PRÁCTICA FINAL
FUNDAMENTOS DE PROGRAMACIÓN

I. Introducción.

La práctica consiste en la programación de software que nos permita la gestión sencilla de una cuadrilla de trabajadores. Para ello el alumno deberá implementar dos clases (Empleado y Cuadrilla) que se especificarán detenidamente más adelante y un programa principal que utilizará ambas clases.

Esta práctica por lo tanto está dividida en tres partes:

1. La implementación de la clase Empleado, que gestionará los datos de un empleado así como la lista de trabajos asignados durante todo el año.
2. La implementación de la clase Cuadrilla, que gestionará una lista de empleados junto con el capataz. Además la clase tendrá métodos para añadir, eliminar y asignar trabajos a los empleados de la cuadrilla.
3. El programa principal que se encargará de gestionar la cuadrilla y realizará consultas sobre la cuadrilla

Para acelerar la codificación por parte del alumno, se le entregará además de la presente documentación:

1. Un proyecto del compilador Dev c++ con el esqueleto de la práctica para que sólo tenga que rellenar el código de los métodos y del programa principal.
2. Dentro del proyecto se han añadido métodos especiales para la lectura y la grabación de los datos en un fichero.
3. Un fichero llamado FileDatos.dat que contiene los datos de una pequeña cuadrilla para que los alumnos no pierdan excesivo tiempo en la entrada manual de datos.

Hay que indicar que una modificación de la estructura y/o del código dado en el proyecto podrá hacer que se produzcan errores inesperados, por eso, solo modificar el código que el alumno haya introducido.

II. Definición de las Clases.

a) Estructura de la clase Empleado.

La clase Empleado, es una clase base que se utiliza para almacenar la información de un solo empleado de la cuadrilla. Dicha información es el **`Nombre`**, **`Dirección`**, **`Banco`** y lista de **`Trabajos`**. Un trabajo es una estructura que contiene los campos **`mes`** y **`dia`** para localizar la jornada de trabajo y **`NHoras`** y **`Precio`** para conocer la remuneración que el empleado obtendrá en el trabajo.

La estructura de dicha clase es la siguiente:

```
typedef char Cadena[100];

struct Trabajo     //Almacena la información de un trabajo
{
   int mes, dia;
   float NHoras;
   float Precio;
};

class Empleado  //Almacena la información de un empleado de la cuadrilla
{
     Cadena Nombre;
     Cadena Direccion;
     Cadena Banco;
     float Sueldo;
     Trabajo Horario[250];  //250 trabajos posibles a lo largo del año
     int NoTrabajos;        //nº de trabajos asignados

   public:
     Empleado();
     void PedirDatos();
     void getNombre(Cadena N);
     void getDireccion(Cadena D);
     void getBanco(Cadena B);
     float CaculaSueldo(int Mes);
     void AniadirTrabajo(Trabajo T);
     Trabajo ConsultarTrabajo(int mes, int dia);
     int operator==(Empleado E);

     void GuardarAFichero(FILE *f);    //No hay que Implementarlo
     void LeerDesdeFichero(FILE *f);   //No hay que Implementarlo

};
Trabajo PedirTrabajo(); //Pide un Trabajo por teclado
```

En esta estructura podemos observar tres cosas importantes:

1. Se utiliza una estructura **`Trabajo`** para almacenar el trabajo realizado por el Empleado.
2. Dos métodos (**`GuardarAFichero`** y **`LeerDesdeFichero`**) que se utilizarán para almacenar y recuperar respectivamente los datos de un Empleado desde un fichero.
3. Una función genérica **`PedirTrabajo`** que devolverá una estructura trabajo con los campos solicitados por teclado.

