Trabajo Practico 1: Parte 3

Fecha de entrega: Definido Según comisión

Objetivos:

En el trabajo práctico integrador de este año, cada grupo deberá desarrollar una aplicación que permita a usuarios registrados acceder a gestionar un medallero olímpico. Esta aplicación tendrá como objetivo principal registrar y mostrar las medallas obtenidas por los países participantes en diversas disciplinas deportivas, así como también el cálculo de estadísticas asociadas.

En las partes anteriores, cada grupo ha especificado la estructura de menús de la interfaz de usuario, definiendo claramente las opciones para registrar medallas, consultar el medallero y generar reportes, definió y codificó los algoritmos necesarios para la gestión eficiente de la información del medallero. En esta nueva parte, que denominaremos “Parte 3”, se procederá a definir y codificar los algoritmos necesarios para la gestión eficiente de la información del medallero con el uso de archivos.

La aplicación deberá ser implementada utilizando los conceptos desarrollados en la asignatura, tales como:

- Estrategia de descomposición modular descendente.
- Funciones y recursividad.
- Paso de parámetros.
- Tipos de datos abstractos.
- Archivos de texto y directos para el almacenamiento persistente de los datos del medallero.

Además, se espera que los alumnos mantengan el desarrollo de una aplicación amigable, que minimice la posibilidad de ingreso de datos erróneos, con mensajes adecuados que faciliten la experiencia del usuario. La aplicación deberá proporcionar una interfaz de usuario intuitiva y funcional.

El desarrollo de este medallero olímpico no solo reforzará el entendimiento de los conceptos teóricos vistos en clase, sino que también proporcionará una experiencia práctica valiosa en el manejo de datos y la resolución de problemas complejos.

ESTRUCTURAS DE DATOS A UTILIZAR

En base a lo desarrollado en las partes anteriores, en esta nueva iteración se trabajará con un conjunto de requerimientos definidos en base a las estructuras de datos propuestas en este apartado. Los grupos NO PODRÁN agregar nuevas estructuras de datos en sus soluciones.

En este sentido, se propone trabajar con la siguiente estructura:

```
struct archivoCompetencia {
        competidores competencia[87];
        int deporte_medallas[87][3];
        tm fechaCreacion;
        int legajo;
};
```

Particularmente, en el caso del campo fechaCreacion utilizaremos la estructura tm de la librería time.h para obtener fecha y hora del sistema:

```
struct tm{        //representación del tiempo en formato de calendario (fecha/hora)
 int tm_sec;     //Indica los segundos después de un minuto(0 - 59)
 int tm_min;     //Indica los minutos después de una hora (0 - 59)
 int tm_hour;    //Hora [0,23]
 int tm_mday;    //Día del mes[1,31]
 int tm_mon;     //Meses que han pasado desde enero [0,11]
 int tm_year;    //Años desde 1900, si quieres saber el año actual sumas 1900
 int tm_wday;    //Dia de la semana, desde el domingo [0,6]
 int tm_yday;    //Dias desde el 1 de Enero [0,365]
 int tm_isdst;   //No se xDD, algo de daylight
}

time_t time(time_t *)
//devuelve la fecha/hora (time_t) actual o -1 en caso de no ser posible. Si el
//argumento que se le pasa no es NULL, también asigna la fecha/hora actual a dicho
//argumento.

struct tm *localtime(time_t *)
//recibe un puntero a una variable de tiempo (time_t*) y
// devuelve su conversión como fecha/hora LOCAL.
```

A continuación, un ejemplo:

```
#include <time.h>
using namespace std;

int main(){
   time_t ahora;
   struct tm *fecha;
   time(&ahora);
   fecha = localtime(&ahora);
   cout << fecha->tm_mday <<"-"<<  fecha->tm_mon+1 <<"-"<<
fecha-> tm_year+1900;
}
```

I. REQUISITOS FUNCIONALES DE LA APLICACIÓN

Sobre la base del menú desarrollado en la Parte 1, se debe actualizar el contenido para visualizar las siguientes opciones:

MENÚ Principal

```
        Menú Principal
==========================
1.- Generar Competencia
2.- Cargar Medallas por Deporte
3.- Mostrar Medallero
4.- Mostrar Estadísticas
X.- Salir de la aplicación
    Ingrese una opción:
```

En este menú, el funcionamiento solicitado es el siguiente:

- La *opción 1* dirige al usuario al submenú correspondiente.
- La *opción 2* dirige al usuario al submenú correspondiente únicamente si se ha generado una competencia con anterioridad (es decir, se ha ejecutado con éxito la opción 1).
- La *opción 3* dirige al usuario al submenú correspondiente únicamente si se han cargado medallas por deporte (es decir, se ha ejecutado al menos una vez la opción 2).
- La *opción 4* dirige al usuario al submenú correspondiente únicamente si se han cargado medallas por deporte (es decir, se ha ejecutado al menos una vez la opción 2).
- Cuando se sale de la aplicación, confirmado por el usuario, la estructura utilizada durante la ejecución es actualizada en el archivo correspondiente.

