UNIVERSIDAD DE MÁLAGA
Departamento de Lenguajes y Ciencias de la Computación

LABORATORIO DE PROGRAMACIÓN

Examen de Septiembre

Curso 2010/2011

01/09/2011

APELLIDOS_____NOMBRE_____

DNI _____ ORDENADOR _____GRUPO (A/B/C)_____

Se han instalado 5 estaciones de seguimiento a lo largo del río Guadalhorce para determinar la calidad de sus nutrientes.
De cada medida realizada se quiere almacenar la siguiente información: código de la estación (numérico), nutriente y unidad (ambos cadenas de caracteres), medida y profundidad (números reales). Como se muestra en la figura adjunta, para almacenar toda la información usaremos un array que almacenará en cada posición una lista con las mediciones realizadas en la estación correspondiente. Se pide implementar un programa llamado `sept11.cpp` que presente el siguiente menú:

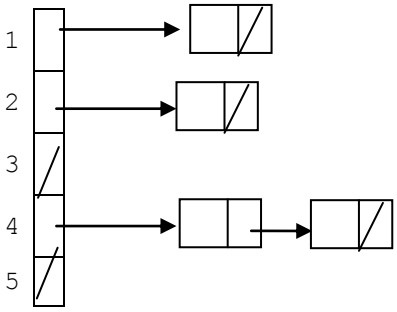

Figura. Esquema de la Estructura `TRio`

```
Nombre: (Apellidos, Nombre)        Curso: 1°
Especialidad: Sistemas             Grupo: A/B/C
Puesto: número de ordenador        Fecha: 01/09/2011

Control Nutrientes
==================
A. Insertar Medida.
B. Cargar Medidas desde Fichero de Texto.
C. Mostrar Todas las Medidas.
D. Mostrar Promedio Profundidad de Nutriente.
E. Salvar Todas las Medidas a Fichero de Texto.
F. Mostrar Medidas Ordenadas por Nutriente.
X. Salir del Programa

Introduzca Opción:
```

A. Leer Medición (1 punto). Se pedirá desde teclado toda la información para insertar una medición en el sistema (código de estación, nutriente, unidades, valor de la medida y profundidad). La inserción en la lista de medidas de la estación debe realizarse ordenada por nutriente y puede haber varias mediciones del mismo nutriente en cada estación. El código de la estación es almacenado también junto al resto de datos de una medida. Si la estación indicada es errónea o no hubiera memoria se informaría del error.

Ejemplo Fichero: medidas.txt

```
1 PH#na#8.27 0
1 Nitrato#mg/l N#0.99 0
1 Oxigeno Saturado#%#103.4 0.5
5 Fosforo#mg/l P#0.37 1
5 PH#na#9.27 3
5 Oxigeno Saturado#%#105.5 1.5
```

B. Cargar Medidas desde Fichero de Texto (1,75 puntos). Se pedirá al usuario el nombre de un fichero de texto y se cargará toda la información en memoria. Se deberá informar de cualquier error relativo al manejo de ficheros o de memoria. Se mantendrán los datos que existan en memoria previos a la carga del fichero, añadiendo los nuevos datos a los ya existentes.
Formato recomendado del Fichero:

```
<COD_ESTACION><SP><NUTRIENTE>#<UNIDADES>#<VALOR_MEDIDA><SP><PROFUNDIDAD><ENTER>
<COD_ESTACION><SP><NUTRIENTE>#<UNIDADES>#<VALOR_MEDIDA><SP><PROFUNDIDAD><ENTER>
<COD_ESTACION><SP><NUTRIENTE>#<UNIDADES>#<VALOR_MEDIDA><SP><PROFUNDIDAD><ENTER>
```

C. Mostrar Todas las Medidas (1 punto). Se mostrará por pantalla un listado de todas las medidas realizadas en cada estación.

D. Mostrar Promedio Profundidad de Nutriente (1,5 puntos). Se pedirá al usuario un nombre de nutriente y se mostrará por pantalla el promedio de profundidad de todas las mediciones realizadas de ese nutriente. En caso de que no existan mediciones de ese nutriente, se mostrará un mensaje advirtiendo de ello.

