INSTITUTO TECNOLÓGICO DE COSTA RICA

TALLER DE PROGRAMACIÓN

mastermindV2

DOCUMENTACIÓN MASTERMINDV2

Año: 2025

Autor: Joshua Brenes Hernández

Contenido

Objetivos: 3
Requerimientos: 3
Definición: 6
Temas Investigados: 7
Marco teórico 7
Tabla de IA 9
Conclusiones del trabajo 13
Problemas – Soluciones: 13
Aprendizajes Obtenidos: 14
Estadísticas de tiempo: 15
Lista de revisión del proyecto 16
Referencias bibliográficas 17

Objetivos:

Realizar la actividad de mantenimiento de software.

- Continuar aplicando el ciclo completo de la metodología general de desarrollo de programas: entender el problema, diseñar algoritmo, codificar algoritmo y probar y evaluar programa

- Aplicar y reforzar aspectos del lenguaje Python 3.

- Implementar el tipo de datos abstracto (TDA) pila.

- Implementar el tipo de datos abstracto árbol binario de búsqueda (ABB). El manejo de este árbol debe utilizar la técnica de recursión.

- Aplicar conceptos de programación orientada a objetos (POO): clases, atributos, métodos y objetos.

– Aplicar buenas prácticas básicas de programación: documentación interna y externa del programa, nombres significativos, desarrollo y reutilización de funciones, uso de programación modular (técnica divide y vencerás: dividir el proyecto en partes, luego desarrollar cada una de esas partes), seguir un estilo de programación, etc.

- Validar los datos de entrada.

- Usar archivos de datos.

- Usar software de control de versiones.

- Fomentar la habilidad de investigación: los aspectos investigados deben ser explicados detalladamente en la documentación del proyecto. Para este proyecto los temas mínimos a investigar son:

Software de control de versiones y colaboración: ¿A qué se refiere este software y su importancia en el desarrollo de software?

Mencionar y explicar brevemente tres software de ese tipo.

Diferencia entre git y github.

Explicar las características de git (github) que usó en este proyecto.

Requerimientos:

1. Botones de Deshacer Movimiento y Rehacer Movimiento

Se agregan dos botones nuevos dentro del juego.

Deshacer Movimiento

- Solo funciona dentro de la jugada actual y antes de calificar.
- Deshace el último movimiento realizado, devolviendo la casilla al valor anterior.
- Se implementa usando una pila:
    - pila_movimientos_hechos
    - Cada jugada inicia con esta pila vacía.
    - Cada movimiento se guarda en la pila como (casilla, valor_anterior, valor_nuevo).
    - Al presionar Deshacer, se saca el último movimiento y se restaura el valor previo.

Rehacer Movimiento

- Solo funciona cuando se han deshecho movimientos y antes de calificar.
- Reconstruye el último movimiento que se deshizo.
- Usa otra pila:
    - pila_movimientos_deshechos
    - Al deshacer, el movimiento eliminado se pasa a esta pila.
    - Al presionar Rehacer, se aplica el último movimiento guardado y se vuelve a pasar a pila_movimientos_hechos.

2. Nuevo grupo de elementos en el panel de configuración

Actualmente los grupos son:

- Colores
- Letras
- Números

Se debe agregar un cuarto grupo:
Otros símbolos (por ejemplo: *, +, /, -, >, <)

El jugador puede cambiar los elementos en cualquier momento.

3. Juegos Multinivel

Permite avanzar automáticamente por niveles.

Funcionamiento:

1. El jugador empieza en Fácil.
2. Si gana, pasa automáticamente a Medio.
3. Si gana en Medio, pasa automáticamente a Difícil.
4. Si gana en Difícil, sigue jugando en Difícil.
5. Cada nivel registra su marca por separado.

Cambios en Configurar:

- En Nivel de dificultad, agregar la opción Multinivel.
- Si Multinivel usa temporizador:
    - Se deben pedir los tiempos máximos para cada nivel (horas, minutos, segundos):
        - Nivel Fácil
        - Nivel Medio
        - Nivel Difícil

4. Funcionalidad adicional definida por el programador

Debo agregar una mejora personal.
Puede ser pequeña pero útil

5. TOP-10 se transforma en REGISTRO DE MARCAS (ABB)

En vez de guardar solo las 10 mejores marcas, ahora se deben guardar todas.

