**Trabajo Práctico #2 : Archivos Binarios y Estructuras de Datos Dinámicas.**

Se requiere de un proceso que informe los datos de los corredores de un rally. Para ello se cuenta con un archivo de datos “**EtapasRally.Dat**” (binario) desordenado conteniendo: a: Número de Etapa (1 díg.), b: Tipo de Vehículo (1 díg.), c: Número de vehículo (2 dígs.), d: Nombre del Piloto (str20), e: Nombre del CoPiloto (str20), f: Marca del vehículo (str15), g: Tiempo en minutos (3 dígs.)**.**

Cantidad de **Etapas cinco (5)**, cantidad de **Tipos de Vehículos cuatro (4)**. El número de vehículo está asociado a un único corredor y solo se repite si corresponde al mismo corredor. Si un vehículo **abandona la carrera, el tiempo en minutos será de novescientos noventa y nueve (999)**, y en las próximas etapas restantes también por cada una de ellas el tiempo será de (999). Si un corredor abandona la carrera en una etapa, las etapas posteriores **NO** se registran en el archivo. Se deberán utilizar **constantes con nombres** para establecer los valores indicados anteriormente. Cada uno de los registros leídos del archivo EtapasRally.Dat **deberán generarse nodos de nro.Vehículos sin repetir y volcados a una estructura de datos interna, pero, NO todos sus campos, sino, los necesarios, entre ellos estará una referencia o posición, la cual permitirá más adelante del proceso volver a acceder a ese registro para obtener los datos que se requieran**. Se deberán determinar en algún momento del proceso las cantidades de **vehículos que abandonaron la carrera**.

Se cuenta con el bloque principal, solo se establecen las invocaciones a los módulos, ud. deberá declarar las variables pertenecientes a este módulo, reemplazar los nombres indicados en los argumentos por nombres de identificadores representativos o con significado para su propósito:

main() {

//declare ud. las variables de esta función y su inicialización, si corresponde.

Abrir (EtapasRally); // Dentro del módulo **ProcEtapasRally**, se deberá invocar

ProcEtapasRally(argumentos…); // al módulo *ExisteNodo* y a *CrearNodo*.

ListadoCorredores(argumentos…); // Dentro del módulo **ListadoCorredores** se deberá

Cerrar (EtapasRally); // invocar al módulo *SacarPrimerNodo*.

**return** 0;

} //main

**A continuación se aportan ideas para realizar el Algoritmo y su codificación:**

1. La función *ProcEtapasRally* deberá leer todos los registros del archivo, para emitir los datos indicados en el punto siguiente, volcar a una estructura de datos interna dinámica, como se mencionó anteriormente, y determinar la cantidad de abandonos.
2. Emitir el listado de los datos leídos del archivo el cual debe salir en forma paralela a los registros leídos del archivo EtapasRally, dentro de la función ProcEtapasRally recordando como se indicó más arriba desordenado. El diseño es:

**Datos de EtapasRally.Dat**

**Etp. NroVehic. Nombre del Piloto Nombre del CoPiloto T.Veh. Marca Vehic. Tiempo mins.**

9 9 X(20) X(20) 9 X(15) 999

1. La función ***void ListadoCorredores(argumentos…);*** deberá emitir el Listado de Corredores de Tiempos por Etapas el cuál debe salir ordenado en forma ascendente por Nro.Vehículo.

**Listado de Corredores de tiempos por etapas ord. x Nro. Vehículo**

**T.Vehíc Nro.Vehic. Nom. Piloto Nom. CoPiloto Marca Vehic. Tiempos en minutos**

**9 99 X(20) X(20) X(15) 999 999 999 999 999**

. . . . . .

. . . . . .

**Cant. de abandonos:** 99

**Nota**: Los tiempos en minutos son cinco, uno por cada etapa.

Los listados serán destinados a un archivo en disco, con el nombre de “**ListadoNroVehicRally.Txt**”, formateado con títulos y con los resultados obtenidos.

**Observaciones, restricciones y recursos disponibles:**

Utilizar las siguientes funciones, invocando en donde sea necesario:

* ***tLista ExisteNodo(tLista &Lista, tLista &pAnt, short nroVehic);***
* ***void CrearNodo(tLista &Lista, tLista &pAnt, sInfo valor);*** 
  + ***void InsertaInicio(tLista &Lista, sInfo valor);***
  + ***void InsertaEnLugar(tLista &pAnt, sInfo valor);***
* ***void SacarPrimerNodo(tLista &Lista, sInfo &valor);***
* **const**

CANT\_ETAPAS = 5,

TIEMPO\_ABANDONO = 999; // en minutos.

