Buen día Yanina, ¿todo bien? Considero importante aclarar algunas cuestiones sobre mi trabajo, ya que capaz podrían generar duda.

Convenciones

Las convenciones que utilizo en el trabajo son con base en qué me parece más legible, siempre respetando el uso del **snake_case**.

- Con respecto al código, lo separo en tres partes:

```
...{
declaraciones;

código;

return;
}
```

En caso de no haber una de las partes, hago lo siguiente:

- Si no hay declaraciones:

```
...{

código;

return;
}
```

- Si no hay return:

```
...{
declaraciones;

código;
}
```

- Si solo hay código:

```
...{

código;
}
```

La razón por la que dejo un enter arriba del código pero no abajo es debido a que cuando minimizo el bloque, no me gusta que quede una línea en blanco, ya que si hubiera un enter abajo quedaría como en la izquierda y prefiero el de la derecha:

...{
 	
}
	...{
	}

Añado: Es factible que dentro del 'Código' haya vuelto a separar en secciones dependiendo de lo que está realizando ese bloque de código, como es el caso del for en el main.

Estructuras de control

- Uso del **do while**

Para el ingreso de datos me parecía muy práctico que vuelva a preguntar exactamente lo mismo que siempre, en caso de haber ingresado algo inválido, además del error cometido, y escribir la pregunta afuera del while y dentro del mismo me parecía repetitivo, es por que preferí usar do while en todos los casos.

- Uso del **for**

Estaba entre usar un for y crear una función que represente el trabajo dentro del for y llamarla dos veces, pero sentí que con un for el código sería más entendible.

Extras del código (explicaciones)

- **mayusculizar_variable()**

Creo que si el usuario se confunde e ingresa la letra en minúscula, puede llegar a causarle confusión que el programe le mande error. Es por eso que para validar el ingreso lo convierto en mayúscula.

- **"constantes.h"**

En mi opinión, es más cómodo tener una hoja de constantes y defines aparte, para que así sea más fácil el cambio de las mismas. Esta idea se me ocurrió gracias a un pdf de la cátedra explicando la utilidad del tema.

- **mostrar_datos_ingresados()**

Me parece que es más fácil, tanto para el corrector como para el usuario, tener un resumen de todo lo ingresado, teniendo en cuenta que, en caso que el usuario se haya equivocado muchas veces, el programa pudo haber mostrado muchas palabras y los ingresos haber quedado entre medio de todo eso.

- **Uso de colores**

Decidí usar colores para la ejecución del programa ya que incluso a mí misma me mareaba distinguir entre preguntas, ingresos y errores. Las preguntas las escribí en **magenta**, los

errores en **rojo**, los ingresos en color default, los nombres de GIMLI y LEGOLAS en **azul**, al mostrar los datos ingresados escribí lo pedido en **amarillo** y las condiciones de batalla en **blanco**.

- **Mensajes random de error**

Simplemente porque me parecía más divertido.

- **presentarme()**

Al comienzo le había preguntado al usuario la inicial de su nombre y la había utilizado a lo largo de todo el programa, pero como me informaron que eso debía quitarlo, decidí dejar una simple presentación, más que nada para reutilizar un poco el código que había escrito.

Librerías utilizadas

- #include **<stdio.h>**

Lo básico.

- #include **<stdbool.h>**

Utilizar el tipo de dato booleano.

- #include **<ctype.h>**

Utilizar la función toupper(), para mayusculizar_variable().

- #include **<stdlib.h>**

Utilizar la función rand(), para obtener un número random y usarlo en mostrar_error_letra() y mostrar_error_numero().