Universidad Tecnológica Nacional

Facultad Regional Tucumán

Algoritmos y Estructura de datos

Informe Trabajo Practico Integrador Primer Cuatrimestre

Tema: “Juegos”

**Desarrollo:**

WarCoin:

Para realizar este juego, lo que decidimos hacer fue al enunciado desglosarlo en pequeños requerimientos así se mas fácil para nosotros de entender el problema, buscar soluciones simples y que el código se entienda, por lo que decidimos realizar los siguientes pasos:

1) Se deben ingresar la cantidad mínima y máxima de monedas para retirar.

2) Calculo de la pila de monedas aleatoria entre 10 y 50 monedas, utilizando la función rand() junto con las librerías de “ctime”.

3) Implementación de un sistema el cual elija de manera aleatoria quién comenzará la jugada.

Aquí dependiendo de quién comience tomará un camino u otro, pero en sí, se realiza lo mismo. (Tomaremos como caso que el jugador es el que comienza).

4) El jugador debe retirar una cantidad que esté contemplada con las condiciones anteriormente ingresadas.

5) La computadora retirará una cantidad aleatoria utilizando la función rand(), también deberá estar contemplada con las condiciones anteriormente ingresadas.

De aquí en adelante el ciclo entre 4 y 5 se repite hasta que se de alguna de las condiciones de victoria del juego. En caso de que la computadora comience, el ciclo sería comenzando 5 y luego 4.

Condiciones de victoria:

Existen dos condiciones que pueden llevar a la victoria del jugador o de la computadora:

- El primero es que el jugador pueda sacar la cantidad exacta de monedas que quedan en la pila, impidiendo que la computadora logre sacar.

- La segunda condición que puede llevar a la victoria del jugador es que al momento de sacar monedas la computadora, no lo pueda hacer porque la cantidad mínima de monedas no se lo permita, con un ejemplo se entenderá mejor:

En la pila queda 1 moneda, la cantidad mínima de monedas a sacar es de 2, es el turno de la computadora de sacar monedas, y como no puede sacar 1 moneda porque el mínimo a sacar son 2, por lo tanto, la computadora pierde y el usuario gana.

En cuanto al diseño decidimos mantener el que se muestra como ejemplo en las consignas, ya que nos pareció que muestra de manera clara el sistema del juego. Decidimos también implementar el uso de una librería propia, en la cual se almacenan las funciones que consideramos “menos importantes”, o las cuales estorban a la hora de codificar y entender el código.

En cuanto a casos de prueba pueden hacer que:

* Cantidad mínima para retirar: 2.
* Cantidad máxima para retirar: 5.

De aquí se puede probar retirar 6 monedas, o 1 moneda, y les mostrará un mensaje indicando que no se pueden retirar esa cantidad de monedas y que se vuelva a intentar.

Otro caso interesante para probar es cuando quedan menos monedas en la pila que la cantidad de máxima para retirar, en este caso si sacamos mas monedas de las que hay en la pila, el juego mostrara un error diciendo que no se pueden sacar más monedas de las que hay en la pila.

Ahorcado

Para realizar este juego, antes necesitamos capacitarnos sobre el uso de letras y palabras en c++, ya que en el cursado del primer cuatrimestre fue muy poco lo que nos dijeron sobre este tipo de variables, por no decir nulo, luego de una búsqueda por sitios de internet y videos de YouTube, logramos entender de una forma clara un poco los conceptos de uso para este juego, así que también como en problemas anteriores, buscamos entender el problema de una forma sencilla desglosándolo en pasos simples para poder realizar el juego de una manera más sencilla, que al escribir código lo hagamos de una forma ordenada y que se entienda.

A continuación, se muestra como desglosamos el problema:

1) Ingresar la palabra a descubrir letra por letra y almacenarla en un arreglo.

2) Convertir las letras que se ingresan a mayúsculas.

3) Crear un nuevo vector el cual tenga la cantidad de guiones que tiene la palabra ingresada.

4) Ingreso de letras por parte del 2do jugador, mostrar error en caso de que se ingrese un número.

5) Convertir la letra que ingreso el jugador a mayúsculas

6) Comprobar que la letra este dentro del arreglo que contiene la palabra original.

7) Si la letra se encuentra, remplazar en el arreglo que se muestra la letra.

8) Mostrar las letras que ya se ingresaron, para que así el jugador no las vuelva a repetir.

8) Implementar el sistema de vida del jugador, el cual deberá restar una vida, cada vez que la letra que se ingreso no esté en la palabra original

9) Implementar las partes del cuerpo por medio de texto, el cual ira agregando partes cada vez que se falle al ingresar una letra errónea.

10) Implementar el sistema de puntaje con la fórmula que fue proporcionada en el enunciado del problema.

Condición de victoria: El jugador 2 ganará cuando adivine la palabra, obteniendo un puntaje que dependerá de la cantidad de partes que se mostrar del cuerpo.

En cuanto al diseño decidimos mantener el que se muestra como ejemplo en las consignas, ya que nos pareció que muestra de manera clara el sistema del juego. Decidimos también implementar el uso de una librería propia, en la cual se almacenan las funciones que consideramos “menos importantes”, o las cuales estorban a la hora de codificar y entender el código, como por ejemplo la función que muestra el estado del cuerpo, al usar switch ocupo muchas líneas de código lo cual se nos hizo conveniente llevarlo a una librería para que no estorbe con la lectura del código principal del programa.

En cuanto a casos de prueba pueden hacer que:

Se puede ingresar cualquiera de las palabras propuestas en el enunciado, e ir probando los distintos casos a continuación:

Ingresando la palabra correcta a la primera mostrará el puntaje de 50 puntos.

Otro caso para probar es cuando se ingresa un número, el será notificado como un error, y no descontará la jugada.

Otro caso interesante para probar es el ingresar palabras en vez de letras, lo que hará el programa será solo tomar la primera letra que se ingresó, y descartar todas las demás que se ingresaron.