Se usará un Árbol Binario de Búsqueda (ABB):

- Cada marca es un objeto de la clase Marca.
- Cada nodo del árbol es un NodoABB que almacena una marca y dos hijos.
- Se deben implementar métodos recursivos:
    - Insertar nodo de forma ordenada según tiempo.
    - Recorrer el árbol en orden (para obtener las marcas ordenadas de menor a mayor).

Tres árboles ABB:

- abb_facil
- abb_medio

- abb_dificil

Cada marca se inserta en el árbol del nivel que corresponda.

Cambios en el menú:

- “Top-10 (Resumen)” => Resumen de Marcas
- “Top-10 (Detalle)” => Detalle de Marcas
- Ahora muestran todas las marcas ordenadas ascendentemente con recorrido en-orden.

6. Opción de Ayuda

Debe mostrarse en pantalla el manual del usuario actualizado, que incluya:

- Deshacer/Rehacer
- Nuevo grupo de símbolos
- Juegos multinivel
- Registro de marcas (ABB)

Definición:

En la ingeniería de software el ciclo de vida de desarrollo de software (SDLC: System Development Life Cycle) se refiere al conjunto de etapas necesarias para la creación y utilización de software a través del tiempo. Hay diferentes metodologías que soportan este ciclo de vida, algunas de ellas son: modelo en cascada (el más antiguo), modelo en espiral, prototipos, modelo incremental e iterativo, desarrollo ágil (usando herramientas como SCRUM, programación extrema-XP, Kanban, etc.). Dentro de las metodologías de desarrollo está la etapa de mantenimiento de software. El mantenimiento de software es una actividad natural del ciclo de vida del software, por eso es muy común que nos encontremos trabajando en ello. El objetivo del mantenimiento de software es garantizar que el mismo siga vigente a lo largo del tiempo, lo cual se logra mediante un proceso de mejora continua que involucra estas actividades:

- Corregir errores

- Adaptar el software a cambios en el entorno (cambios en hardware, software, regulaciones gubernamentales, otros)

- Mejoras (nuevas funciones, mejor rendimiento, mayor seguridad, otros) En este proyecto vamos a hacer mantenimiento al software realizado en el proyecto

anterior: específicamente vamos a hacer mejoras ya que agregaremos nuevas funcionalidades.

Temas Investigados:

Marco teórico:

Software de control de versiones.

El software de control de versiones es una herramienta que ayuda a guardar y organizar todos los cambios que se hacen en un proyecto, especialmente cuando se trabaja con muchos archivos. Cada vez que alguien modifica algo, el sistema guarda una copia del estado del proyecto, lo que permite volver atrás si hay un error, comparar cómo estaba antes o saber quién hizo cada cambio. Según Microsoft (2023), este tipo de software también facilita que varias personas trabajen al mismo tiempo sin estorbarse, ya que cada integrante puede hacer ajustes por separado y luego unirlos al proyecto principal de forma segura. Esto evita pérdidas de información, confusiones o que un archivo se sobrescriba por accidente.

La importancia en el desarrollo de software es muy grande porque permite trabajar en equipo de manera ordenada y responsable. Gracias al control de versiones, los grupos pueden probar ideas nuevas sin arriesgar el proyecto, corregir errores con facilidad y mantener todo el trabajo bien registrado. En proyectos grandes, donde muchas personas hacen modificaciones todos los días, estas herramientas garantizan que el proyecto avance de forma organizada y segura. Por eso, el control de versiones es considerado una base esencial para cualquier equipo que quiera desarrollar software de manera profesional.

Tipos de software de control de versiones.

Según BuiltIn (2023) tres tipos de estos software son:

- **Git:** Es un sistema de control de versiones distribuido, lo que significa que cada desarrollador tiene una copia completa del proyecto en su computadora. Esto permite trabajar sin conexión, crear ramas para probar nuevas ideas y luego unirlas al proyecto principal. Es rápido, flexible y el más utilizado en la industria.
- **Subversion (SVN):** Es un sistema de control de versiones centralizado. Todo el código se almacena en un servidor principal, y los desarrolladores descargan y suben cambios allí mismo. Es sencillo de usar y suele usarse en equipos que necesitan un control más estricto sobre el acceso y la organización del código.
- **Mercurial:** También es un sistema distribuido, similar a Git, pero diseñado para ser más simple y fácil de aprender. Permite trabajar con ramas,

mantener historiales completos y colaborar sin depender siempre de un servidor, siendo una opción ligera y eficiente.

