarios para su ejecución, incluyendo el archivo de datos en formato Excel. Inicialmente, tuvimos problemas porque la estructura no era la adecuada, lo que impedía que el programa funcionará correctamente.

Isidora Henríquez: Lo más complejo para mí fue aprender el lenguaje C, junto con la plataforma GitHub. Sin embargo, al utilizar herramientas como YouTube y ChatGPT logré aprender como estos funcionaban y así poder llevar a cabo la tarea.

Javier Torres: Lo más complejo fue comprender la teoría detrás de la generación del ejecutable, especialmente dentro de la Shell en Replit. Fue un reto entender cómo utilizar gcc para compilar el código y convertirlo en un archivo ejecutable dentro de la plataforma, asegurando que funcionara correctamente.

¿Cómo enfrentaron los errores, pruebas y debugging?

Claudia Sofía Meneses: Se consultó *YouTube* y *ChatGPT* para aprender a resolver errores. Se llevaron a cabo pruebas unitarias en pequeñas partes del código y depuración paso a paso para identificar y solucionar problemas.

Benjamín Ventura
Se revisó la estructura del archivo CSV utilizando prints para verificar que los datos se leyeran correctamente. También se realizaron pruebas con archivos de ejemplo para asegurar que el programa los procesara bien.

Isidora Henríquez:
Se usó gdb para depurar los errores relacionados con punteros a funciones y se realizaron pruebas con distintas métricas para verificar que los resultados fueran correctos. También se vieron impresiones para asegurar que los parámetros se pasaban correctamente.

Javier Torres:
Al compilar en el código, se revisaron los mensajes de error que aparecían y se hicieron pruebas con fragmentos de código para asegurar que gcc generará el ejecutable correctamente. También se usó documentación para entender mejor cómo compilar el código.

¿Qué lecciones aprendieron al implementar en C este tipo de lectura de archivos y cálculos de métricas?

Claudia Sofía Meneses:
Al implementar la lectura de archivos y el cálculo de métricas en C, Claudia aprendió a manejar las cadenas de texto de forma más eficiente, utilizando funciones como strtok() para separar los datos del archivo CSV. También comprendió cómo C maneja la memoria de

manera manual, lo que la ayudó a ser más cuidadosa con el uso de punteros y evitar errores comunes al manipular cadenas.

Benjamín Ventura:
Benjamín aprendió a organizar los datos usando estructuras, lo que fue clave para calcular las métricas correctamente. Además, se dio cuenta de la importancia de manejar las cadenas adecuadamente al leer los archivos CSV, lo que permitió extraer la información de manera más eficiente.

Isidora Henríquez:
Isidora aprendió a utilizar punteros a funciones para hacer el código más flexible y modular. También se enfrentó al reto de manejar la memoria dinámica, lo que le permitió entender mejor cómo C gestiona los recursos y mejorar sus habilidades al trabajar con archivos CSV.

Javier Torres:
Javier comprendió cómo leer archivos CSV con fopen() y fgets(), y la importancia de estructurar el código de forma modular para calcular las métricas eficientemente. Esta experiencia le permitió mejorar sus habilidades para optimizar los cálculos y manejar los datos de manera más efectiva.

2.

Explicación de cómo usaron IA

Se utilizó ChatGPT como herramienta de apoyo en varios momentos del desarrollo del proyecto. En primer lugar, se recurrió a la IA para entender el manejo de punteros a funciones en C, especialmente al implementar las métricas. La IA sugirió cómo utilizar punteros a funciones para hacer el código más flexible y modular, lo cual fue una gran ayuda para estructurar el programa de manera eficiente.

Además, durante la depuración, la IA ayudó a explicar ciertos errores relacionados con la manipulación de cadenas y el uso incorrecto de strtok(), lo que permitió entender mejor cómo manejar las cadenas en C.

También se consultó a la IA sobre la mejor forma de organizar las estructuras de datos, y cómo optimizar el proceso de lectura de archivos CSV. La IA sugirió una estructura más eficiente para almacenar los datos, lo que permitió reducir redundancias en el código.

En cuanto a la validación de la información, se contrastó las sugerencias de la IA con documentación oficial de C y pruebas de código. Se probaron varias soluciones propuestas por la IA, y las que funcionaron correctamente fueron implementadas.

Conclusión

A través de esta tarea, el grupo pudo profundizar en el uso de herramientas y conceptos fundamentales en C, enfrentando diversos desafíos técnicos que permitieron fortalecer nuestras habilidades de programación. El trabajo con archivos CSV, cadenas de texto y estructuras de datos fue clave para entender cómo gestionar y procesar grandes volúmenes de datos de manera eficiente. La implementación de las métricas solicitadas, como la pizza más vendida o el promedio de pizzas por orden, permitió aplicar principios de programación modular, lo que hizo que el código fuera más organizado y fácil de mantener.

Además, el uso de punteros a funciones resultó ser una herramienta poderosa para hacer el programa más flexible y escalable, permitiendo que las métricas se ejecutarán dinámicamente según se necesitaran. A través de las pruebas, depuración y el apoyo de herramientas como ChatGPT, optimizamos el código y resolvimos errores relacionados con el manejo de punteros y cadenas, lo que mejoró la eficiencia del desarrollo.

Finalmente, esta experiencia no solo permitió mejorar nuestras habilidades en C, sino también aprender a enfrentar problemas de programación en un entorno más cercano a los desafíos del mundo real, donde la gestión de datos, la eficiencia y la modularidad son cruciales.