La descripción de las funciones y los métodos a implementar es la siguiente:

Empleado::Empleado();
Método constructor que inicializa los atributos de la clase a un valor inicial, **0** para los numéricos, **""** para las cadenas.

void Empleado::PedirDatos();
Método que pedirá por teclado el **Nombre**, **Dirección** y el **Banco** del Empleado. Solicitará además la confirmación de si son correctos. Si no lo son volverá a solicitar nuevamente los datos.

void Empleado::getNombre(Cadena N);
void Empleado::getDireccion(Cadena D);
void Empleado::getBanco(Cadena B);
Métodos que devuelven por parámetro respectivamente el **nombre**, la **dirección** y el **Banco** del Empleado.

float Empleado::CalculaSueldo(int Mes);
Método que devuelve el sueldo ganado en el mes especificado por parámetro. Se calcula sumando el trabajo realizado (**NHoras*Precio**) de todos los **días** que se encuentre en el vector **Horario**, del mes especificado.

void Empleado::AniadirTrabajo(Trabajo T);
Método asigna el trabajo **T** pasado por parámetro al vector de Trabajos (**Horario**). Si en el vector ya existe un trabajo en el mismo **d**ía y **mes**, éste se actualiza con los datos del trabajo T, si no se añade el trabajo **T** en la primera posición libre dentro del vector **Horario**.

Trabajo Empleado::ConsultarTrabajo(int mes, int dia);
Método que devuelve el trabajo almacenado en el vector Horario, realizado en el día y mes pasado por parámetro. Si no se encuentra ninguno devuelve un trabajo vacio (Todos los campos con valor 0) que indicará que no se ha localizado ninguno en el horario del empleado.

int Empleado::operator==(Empleado E);
Método que sobrecarga el operador igualdad (==) para comparar dos Empleados. Este método devuelve 1 si el Empleado pasado por parámetro (E) coincide con el nombre del objeto que ejecuta el método, en caso contrario devolverá 0.

b) Estructura de la clase Cuadrilla.

La clase Cuadrilla se utiliza para almacenar un Encargado y 30 Empleados como máximo. Esta clase Cuadrilla tiene operaciones básicas para añadir, consultar y eliminar Empleados así como la de asignar Trabajos. También posee dos métodos adicionales para recuperar la Cuadrilla desde fichero y para almacenar los datos de la Cuadrilla a un fichero.

Para implementar los métodos de esta clase se puede utilizar los métodos y funciones definidos para la clase Empleado.

La estructura de la clase Cuadrilla es la siguiente:

```
class Cuadrilla
{
   Empleado Encargado;
   Empleado Plantilla[30];
   int NEmpleados;
 public:
   Cuadrilla();
   Empleado getEncargado();
   void setEncargado(Empleado P);

   Empleado operator[](int Pos);
   int NoEmpleados() {return NEmpleados;}

   Cuadrilla operator+(Empleado E);
   Cuadrilla operator-(Empleado E);

   Cuadrilla operator+(Trabajo T);

   bool GuardarAFichero(Cadena NombreFichero);
   bool LeerDesdeFichero(Cadena NombreFichero);
};
```

La descripción de los métodos a implementar es la siguiente:

Cuadrilla::Cuadrilla();

Método constructor que debe inicializar los datos de la Cuadrilla. Para ello, simplemente debe inicializar a 0 el nº de empleados ya que cada objeto Empleado almacenado en la clase, ejecutará automáticamente su propio constructor para inicializarse correctamente.

Empleado Cuadrilla::getEncargado();

Método que devuelve el Encargado de la cuadrilla.

void Cuadrilla::setEncargado(Empleado P);

Métodoque asigna el Empleado P como encargado de la Cuadrilla.

Empleado Cuadrilla::operator[](int Pos);

Método que devuelve el empleado que está situado en la posición del vector Plantilla que se especifica por parámetro. Si la posición especificada no es correcta (no existe tal empleado en el vector Plantilla) devolverá un empleado vacío (sin nombre, dirección, banco y ningún trabajo asignado).

Cuadrilla Cuadrilla::operator+(Empleado E);

Método que devolverá una cuadrilla con los datos del objeto cuadrilla que ejecuta el método más el Empleado pasado por parámetro. Si el Empleado ya existe en la cuadrilla actual, este método solo devolverá una copia de la cuadrilla sin ninguna modificación.