Diferencia entre Git y GitHub

Según lo explicado por Kinsta (2023), Git es un sistema de control de versiones que se instala en el computador del usuario y permite llevar un registro de todos los cambios realizados en un proyecto. Con Git, cada desarrollador puede trabajar de manera local, crear ramas para probar nuevas ideas, comparar versiones anteriores y restaurarlas si es necesario, incluso sin conexión a internet. Su función principal es mantener el historial del código organizado y facilitar el trabajo individual o en equipo de forma segura y estructurada.

Por otro lado, GitHub es una plataforma en línea que utiliza Git como base, pero agrega herramientas extras que facilitan la colaboración entre varias personas. Almacena los proyectos en la nube y permite que cualquier miembro del equipo acceda al repositorio desde cualquier lugar. Además, GitHub añade funciones como solicitudes de cambios (pull requests), revisión de código, seguimiento de errores, gestión de tareas y un sistema social que permite seguir proyectos y desarrolladores. Mientras Git se encarga del control de versiones, GitHub proporciona un entorno colaborativo donde esos proyectos pueden compartirse y desarrollarse en conjunto.

características de git (github) utilizadas en este proyecto

Gestión de versiones del código

Se implementó Git como sistema de control de versiones distribuido con el fin de conservar un seguimiento cronológico de todas las modificaciones efectuadas en el programa.
Esto permitió:

- Mantener un historial estructurado del desarrollo.
- Comparar versiones previas del código.
- Recuperar estados estables cuando se introdujeron errores.
- Facilitar el mantenimiento evolutivo y correctivo.

Comandos utilizados con frecuencia:

- git init
- git add
- git commit

Commits estructurados y mensajes explicativos

Para garantizar la transparencia del proceso de mantenimiento, se realizaron commits frecuentes con mensajes claros y descriptivos. Esto favoreció:

- Identificar con precisión qué se modificó en cada etapa.
- Ubicar fácilmente la introducción o corrección de errores.
- Documentar cada mejora obligatoria (ABB, símbolos personalizados, repetidos, multinivel, video de victoria, etc.).

Manejo de archivos del proyecto

Git permitió administrar de manera ordenada:

- Archivos .py (código fuente).
- Archivos .dat (configuraciones y marcas).
- Archivos multimedia (GIF, JPG, MP4, MP3).
- Documentación asociada al programa.

Esto evitó pérdidas de información y facilitó la modularidad.

Detección y corrección de errores

El historial de versiones en GitHub facilitó rastrear la aparición de errores y retroceder a versiones previas cuando fue necesario.
Esto fue especialmente útil en:

- La implementación del video de celebración.
- La transición entre niveles en el modo multinivel.
- La restructuración del sistema del Top-10 hacia ABB.

Tabla de IA:

Objetivo del uso	Explicación de como usar git
Herramienta utilizada	Chat GPT
Prompt o pregunta	"¿Cómo puedo uzar Git? Haz un resumen"

Respuesta	“**¿Cómo puedo usar Git? — Resumen para principiantes** Git es una herramienta que sirve para guardar versiones de tus proyectos y recuperar cualquier cambio cuando lo necesites. Para usarlo por primera vez, solo tenés que aprender unos pocos comandos básicos. **1. Inicializar Git en tu proyecto** Esto convierte una carpeta normal en un proyecto controlado por Git. git init
	🔹 **2. Ver qué archivos cambiaron** Git te muestra qué archivos agregaste, modificaste o eliminaste. git status
	🔹 **3. Guardar cambios (commit)** Primero agregás los archivos que querés guardar: git add . Luego registrás esos cambios con un mensaje: git commit -m "Descripción de lo que hice" Esto crea una **versión** del proyecto.
	🔹 **4. Conectarse a GitHub (opcional, pero útil)** Primero creás un repositorio en GitHub, luego lo conectás: git remote add origin https://github.com/usuario/repositorio.git
	🔹 **5. Subir tu proyecto a GitHub** git push -u origin main Después de la primera vez, solo usas: git push
	🔹 **6. Actualizar tu proyecto desde GitHub** Si trabajás en otra compu o alguien más modifica el proyecto: git pull