* **typedef** **struct** sNodo \* tLista;
* **typedef** **short** tvTmpEtp[CANT\_ETAPAS];

**Espacio en disco**: Solo para generar los archivos de los listados en formato texto.

**Espacio para arreglos**: Lo necesario que requiera este proyecto **dentro de un nodo**.

**Espacio en memoria dinámica**: Para generar nodos a cada uno de los nros. de vehículos sin repetir.

**Accesos a los archivos**: **un solo recorrido secuencial**, para leer a cada uno de ellos **más un acceso al azar** también para leer. Se aclara que se harán accesos a registros del archivo en forma parcial y NO total, ya que alcanza acceder a un registro para cada nro. de vehículo sin repetir. Dicho más claramente, hay varios registros para un mismo nro. de vehículo, solo alcanza hacer una referencia a uno de ellos.

**Bloque Principal**: sólo invocaciones a módulos, según lo establecido anteriormente.

**Optimización**: dado que el uso de ciclos afecta el tiempo de ejecución de un proceso, se evaluará la eficiencia en el uso de los mismos.

Utilizar nombres significativos para los identificadores, dibujos para las estructuras de datos a utilizar, rotulando cada elemento, tamaño, breve leyenda de cómo se generan y estado inicial, respetar esos nombres para utilizarlos en el algoritmo. Preparar una muestra de datos para el archivo de datos e imprimirla, para ejecutar el programa. En el disco solo contendrán los archivos “**TP2V\_K1\_ \_ \_G\_-Apellido Nombre.cpp**”, y los archivos de datos “**EtapasRally.Dat**” (*binario*), y el archivo de texto de la salida de los listados “**ListadoNroVehicRally.Txt**” ubicados en el directorio raíz de la unidad.

* **Cada grupo debe crear su propia muestra de datos para el archivo.**

Se deben utilizar constantes con nombres para indicar estas cantidades.

La muestra de datos para EtapasRally.Dat debe ser: 5 etapas, 4 tipos de vehículos y hasta un máximo de 25 vehículos de cada etapa-tipo, la cual será emitida con títulos apropiados en la entrega adjuntada como documentación, recordando que un mismo corredor puede estar como máximo 5 veces una por cada etapa, items indicados en la sección de archivos.

El Trabajo Práctico deberá ser entregado de acuerdo a las pautas indicadas más abajo, el cual se aprobará si reúne los requerimientos solicitados en **tiempo y forma**:

(**Se debe respetar el orden indicado a continuación**)

1. Entregar en carpeta tamaño A4 de tapa transparente y con sujetador de gancho para las hojas lo siguiente:
2. Carátula con los datos de los integrantes del TP2, entre 2 y 4 alumnos por grupo del mismo curso.
3. Esta misma hoja que establece el enunciado del problema a resolver. Cada grupo elegirá un líder del proyecto, que será el responsable de realizar las entregas del TP. Si un líder abandona la cursada, se deberá elegir otro líder.
4. Diseñar las estructuras de Datos graficándolas indicando con rótulos apropiados, cada elemento, su tamaño en bytes y las variables utilizadas. Las estructuras de datos a graficar son:
   1. *sEtapaRally*, el cual corresponde a cada uno de los registros de datos a leer del archivo de EtapasRally.
   2. *sNodo*, el cual corresponde a la **estructura de datos dinámica** que permite procesar los datos para poder emitir posteriormente los listados.
   3. Dibujar la estructura de datos dinámica de tipo **tLista** con al menos 5 nodos.
5. Graficar el Bloque Principal.
6. Graficar cada uno de los módulos –funciones- a utilizar, cabecera y cuerpo.
7. Construir una muestra de datos, para el archivo de datos, la cual se la utilizará para probar el Algoritmo. En la hoja impresa a entregar debe haber rótulos apropiados, de acuerdo al siguiente diseño:

Se deberá ajustar la escritura de los datos respetando a raja tabla este formato en donde cada columna representa un dato y su tipo de dato y ancho:

9 9 99 X(20) X(20) X(15) 999

Cada columna representa de izquierda a derecha lo siguiente:

**Nro.Etapa Tipo Vehic. Nro. Vehic. Nombre del Piloto Nombre del Copiloto Marca Vehic. Tiempo Mins.**

(1 díg.) (1 díg.) (2 dígs.) (20 caracteres) (20 caracteres) (15 caracteres) (3 dígs.)

Máx. Máx. Máx. Máx. Máx.

1. Emitir según la muestra establecida, los resultados esperados, siempre acompañada de los rótulos apropiados.
2. Codificación del Algoritmo en el Lenguaje C++, emitiendo números de líneas. Usar Code-Blocks. Las primeras líneas serán de comentario indicando: Nombre del programa, fecha entrega, Nro. versión, breve comentario del objetivo del programa, datos del curso, nombre del día, turno, nro. del grupo e integrantes (Apellido, Nombre).

El archivo binario de datos se generará con un programa realizado por el grupo para convertir el archivo de texto del TP1 a binario para que sea procesado por el TP2. Ese otro programa es independiente del TP2 y no deberá ser entregado.

Cada nueva entrega además del nombre indicado para el archivo del código en C++ irá acompañado de la versión entregada, iniciando la primera entrega con el sufijo V1, luego la segunda entrega V2, y así sucesivamente. (S.E. ú O.).