Bloque 1: programación secuencial.

Práctica 1: MasterMind.

Curso programación.

mauriciodg00@gmail.com

Por: Mauricio de Garay Hernández (con base en la práctica diseñada por Joel Romero Gómez).
Formato IEEE.

Abstract—

En esta práctica realizarán un programa que sea un juego conocido como mastermind, donde el usuario debe de adivinar una contraseña, disponiendo de varias herramientas (historial, trampa, rendir). Practicarán el uso de matrices, cadenas, ciclos, validaciones, programación modular y colaboración por git.

I. Objetivos

- ❏ Practicar ciclos y comparaciones entre arreglos/matrices.
- ❏ Poder manejar cadenas y validaciones.
- ❏ Poder entender el paradigma de programación y diseño modular.
- ❏ Reforzar sus habilidades de colaboración a través de git, ya que la práctica es en parejas.

II. Descripción del sistema.

Se desea un programa que tenga la capacidad de llevar a cabo un juego conocido como "MasterMind". El programa debe de:

1. Desplegar el título del programa, fecha de creación y nombre de creadores.
2. Generar una contraseña que sea una combinación de 4 "colores"/caracteres (de 7 posibles/en total) que el usuario debe de adivinar.
3. Recibir y manejar de manera correcta los caracteres especiales para ver:
    a. H: historial de jugadas.
    b. T: trampa.
    c. S: salir/rendirse.
4. Tener un conteo de turnos/vidas disponibles.

III. Alcances y limitaciones

1. Todos los valores deben de ingresarse a través del teclado a una terminal.
2. Se debe de validar que el usuario no inserte jugadas que ya ingreso previamente.
3. Se debe de validar que la jugada del usuario no tenga caracteres inválidos (es decir, que todo lo que ingrese el usuario sean jugadas válidas/alguna de las opciones especiales). O sea, el usuario no puede ingresar ningún valor que no sea uno de los 7 colores disponibles o los 3 comandos especiales (H, T y S).

4. Se debe de validar que la contraseña no tenga ningún color/carácter repetido.

IV. Requisitos funcionales.

El programa debe de subirse a su rama de github en el apartado Bloque1/Prácticas/Practica2. Aquí, debe de estar la documentación más el código .c que realizaron. La documentación debe de tener: especificación de requerimientos (con entradas, salidas, alcances, limitaciones, diseños de pantalla, etc), diagramas de diseño (**importante** que sean diagramas de diseño modular) y nombres de sus módulos a utilizar/una breve descripción de cada uno.
Al comenzar, el programa debe de desplegar al usuario el nombre del desarrollador y la fecha en que se creó, esperando a que el usuario ingrese <enter>. Una vez que el usuario ingrese <enter>, se limpiará la pantalla.
A continuación, el programa generará una contraseña aleatoria que sea una combinación aleatoria de 4 de los siguientes 7 colores, donde ninguno se puede repetir (utilicen srand, rand y la librería time.h):

- R: rojo
- A: azul
- Y: amarillo (yellow).
- M: morado.
- B: blanco.
- N: negro.
- C: cafe.

Es decir, unos ejemplos de contraseñas que podrían generar aleatoriamente serían:

- RBNC
- RAYM
- MYNC
- CAYB

Entre varias otras combinaciones de esos 7 colores.
Posteriormente, el usuario deberá ingresar sus jugadas, tiene un total de 16 vidas. Una vez que un usuario ingrese una jugada, se le debe decir cuántos colores metió correctamente, y cuántos estaban en su posición correcta. Ejemplo, si la contraseña generada es RAYM:

Vidas restantes: 16.
Dame tu intento: RAMC ← entrada de usuario
Tienes 3 colores correctos, 2 están en su posición correcta.
Presiona <enter> para continuar…

Vidas restantes: 15.
Dame tu intento: RAMY ← entrada de usuario
Tienes 4 colores correctos, 2 están en su posición correcta.
Presiona <enter> para continuar…

Vidas restantes: 14.
Dame tu intento: RAYM ← entrada de usuario
Tienes 4 colores correctos, 4 están en su posición correcta.
Felicidades! Has ganado!!!
Presiona <enter> para continuar…

Además, se deben de contar con 3 opciones especiales: H para ver historial (se debe de validar que esté vacío o no), T para hacer trampa y S para rendirse. Ejemplo si la contraseña generada es RAYM:

Vidas restantes: 16.
Dame tu intento: H ← entrada de usuario
Tu historial está vacío, vuelve a intentarlo.
Dame tu intento: RAMC ← entrada de usuario
Tienes 3 colores correctos, 2 están en su posición correcta.
Presiona <enter> para continuar…

Vidas restantes: 15.
Dame tu intento: RAMY ← entrada de usuario
Tienes 4 colores correctos, 2 están en su posición correcta.
Presiona <enter> para continuar…

Vidas restantes: 14.
Dame tu intento: H ← entrada de usuario
Tu historial de jugadas es:
RAMC
RAMY
Dame tu intento: T
La respuesta es:

RAYM.
Tramposo!!
Dame tu intento: S
Gracias por jugar! Pero no te rindas a la proxima!
Presiona <enter> para continuar…

Se deberá de validar que no se ingrese una jugada que ya se ingresó o caracteres inválidos (es decir, colores que no sean uno de esos 7 o que no sea una de las opciones especiales); además de validarse que no ingrese el mismo carácter dos veces o una jugada con más o menos de 4 colores. Ejemplo, con la misma contraseña RAYM:

Vidas restantes: 16.
Dame tu intento: jk ← entrada de usuario
Entrada inválida, vuelve a intentarlo.
Dame tu intento: RAMK ← entrada de usuario
Entrada inválida, vuelve a intentarlo.
Dame tu intento: RARM ← entrada de usuario
Entrada inválida, vuelve a intentarlo.
Dame tu intento: RAYMC ← entrada de usuario
Entrada inválida, vuelve a intentarlo.
Dame tu intento: RAYM ← entrada de usuario
Entrada inválida, vuelve a intentarlo.
Tienes 4 colores correctos, 4 están en su posición correcta.
Felicidades! Has ganado!!!
Presiona <enter> para continuar…

El programa debe de tratar entradas en minúsculas y mayúsculas de la misma manera (es decir, se vale ingresar “ramy” o “RAMY”, “h” o “H”). Para hacer esto, utilicen las funciones toupper() y/o tolower() de la librería ctype.h.

V . Restricciones de programación

- Deben de utilizar módulos. Uno de los objetivos **principales** es su diseño modular.
- Deben de utilizar cadenas/matrices.
- Se permiten utilizar estructuras (si es que le encuentran un buen uso), siempre y cuando no utilicen un arreglo de estructuras para evitar trabajar con matrices de caracteres/arreglos de strings (eso NO se vale). Otro objetivo importante es que trabajen con matrices.
- No se pueden utilizar variables globales.
- No se puede utilizar fuerza bruta (usen ciclos).
- Utilizar funciones.
- Hacer comentarios por funciones y por módulo como se vio en clase.
- No se puede utilizar recursión.
- Utilicen las librerías que gusten.

VI.- Fecha de entrega

Miércoles 13 de enero del 2020, a través de github.