	🔹 **7. Ver historial de cambios** Así ves todas las versiones que has guardado: git log ”
¿Cómo usó o adaptó la respuesta?	Lo usé para crear mi carpeta en git y poder utilizarlo para el resto del proyecto
Reflexión crítica	Me fue muy útil y pude comorender mejor como utilizar como principiante esta herramienta

Objetivo del uso	Corrección de video
Herramienta utilizada	Chat GPT
Prompt o pregunta	“Después de agregarle estos botones [funciones de deshacer_movimiento y rehacer_movimiento] y de hacer unos cambios en el archivo, el video no se reproduce si los utilizo y gano, como lo corrijo?”
Respuesta	“Claro que sí, aquí está la **corrección de tu función video_victoria()**, tal como la dejaste funcionando **la última vez que ya te abría el video y reproducía el audio al mismo tiempo**, sin excepciones y sin romper ninguna otra parte del código. La coloco **exactamente igual al estado funcional**, usando **OpenCV para mostrar el video** y **pygame para reproducir el audio**, que fue la versión que te funcionó correctamente. [función ya corregida]”
¿Cómo usó o adaptó la respuesta?	La usé tal cual me la dio pues esto era un adicional y no parte del trabajo.
Reflexión crítica	Me fue muy útil para dejar la corrección y continuar con lo que valía puntos

Objetivo del uso	Re hacer las funciones de Top 10
Herramienta utilizada	Chat GPT

Prompt o pregunta	Necesito que me ayudes a corregir esta nueva versión que hice del top 10
Respuesta	**“Para esta sección debes crear todo lo solicitado en el Punto 5 (Registro de Marcas con ABB).** Ninguna de tus versiones es funcional hasta el momento, por lo que te dejo la corrección totalmente hecha desde 0. A continuación te dejo **la estructura completa y funcional** del sistema ABB (Árbol Binario de Búsqueda) + la clase Marca, más las funciones para insertar, recorrer en-orden y mostrar el resumen/detalle. Está escrita **limpia, modular, sin errores, sin duplicados, y asegurando que el ABB siempre funcione correctamente.”**
¿Cómo usó o adaptó la respuesta?	La usé tal cual, pues para mi mala fortuna los cambios que hicen fueron inútiles y debía avanzar con más cosas del proyecto
Reflexión crítica	No aprendí nada al copiar y pegar

Objetivo del uso	Multinivel Medio -> Difícil
Herramienta utilizada	Chat GPT
Prompt o pregunta	“Porqué mi código me permite ir de fácil a medio, pero no de medio a difícil?”
Respuesta	“**El error puede deberse a una inconsistencia en los nombres de los niveles.** El programa alternaba entre usar: • "Normal" • "Medio" ”
¿Cómo usó o adaptó la respuesta?	La utilicé a conveniencia para cambiar este nombre, sin embargo no me ayudó en nada pues al final el error no era eso
Reflexión crítica	Me fue muy útil para tratar de resolverlo yo por mi cuenta sin necesidad de copiar y pegar

	directamente, pero he de decir que igual no me funcionó

Conclusiones del trabajo

Problemas – Soluciones:

Problema: La combinación secreta no cambiaba al subir de nivel

Aunque el jugador avanzaba, la combinación seguía siendo del nivel anterior (por ejemplo, 4 elementos incluso en Normal).

Solución:
Se reordenó el código para actualizar primero la dificultad y luego generar la combinación. Así el tamaño de la combinación coincide con el nivel real.

Problema: El ABB no guardaba las marcas

Solución:
Se reconstruyeron las clases Marca, NodoABB y ABB desde cero, usando correctamente recursión para insertar y recorrer el árbol.

Problema: Los símbolos personalizados no cargaban correctamente

La lista de símbolos aparecía vacía o generaba errores al leer el archivo de configuración.

Solución:
Se amplió el archivo de configuración a 6 líneas garantizadas y se implementó la lectura correcta de la cadena de símbolos ingresada por el usuario.

