**PROYECTO 1: AutosLocos**

**Versión 0.1**

El objetivo de este proyecto, es desarrollar un juego en C++ que simula una carrera de carros con ciertas características.

**Descripción general:**

El juego consiste en definir una serie de vehículos que competirán en una carrera simulada, el objetivo es determinar qué vehículo es el ganador, mediante una apuesta. Esta carrera se simulará con símbolos que se moverán en la pantalla.

**Características del juego:**

El juego se ejecuta en la consola del sistema operativo del usuario, al inicio del programa se indica cuántos vehículos van a participar, que **formarán parte de una lista**. Cada vehículo corre en su propio carril, cada carril es una línea horizontal en la pantalla que va desde el extremo izquierdo al derecho.

Los vehículos son representados en la pantalla por un símbolo que puede ser: #,$,%,&,\*. Además cuenta con los siguientes atributos: Velocidad y resistencia.

Una vez que se indica la cantidad de carriles, se colocan las características de cada vehículo.

En un archivo están las configuraciones de los vehículos, estas son:

* **Velocidad:** perezoso, crucero, el super ferrari, Delorean.
* **Resistencia:** es un valor que determina la capacidad de resistir a obstáculos que se pueden encontrar en el camino, los cuales puede ser: bombas, piedras, líquido resbaladizo, cada atributo tiene una valor que determina el nivel de resistencia para cada tipo de obstáculo, pero eso depende de las características del carro como tipos de cauchos y tamaño.
* **Tipos de cauchos:** 
  + Normales
  + Anti coleo (permiten rodar sobre líquidos que pueden ocasionar pérdida de la dirección sino cuenta con esta característica)
  + Todo terreno (tiene mayor resistencia a los obstáculos, esta va en una escala de 1 a 5, mientras mayor el número podrá evitar mejor los obstáculos) pero no tienen propiedad de anti coleo.
* **Tamaño de los cauchos:** 
  + Pegado al piso (son los carros más rápidos, su velocidad puede ser El súper Ferrari, Delorean),
  + Normales (su velocidad puede ser : perezoso , crucero),
  + MonsterTruck (son los más altos pero también son muy lentos, su velocidad es perezoso)
* En cada carril puede aparecer MÁXIMO un obstáculo solamente de cada tipo.
* De acuerdo a las características anteriores, se pueden configurar muchos tipos de vehículos, cada vehículo debe tener un nombre, seleccionado de la siguiente lista:
  + [El Rocomóvil (The Bouldermobile) conducido por los Hermanos Macana, Piedro y Roco (The Slag Brothers, Rock and Gravel)2](https://www.redbull.com/es-es/los-autos-locos-hanna-barbera-productions-pierre-nodoyuna-patan).
  + [El Espantomóvil (The Creepy Coupe) pilotado por Los Tenebrosos (The Gruesome Twosome)](https://www.redbull.com/es-es/los-autos-locos-hanna-barbera-productions-pierre-nodoyuna-patan).
  + [El Auto/Súper Convertible (Convert-a-car) al volante el profesor Locovich (Pat Pending).](https://www.redbull.com/es-es/los-autos-locos-hanna-barbera-productions-pierre-nodoyuna-patan)
  + El Stuka Rakuda (The Crimson Haybailer) conducido por el Barón Hans Fritz (The Red Max).
  + El Compact Pussycat conducido por Penélope Glamour (Penelope Pitstop).
  + El Súper Ferrari (The Army Surplus Special) conducido por el Sargento Blast y el Soldado Meekly.
  + El Antiguomóvil (The Bulletproof Bomb) conducido por los Hermanos Anticuados (The Ant Hill Mob).
  + El Alambique Veloz (The Roaring Plenty) conducido por Lucas el Loco (Lazy Luke) y Osvaldo el Oso (Blubber Bear).
  + El Superheterodino (The Turbo Terrific) conducido por Peter Perfecto (Peter Perfect).
  + El Troncoswagen (The Arkansas Chugabug) conducido por Tío Tomás (Rufus Ruffcut) y Oso Miedoso (Sawtooth).
  + El Super Convertible (The Mean Machine) conducido por Pierre Nodoyuna (Dick Dastardly) y Patán (Muttley).

En esta versión, cada vehículo tiene asignado los atributos por defecto.