Submenú Generar Competencia (Opción 1 del Menú Principal)

Sobre la base del menú desarrollado en la Parte 2, se deben desarrollar las siguientes opciones:

```
        Menú Generar Competencia
=======================================
1.- Generar Competencia
2.- Cargar Competencia
X.- Volver al menú principal

   Ingrese una opción:
```

En este menú, el funcionamiento solicitado es el siguiente:

- *Opción 1*: Debe realizar la carga del arreglo competencia y la inicialización vacía del medallero tal como se planteó en la Parte 2. Como resultado de esta opción, se carga en memoria una variable del tipo archivoCompetencia con toda la información solicitada. Una vez generada la competencia, el contenido de la estructura archivoCompetencia debe ser almacenado en un archivo .bin. El nombre del archivo deberá generarse con el formato de la fecha y hora de creación de la siguiente manera "AAAA-MM-DD-HH-MM-SS.bin". Si se ha ejecutado esta opción, la competencia que se encuentra en la estructura archivoCompetencia que se acaba de generar es la que se utilizará a lo largo de la ejecución del programa. Al igual que en la Parte 2, esta opción solo puede ejecutarse una vez en la ejecución de la aplicación.
- *Opción 2*: Esta opción permite cargar en memoria el contenido de una competencia (junto con su medallero) que ha sido almacenada en un archivo. Para esto, la aplicación solicita al usuario la fecha de creación, busca el archivo .bin correspondiente, y carga en memoria el contenido de la estructura archivoCompetencia que se encuentra almacenada en dicho archivo. Al igual que la Opción 1, esta opción solo puede ejecutarse una vez en la ejecución de la aplicación. Una vez que el proceso de carga del archivo a memoria ha finalizado, debe mostrar el mensaje de que se realizó correctamente la carga de la información mostrando el legajo del usuario que creo dicha información y la fecha en el que se crearon las competencias.

ES IMPORTANTE NOTAR QUE SI NO SE HAN EJECUTADO NINGUNA DE LAS OPCIONES INDICADAS EN ESTE SUBMENÚ, EL RESTO DE LAS OPCIONES DE LA APLICACIÓN NO PODRÁN ESTAR HABILITADAS (YA QUE NO EXISTE UNA COMPETENCIA NI UN MEDALLERO CARGADOS EN MEMORIA).

Submenú Estadísticas

SOLO ACCESIBLE SI SE CARGARON MEDALLAS EN EL MEDALLERO QUE ACTUALMENTE ESTÁ CARGADO EN MEMORIA.

Sobre la base del menú desarrollado en la Parte 1, se debe actualizar el contenido para visualizar las siguientes opciones:

```
Menú Estadísticas
==========================
1.- País con más medallas
2.- Medallas del país
3.- País con más deportes individuales.
4.- Países premiados del deporte.
X.- Volver al menú principal

Ingrese una opción:
```

En este menú, el funcionamiento solicitado es el siguiente:

- *Opción 1*: Se presenta al usuario el nombre del país que tiene más cantidad de medallas, mostrando el nombre del país, y la cantidad de medallas.
- *Opción 2*: Se muestra al usuario el listado de Países, cuando el usuario selecciona uno del listado se muestran la cantidad de medallas obtenidas para el país seleccionado.
- Opción 3: Se muestra al usuario el nombre del país que tiene más cantidad de medallas en la categoría de deportes individuales, mostrando el nombre del país, y la cantidad de medallas.
- Opción 4: Se le solicita al usuario que ingrese una letra con la que comienza el deporte. Se mostrará el podio de todos los deportes que comienzan con la letra ingresada, si tienen cargado el podio, el nombre del deporte y los nombres de los países que han conformado el podio.

PAUTAS DE ENTREGA

CONFORMACIÓN DE GRUPOS

Se deben mantener los grupos de la Parte 2. En caso de que algún grupo deba cambiar su conformación, DEBE CONSULTAR LA SITUACIÓN CON LOS PROFESORES. No se aceptarán entregas de grupos que no respeten esta pauta y no hayan sido debidamente autorizados por los profesores.

EVALUACIÓN

La solución entregada por cada grupo será revisada y evaluada por parte de los profesores, como así también una prueba de funcionamiento. Si la misma cumple con el funcionamiento requerido, el TP se considerará Aprobado, caso contrario podrán solicitarse correcciones específicas y la reentrega del mismo en plazo determinado por la cátedra. El código entregado no solamente debe compilar, sino también ajustarse a las buenas prácticas de codificación aprendidas a lo largo del curso.

Los trabajos no entregados en tiempo y forma, NO SERÁN EVALUADOS y se considerarán NO APROBADOS. Lo mismo ocurrirá con los trabajos entregados por grupos que no respeten la conformación de grupos.

POLÍTICA ANTIPLAGIO

Las entregas realizadas por los grupos son individuales. Los archivos entregados serán analizados con herramientas de software específicas para detectar plagios de código.

Los trabajos que no cumplan con las pautas de autoría establecidas, NO SERÁN EVALUADOS y se considerarán NO APROBADOS.

CONSULTAS

Las consultas de los grupos serán atendidas por los docentes mediante emails, en el Canal virtual empelado por cada comisión y por mensajes en el Campus Virtual.

DOCUMENTACIÓN A ENTREGAR

Las entregas se realizarán exclusivamente por medio del Campus Virtual en la tarea destinada para tal fin por un único integrante del grupo.

Cada grupo debe entregar:

- Archivo “Integrantes.txt”: Archivo de texto que contendrá el apellido, nombre y correo electrónico de cada uno de los integrantes del grupo (uno por línea).
- El programa C++ (archivo/s .cpp), que debe estar comentado, junto con su respectivo archivo ejecutable (archivo .exe).
- Estos archivos deben comprimirse en un único archivo ZIP/RAR nombrado con el siguiente formato *AEDD_GRUPO_apellido1_apellido2_apellido3*.
    - GRUPO: hace referencia a identificación que los docentes le asignaron (Nombre/Número)