E. Salvar Todas las Medidas a Fichero de Texto. (1,5 puntos). Se pedirá el nombre de un fichero de texto y se salvará toda la información con el mismo formato del apartado B. Si el fichero no existe se creará y si ya existiera se borrará su contenido.

F. Mostrar Medidas Ordenadas por Nutriente (1,75 puntos). Se mostrará por pantalla un listado de las medidas ordenadas por el nombre del nutriente.

X. Salir del Programa. Se pedirá confirmación de salida.

Módulos a Implementar:(LA DEFINICIÓN DE LOS MÓDULOS NO SE PODRÁ MODIFICAR)

- `MError`: Manejo de errores.
  - Define el tipo `TError` como un enumerado con los posibles errores generados.
  - `MostrarError(err)`: Muestra el error por pantalla.

- `MMedida (MMedida.h, MMedida.cpp)`: Define el tipo `TMedida`. Las funciones que ha de contener son:

```
/* Lee una medida desde teclado */
void LeerMedida(TMedida &m);
/* Escribe una medida por pantalla */
void EscribirMedida(TMedida m);
/* Lee una medida desde fichero */
void LeerMedidaFicheroTXT(ifstream &fich, TMedida &m);
/* Escribe una medida a fichero */
void EscribirMedidaFicheroTXT(ofstream &fich, TMedida m);
```

- `MListaMedidas(MListaMedidas.h, MListaMedidas.cpp)`:Implementación de una lista dinámica de medidas **ordenada** por nutriente. Define el tipo `TListaMedidas`. Funciones a implementar:

```
/* Crea una lista de medidas*/
void CrearLista(TListaMedidas &l);
/* Indica si la lista esta vacia */
bool ListaVacia(TListaMedidas l);
/* Indica si la lista esta llena */
bool ListaLlena(TListaMedidas l);
/* Inserta en la lista la medida proporcionada ordenada por nutriente */
void InsertarLista(TListaMedidas &l, TMedida m, TError &err);
/* Extrae de la lista el elemento cuya posición se indica (el elemento es eliminado de la
lista) */
void SacarMedidaLista(TListaMedidas &l, int posicion, TMedida &m, TError &err);
/* Devuelve la longitud de la lista */
int LongitudLista(TListaMedidas l);
/* Destruye la lista */
void DestruirLista(TListaMedidas &l);
```

- `MRio(MRio.h, MRio.cpp)`. Define el tipo `TRio` que almacena las listas de mediciones realizadas en cada estación. Las funciones que tendrán que implementarse en este módulo son:

```
/*Crea una variable de tipo TRio */
void CrearRio(TRio &rio);
/* Inserta una medida en el sistema */
void InsertarMedidaRio(TRio &rio, TMedida m, TError &err);
/* Muestra la estructura completa por pantalla */
void MostrarRio(TRio &Rio, TError &err);
/* Carga el contenido de un fichero en la estructura */
void CargarMedidasFichero(TRio &Rio, string nomFich, TError &err);
/* Muestra el promedio de profundidad del nutriente indicado */
void MostrarPromedioProfundidad(TRio &Rio,string nutriente,TError &err);
/* Graba el contenido de la estructura en fichero */
void SalvarRioFichero(TRio &Rio, string nomFich, TError &err);
/* Muestra las medidas ordenadas por nutriente */
void MostrarMedidasPorNutriente(TRio &Rio, TError &err);
/* Destruye la estructura */
    void DestruirRio(TRio &Rio);
```

Observaciones:

- Las opciones del menú suman 8,5 puntos. El 1,5 restante se utilizará para valorar la correcta definición de tipos, la modularización y el tratamiento de errores.
- NO SE PODRÁN AÑADIR OPERACIONES EN LA PARTE DE DEFINICIÓN DE UN MÓDULO.
- NO SE PODRÁ hacer uso de detalles de implementación de las listas dentro del programa principal o del módulo MRio, es decir, nada de operaciones del tipo lista->sig=NULL o similares.
- SE PODRÁN implementar todos los procedimientos y funciones auxiliares que se consideren necesarios dentro del programa principal o en la parte de IMPLEMENTACIÓN de los módulos.
- La implementación del módulo MCadena es opcional. Se puede usar directamente el tipo predefinido *string* de la librería *string.h*