Debe considerar una lista de vehículos. Cada vehículo tiene los siguientes atributos:

* Nombre en español
* Nombre el inglés
* Nombre del o los conductores
* Tipo de caucho
* Tamaño de caucho
* Velocidad
* Tiempo de disminución de velocidad

**Factor de resistencia**

El factor de resistencia debe calcularse y representa un valor en mili-segundos, es una reducción de la velocidad del vehículo a cero por cierto tiempo. Este factor depende del tipo de obstáculo y velocidad del vehículo, se calcula multiplicando el obstáculo por velocidad por 100, el resultado está dado en mili-segundos, por ejemplo:

Carro tipo crucero, con tamaño de cauchos normales, tipo de caucho normales, que choca con una bomba se frenará por un factor de resistencia calculado de la siguiente manera:

FR(milisegundos) = 20% \* 120 \* 100 = 1000 mili-segundos (1 seg).

El valor 100 es una constante que se debe colocar siempre.

Tabla de valores según las características del vehículo y valores de los obstáculos

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Velocidad** | **Tipo Caucho** | **Tamaño Caucho** | **Bomba** | **Piedra** | **Liquido** | **Velocidad km/h** |
| Perezoso | Todo Terreno | Monster Truck 1 | 15% | 20% | 30% | 140 |
| Perezoso | Todo Terreno | Monster Truck 2 | 12% | 17% | 25% | 120 |
| Perezoso | Todo Terreno | Monster Truck 3 | 10% | 15% | 20% | 110 |
| Perezoso | Todo Terreno | Monster Truck 4 | 7% | 10% | 15% | 100 |
| Perezoso | Todo Terreno | Monster Truck 5 | 5% | 5% | 10% | 80 |
| Perezoso | Normales | Normales | 15% | 20% | 30% | 120 |
| Crucero | Normales | Normales | 20% | 25% | 35% | 140 |
| El super ferrari | Anti Coleo | Pegado al piso | 25% | 30% | 20% | 160 |
| Delorean | Anti coleo | Pegado al piso | 35% | 40% | 15% | 180 |

NOTA: Los valores deben estar en un archivo de texto plano para que pueda modificarse cambiándolos en el archivo.

**Aplicación de gestión de vehículos.**

Se debe crear una aplicación por separado que permite:

* **Agregar**: permite agregar un vehículo nuevo con la información de los atributos. Al registrar la nueva información se debe indicar al usuario con un mensaje indicando que fue realizado satisfactoriamente el registro y mostrar la información completa del mismo.
* **Modificar**: permite modificar la información de un vehículo que es previamente seleccionado por consola. EL flujo debe ser el siguiente:
  + Se muestra la lista de vehículos con un número de opción que lo identifique.
  + El usuario selecciona la opción del vehículo que desea modificar.
  + Se muestra toda la información del vehículo acompañado con un número de opción que identifica el atributo.
  + El usuario selecciona la opción del atributo que desea modificar.
  + Modificar el valor del atributo.
  + Se realiza la modificación.
  + Se muestra al usuario un mensaje si la modificación se hizo exitosamente o si no se pudo hacer indicando por qué.
  + Si fue exitosa la modificación se muestra toda la información del vehículo.
* **Eliminar**: permite eliminar la información de un vehículo que es previamente seleccionado mediante un número de opción asociado a cada vehículo.
* Una vez eliminado el vehículo, se muestra el mensaje que se eliminó exitosamente y se muestra al usuario el vehículo eliminado junto con la información de los atributos.
* **Lista**: muestra todos los vehículos con sus atributos.
* Consultar: permite consultar la información de un vehículo. Se muestra un listado de vehículos, el usuario selecciona el vehículo que desea consultar y la aplicación muestra toda la información.
* Consultar: permite seleccionar un vehículo mediante un número de opción y muestra los valores de los atributos.
* **Regresar**: regresa siempre a la pantalla anterior.
* **Salir**: sale del sistema.

Cada vez que se agrega, modifica o elimina, debe realizar la acción en la lista de vehículos y en el archivo de texto que mantiene la información.

**Los carriles.**

Una vez que se ha seleccionado la cantidad de carriles, el usuario podrá elegir una de dos opciones:

1. Vehículos aleatorios: asigna un vehículo a cada carril de manera aleatoria, los vehículos pueden repetirse en cuanto a sus configuraciones.
2. Selección de vehículos: permite al usuario seleccionar el vehículo que ocupará cada carril.