Cuadrilla Cuadrilla::operator-(Empleado E);

Método que devolverá una cuadrilla con los datos del objeto cuadrilla que ejecuta el método menos el Empleado pasado por parámetro. Si el Empleado no existe en la cuadrilla actual, este método solo devolverá una copia de la cuadrilla sin ninguna modificación.

Cuadrilla Cuadrilla::operator+(Trabajo T);

Método que devolverá una cuadrilla con los datos del objeto cuadrilla que ejecuta el método pero añadiendo el trabajo pasado por parámetro a cada objeto Empleado de la cuadrilla a devolver.

bool Cuadrilla::GuardarAFichero(Cadena NombreFichero);

Método que almacena los datos de la cuadrilla a un fichero cuyo nombre es especificado en el parámetro NombreFichero, devolverá true si se han guardado correctamente los datos, false en caso contrario.

bool Cuadrilla::LeerDesdeFichero(Cadena NombreFichero);

Método que leen los datos de la cuadrilla almacenados en el fichero especificado en el parámetro NombreFichero, devolverá true si se ha realizado correctamente la lectura de los datos, false en caso contrario.

III. Programa Principal

El programa principal de la práctica antes de presentar el menú solicitará al usuario si desea cargar los datos desde el fichero. Si éste contesta afirmativamente, el fichero será cargado mediante el método LeerDesdeFichero de la clase Cuadrilla. El programa mostrará un mensaje si se han cargado correctamente o no los datos desde fichero según el valor devuelto por el citado método.

También el programa principal, una vez seleccionado la opción de salir del programa, deberá solicitar si el usuario desea almacenar los cambios realizados a la plantilla en el fichero. Si la respuesta es afirmativa, el fichero será actualizado mediante el método GuardarAFichero de la clase Cuadrilla. El programa mostrará un mensaje si se ha guardado correctamente o no los datos al fichero según el valor devuelto pro el citado método.

El menú principal de la práctica consta de 7 opciones incluida la opción de salir. El programa no finalizará hasta que la opción salir sea seleccionada.

```
**** MENU PRINCIPAL ****
**********************
1.- Añadir Encargado
2.- Añadir Empleado
3.- Eliminar un Empleado
4.- Añadir un Trabajo
5- Mostrar la Cuadrilla
6- Buscar un empleado
7.- Salir
    Elige Opción:
```

Opción 1: Esta opción añade el encargado de la Cuadrilla, solicitando previamente un Empleado por teclado, si ya existía modificará su contenido con el nuevo encargado.

Opción 2: Esta opción añadirá un Empleado a la cuadrilla, solicitando previamente el Empleado por teclado, si ya existía no se añadirá.

Opción 3: Esta opción Eliminará el empleado de la cuadrilla que previamente se ha solicitado por teclado, si no existía no realizará ninguna acción.

Opción 4: Esta opción añadirá a todos los empleados de la cuadrilla incluído el encargado el trabajo que previamente se ha solicitado por teclado.

Opción 5: Esta opción muestra el nombre del capataz y de todos los empleados que forman la plantilla. Si la plantilla no tiene Empleados o no tiene Encargado se mostrará el correspondiente mensaje indicando dicha situación anómala.

Opción 6: Esta opción busca el empleado en la cuadrilla que previamente ha solicitado por teclado. El Empleado a buscar puede ser o el Capataz o un Empleado de la cuadrilla. En cualquier caso, una vez localizado el empleado, se mostrarán todos los datos, Nombre, Dirección, Banco, Salario y la lista de todos los trabajos (mes, día, nº horas y precio) del mes actual. El mes actual es solicitado también por teclado una vez verificado que existe el empleado.
En caso de que el empleado no exista se mostrará un mensaje de error.

Opción 7: Esta opción hace que el programa termine.

IV. Entrega de la práctica

La práctica se entregará subiendo el proyecto completo a la página web de la asignatura con fecha tope antes de la realización de la prueba de modificación de la práctica final, que será el último día de clase de laboratorio del cuatrimestre.