

**Universidad autónoma de Aguascalientes.**

**Centro de Ciencias Básicas**

**Ingeniería en Sistemas Computacionales**

**Aguascalientes, Ags.**

Departamento de Sistemas Electrónicos

Materia: Estructuras de Datos

Profesora: Dra. Blanca Estrada Rentería

Grado y grupo: 3ºA

Fecha de entrega: Diciembre de 2023

Equipo: 1

Integrantes del Equipo:

Miguel Ángel Batres Luna ID: 350553

Alan Gael Gallardo Jiménez ID: 351914

Luis Ernesto López Cárdenas ID: 350346

Ariel Emilio Parra Martínez ID: 280862

**“Klotski”**

**PROYECTO FINAL**

# Resumen descriptivo

# Temas investigados

**Klotski**

El nombre "Klotski" en sí mismo proviene de la palabra polaca que significa "bloque pequeño", este rompecabezas, consta de una cuadrícula rectangular que presenta un bloque grande, y varios bloques más pequeños, cada uno de los cuales se puede deslizar en una dirección específica para liberar el bloque más grande a la salida, intentando hacerlo en el menor número de movimientos posibles.

El rompecabezas consta de un número predeterminado de bloques, cada uno con un tamaño, forma y colores únicos. Entre los bloques, se encuentran uno grande de 2×2 unidades, cuatro rectángulos medianos de 2×1 unidades y cuatro cuadrados pequeños de 1×1 unidad.

**Compatibilidad de Windows y Sistemas basados en Unix**

Es crucial lograr compatibilidad entre el código de Windows y sistemas basados en Unix compatibles con estándares POSIX como lo seria Linux y Mac Os X. Esto con la finalidad de obtener un código portable que funcione en casi cualquier sistema operativo y no solo el sistema Operativo de Microsoft, Windows.

Esto se logra mediante el uso de directivas del compilador GCC para detectar el sistema operativo y se usan caracteres de escape ANSI de C para la manipulación de caracteres en la terminal, aunque en Windows no está activada directamente la dicha compatibilidad, pero usando la librería windows.h y sus funciones API de sistema, se activa una “consola virtual” para poder soportar estos caracteres ANSI y también se pueden activar la compatibilidad con caracteres Unicode con codificación UTF8, en la consola de Windows, CMD.

Para la entrada del usuario, se optó por la función getch() de conio.h y la función getch() de ncurses.h, adaptada para trabajar como la de conio.h en Windows.

# Bitácora de trabajo

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Integrantes del equipo | | | | Actividades realizadas | tiempo  (días) |
| Miguel Batres | Alan Gallardo | Luis López | Ariel Parra |
|  |  |  | X | Investigación e implementación de compatibilidad de Sistemas Operativos | 1 |
|  |  |  | X | Manejo de archivos | 2 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Conclusiones

# Referencias consultadas

* Gracie. (2023). *The Ultimate Guide To A Klotski Puzzle.* Recuperado de: <https://puzzlemechanics.com/the-ultimate-guide-to-a-klotski-puzzle/>
* Microsoft. (2021). *SetConsoleMode function*. Recuperado de: <https://learn.microsoft.com/en-us/windows/console/setconsolemode>
* Microsoft. (2022). *Sleep function*. Recuperado de: <https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-sleep>
* Microsoft. (2022). *Virtual Terminal Sequences*. Recuperado de: <https://learn.microsoft.com/en-us/windows/console/console-virtual-terminal-sequences>
* Microsoft. (2022.). *Console and Port I/O*. Recuperado de: <https://learn.microsoft.com/en-us/cpp/c-runtime-library/console-and-port-i-o?view=msvc-170&redirectedfrom=MSDN>
* Microsoft. (2023). *Operador condicional ?:*. Recuperado de: <https://learn.microsoft.com/es-es/cpp/cpp/conditional-operator-q?view=msvc-170>
* Nandakumar. (2014). *Write Your Own conio.h*. Recuperado de: <https://www.opensourceforu.com/2014/03/write-conio-h-gnulinux/>
* Petersen, C. (2021). *ANSI Escape Sequences*. Recuperado de: <https://gist.github.com/fnky/458719343aabd01cfb17a3a4f7296797>
* Sandroid. (2004). *Turbo C Entities.* Recuperado de: <https://www.sandroid.org/TurboC/functionlist.html#Entitites>
* Silisteanu, P. (2019). *ansi-escape-codes-windows-posix-terminals-c-programming-examples*. GitHub. Recuperado de: <https://github.com/sol-prog/ansi-escape-codes-windows-posix-terminals-c-programming-examples>
* Spaans, R. (2009). Solving sliding-block puzzles. Recuperado de: <https://www.pvv.ntnu.no/~spaans/spec-cs.pdf>
* The Linux Programming Interface. (2023). *console\_codes*. Recuperado de: <https://man7.org/linux/man-pages/man4/console_codes.4.html>
* unicode.org (2023). *The Unicode Standard, Version 15.1* . Recuperado de: <https://www.unicode.org/Public/UCD/latest/charts/CodeCharts.pdf>
* user2019716. (2011). *How can I use ANSI escape codes for outputting colored text in C and C++?* . Stack Overflow. Recuperado de: <https://stackoverflow.com/questions/7414983/how-can-i-use-ansi-escape-codes-for-outputting-colored-text-in-c-and-c>
* Vidal, P. J. (2002). *UConio*. Recuperado de: <https://web.archive.org/web/20021016233403/http://crazylovetrain.hypermart.net/projects.htm>
* Wen, X., Zhi, L., & Zhen, R. (2015). *Huarong Dao Puzzle Solution (Search)*. Recuperado de: <https://wendy-xiao.github.io/files/huarongdao.pdf>

# Manual de usuario