La cantidad de carriles los genera el usuario. En cada carril puede aparecer de manera aleatoria: una bomba, una piedra, o líquido resbaladizo, que se muestra en la pista con los caracteres: ¤, ¶, ≠, respectivamente, estos obstáculos afectará al vehículo disminuyendo la velocidad por un tiempo el cual está indicado por los atributos.

Cada carril debe estar identificado del lado izquierdo con el nombre del vehículo.

Si el vehículo no está preparado para el obstáculo perderá velocidad la velocidad depende de los atributos, caso contrario no tendrá ninguna consecuencia, seguirá con la misma velocidad.

Es necesario definir la distancia de la pantalla donde el avance de cada posición representa un kilómetro, la distancia será la cantidad de caracteres que ocupa la pista.

La carrera finaliza cuando todos los vehículos han llegado a la meta, y se muestra al usuario el tiempo que ha tardado cada vehículo en una tabla, indicando el ganador, estos valores son guardados en un archivo de texto.

**Requerimientos:**

**Aplicación de simulación de carreras**

1. Configurar la carrera.
2. Cantidad de carriles.
3. Generación de carriles
4. Selección de vehículos:
5. Comenzar la carrera.
6. Efecto de los obstáculos.
7. Guardar tabla de posiciones por cada carrera. Indicar por vehículo: nombre, tiempo, posición.
8. Opción que muestre la tabla de posiciones de cada vehículo (es una ventana que es mostrada después que el usuario seleccione la opción continuar cuando finaliza la carrera).
9. Regresar en cada pantalla menos la de inicio de carrera.
10. Salir en cada ventana menos la de iniciar la carrera.

**Aplicación de gestión de vehículos**

1. Agregar
2. Modificar
3. Eliminar
4. Consultar
5. Listar

**Grupos de proyecto:**

* El grupo de proyecto debe ser realizado por 3 personas máximo, los mismos deben estar presentes para la defensa del proyecto y se corrige individualmente.

**Características técnicas**.

* El programa debe ser implementado en C++
* Se debe implementar TDA donde sea necesario (cada TDA debe ser implementado con sus operaciones en librerías), no usar arreglos sino listas dinámicas para almacenar la información que se muestra en pantalla.
* Todas las lecturas de los archivos deben almacenarse en listas dinámicas primero antes de utilizarse, cualquier nuevo elemento, modificación o eliminación debe realizarse sobre la lista y después guardar la lista en archivos de texto.
* Como buena práctica de desarrollo, no se debe repetir código.
* Se debe modularizar creando librerías (.h) que se utilicen desde el código.
* Cada función debe hacer una única tarea.
* Usar solamente las librerías que son estándares de C++, no se permite otro tipo de librería.
* El código del proyecto debe estar documentado (estructuras, funciones, librerías).

**Entrega (fecha de entrega: 05/06/2023 al 09/06/2023). Porcentaje: 20%**

* Documentos a entregar:
  + Código fuente del proyecto.
  + Librerías propias necesarias para ejecutar el software.

**Recomendaciones generales:**

* Definir un jefe de proyecto en el grupo que supervise las actividades (por supuesto, este integrante debe desarrollar también).
* Definir las actividades que va a realizar cada integrante, analicen primero, definan tareas a realizar, definan cuáles tareas son prioritarias, definan estructuras y TDA.
* Pensar en desarrollos que sean lo más modular posible, no repetir código: principio DRY (don’t repeat yourself) y KISS (“Keep it Simple, Stupid")
* **Documentos a entregar:**
  + Código fuente del proyecto.
  + Librerías propias necesarias para ejecutar el software.
  + Los archivos deben estar en una carpeta con el nombre: AlgIIProyecto1GrupoX.
  + Debe ser enviado empaquetado en formato zip con el mismo nombre por correo.
  + El asunto del correo debe ser: AlgIIProyecto1GrupoX.
  + En el cuerpo del correo deben colocar los integrantes con su cédula.

EL grupo se definirá en clases, el proyecto se envía el día antes de la fecha de la defensa.

**Recomendaciones generales:**

* Definir un jefe de proyecto en el grupo que supervise las actividades (por supuesto, este integrante debe desarrollar también).
* Definir las actividades que va a realizar cada integrante, analicen primero, definan tareas a realizar, cuáles tareas son prioritarias, estructuras y TDA.
* Pensar en desarrollos que sean lo más modular posible, no repetir código: principio DRY (don’t repeat yourself) y KISS (“Keep it Simple, Stupid")