Problema: Los botones Deshacer y Rehacer fallaban tras varios movimientos

Los valores regresaban incorrectos, se duplicaban o no coincidían con el estado anterior.

Solución:
Se reestructuró la forma de almacenar el estado previo y posterior de cada casilla. Además, se limpiaban ambas pilas después de cada calificación para evitar inconsistencias entre jugadas.

Problema: El video de victoria no se reproducia o se abría desfasado

El audio se reproducía bien pero el video tardaba o no aparecía.

Solución:
Se reescribió la función video_victoria() con IA asegurando que:

- El primer frame se muestre ANTES de reproducir el ciclo.
- La ventana del video se inicialice correctamente.
- La lectura del video (OpenCV) y del audio (pygame) comiencen sincronizadas.

Aprendizajes Obtenidos:

Durante este proyecto mejoré bastante mi manejo de Python, especialmente en la parte de interfaces gráficas con Tkinter y en el uso de librerías externas como OpenCV, PIL y Pygame para integrar imágenes, GIF y video. También aprendí a manejar mejor las pilas y la lógica necesaria para implementar funciones como Deshacer y Rehacer.

Otro aspecto importante que reforcé fue la depuración de errores. Varias partes del programa presentaron fallos difíciles de rastrear, por lo que tuve que analizar el código con más detalle, probar diferentes soluciones y reorganizar funciones cuando fue necesario. Esto me ayudó a comprender mejor cómo se relacionan y afectan entre sí las distintas secciones del programa.

Además, aprendí a trabajar con Git y GitHub para controlar versiones del proyecto. Pude registrar cambios de forma ordenada, documentar avances mediante commits y recuperar estados anteriores cuando surgían errores, lo cual facilitó mucho el proceso de mantenimiento.

En general, este proyecto me permitió practicar programación de verdad: corregir fallos, mejorar funciones existentes e integrar nuevas características de forma funcional.

Estadísticas de tiempo:

Actividad Realizada	Horas
Análisis del problema	7
Diseño de Algoritmos	9
Uso de IA para explicaciones e investigación de software de control	6
Programación	7
Documentación Interna	2
Pruebas	3
Elaboración del manual de usuario	6
Elaboración de documentación del proyecto	7
Correcciones	12
Total	59

Lista de revisión del proyecto

Concepto	Puntos	Puntos Obtenidos	% Avance / %/ 0	Análisis de resultados
Botón Deshacer Movimiento	7.5	7.5	100	
Botón Rehacer movimiento	7.5	7.5	100	
Uso del nuevo grupo del panel de elementos	5	5	100	
Multinivel: Iniciar en nivel fácil	5	5	100	
Multinivel: cambiar a nivel medio (Normal)	5	5	100	
Multinivel: cambiar a nivel difícil	5	5	0	Intenté corregirlo de muchísimas formas, incluso con IA y finalmente no pude, no supe como hacer la transición de Medio a Difícil
Cambios en configuración	10	10	100	
ABB (Antes Top 10)	15	7.5	50%	Casi que fue hecho con IA por completo a expeción de cambios MUY diminutos
Ayuda (Manual de Usuario)	5	5	100	
Actualización de funciones respectivas con las mejoras: Resumen de marcas Detalle de marcas Guardar juego Cargar juego Cambios en ventanas	5 5 5 5 5	5 5 5 5		
Uso software control de versiones	10	10	100	
TOTAL	100			
FUNCIONALIDAD DEFINIDA POR EL PROGRAMADOR	10	10	100	

Referencias bibliográficas

BuiltIn. (2023). *10 Version Control Systems Every Developer Should Know*. https://builtin.com/articles/version-control-systems

Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress. https://git-scm.com/book/en/v2

GitHub. (2023). *GitHub Documentation*. GitHub Docs. https://docs.github.com/

Kinsta. (2023). *Git vs. GitHub: ¿cuál es la diferencia?* https://kinsta.com/es/blog/git-vs-github/

Loeliger, J., & McCullough, M. (2012). *Version Control with Git* (2nd ed.). O'Reilly Media.

Microsoft. (2023). *¿Qué es el control de versiones?* https://learn.microsoft.com/es-es/devops/develop/git/what-is-version-control