Universidad de Córdoba
Escuela Politécnica Superior de Córdoba

ESTRUCTURAS DE DATOS

GRADO EN INGENIERÍA INFORMÁTICA
Segundo curso. Segundo cuatrimestre.

Curso académico 2016 – 2017

SEGUNDA PRÁCTICA

ASIGNATURA COMPUESTA POR UNA LISTA DE ESTUDIANTES

- **Objetivo**
  - Implementar el tipo abstracto de datos **Asignatura** utilizando una lista doblemente enlazada y ordenada de Estudiantes
  - Los estudiantes están ordenados por apellidos y nombre
  - No hay estudiantes repetidos

- **Clases**
  - AsignaturaInterfaz
    - Clase abstracta codificada por el profesor
  - **Asignatura**
    - **Código que debe codificar el estudiante**
  - ListaOrdenadaEstudiantesInterfaz
    - Clase abstracta codificada por el profesor
  - **ListaDoblementeEnlazadaOrdenadaEstudiantes**
    - **Clase que debe codificar el estudiante**
  - NodoEstudianteInterfaz
    - Clase abstracta codificada por el profesor
  - NodoDoblementeEnlazadaEstudiante
    - Clase codificada por el profesorr
  - EstudianteInterfaz
    - Clase codificada por el profesor
  - Estudiante
    - Clase codificada por el profesor

- **Descripción de las clases**
  - Clase **AsignaturaInterfaz**
    - Clase abstracta que especifica los métodos “virtuales puros” para acceder y modificar los atributos de una asignatura
      - Observadores
        - Asignatura vacía
        - Código de la asignatura
        - Nombre asignatura
        - Número de estudiantes

        - Existe un estudiante con nombre y apellidos
        - Acceso
            - a un estudiante con nombre y apellidos
            - a un estudiante i-ésimo
    - Modificadores
        - Código de la asignatura
        - Nombre asignatura
        - Insertar un estudiante de forma ordenada según los apellidos y el nombre
          Borrar estudiante con nombre y apellidos

- Clase **Asignatura**
    - Esta clase hereda de la clase **AsignaturaInterfaz**
    - Atributos
        - Código de la asignatura
        - Nombre de la asignatura
        - Una **Lista de Estudiantes doblemente enlazada y ordenada**
    - Métodos
        - Constructor
            - Parametrizado que recibe valores por defecto para el código y el nombre de la asignatura
        - Debe redefinir los métodos virtuales puros de la clase **AsignaturaInterfaz**
        - Observadores
            - Asignatura vacía
            - Código de la asignatura
            - Nombre de la asignatura
            - Número de estudiantes
            - Existe estudiante con nombre y apellidos
            - Estudiante
                - con nombre y apellidos
                - i-ésimo
        - Modificadores
            - Modificar código de asignatura
            - Modificar nombre de asignatura
            - Insertar un estudiante con nombre, apellidos, nota de teoría y nota de prácticas
            - Borrar estudiante con nombre y apellidos
        - Entrada y salida
            - Cargar una Asignatura desde un fichero de texto
            - Grabar una Asignatura en un fichero de texto

- Clase **ListaEstudiantesInterfaz**
    - Clase abstracta que especifica los métodos “virtuales puros” de una lista de estudiantes
        - Consulta
            - Comprobar si la lista está vacía
            - Consultar el número de estudiantes
            - Consultar el campo informativo

        - actual
        - previo
        - siguiente
      - Comprobar si el cursor es
        - el primer ítem o estudiante
        - el último ítem o estudiante
    - Modificadores
      - Buscar (find) un estudiante: el cursor se coloca en el nodo que contiene al estudiante
      - Ir (se modifica el cursor)
        - a la cabeza
        - al último
        - al anterior
        - al siguiente
      - Modificar el estudiante actual, sino cambia su clave.
      - Insertar un estudiante de forma ordenada por apellidos y nombre
      - Borrar el estudiante actual

- Clase **ListaEstudiantes**
  - Descripción
    - Lista ordenada y doblemente enlazada de elementos de la clase Estudiante
    - Los estudiantes están ordenados por apellidos y nombre y se supone que no hay estudiantes con el mismo nombre y apellidos.
    - Esta clase hereda de forma pública de la clase **ListaEstudiantesInterfaz**
  - Atributos
    - Cabeza: puntero a un nodo del tipo **NodoDoblementeEnlazadoEstudiante**
    - Actual: puntero a un nodo del tipo **NodoDoblementeEnlazadoEstudiante**
  - Métodos privados
    - Consulta de Cabeza y Actual
    - Modificadores de Cabeza y Actual
  - Métodos públicos
    - Constructor sin argumentos
    - Destructor
    - Consulta
      - Comprobar si la lista está vacía
      - Consultar el número de estudiantes
      - Consultar el campo informativo
        - actual
        - previo
        - siguiente
      - Comprobar si el cursor es
        - el primer ítem o estudiante
        - el último ítem o estudiante
    - Modificadores
      - Buscar (find) un estudiante: el cursor se coloca en el nodo que contiene al estudiante
      - Ir (se modifica el cursor)

        - a la cabeza
        - al último
        - al anterior
        - al siguiente
      - Modificar el estudiante actual, sino cambia su clave.
      - Insertar un estudiante de forma ordenada por apellidos y nombre
      - Borrar el estudiante actual
      - Operador de asignación

- Clase **NodoDobleInterfaz**
  - Clase abstracta que especifica los métodos “virtuales puros” de un nodo doblemente enlazado
    - Destructor
    - Consultar el campo informativo
    - Modificar el campo informativo

- Clase **NodoDobleEstudiante**
  - Clase que hereda de forma pública de la clase abstracta **NodoEstudianteInterfaz**
  - Atributos
    - Campo informativo: dato de tipo **Estudiante**
    - Enlace al nodo anterior: puntero del tipo **NodoDobleEstudiante**
    - Enlace al nodo siguiente: puntero del tipo **NodoDobleEstudiante**
  - Constructor parametrizado
  - Destructor
  - Métodos
    - Consulta
      - El campo informativo
      - El nodo anterior
      - El nodo siguiente
    - Modificadores
      - El campo informativo, sino cambia la clave
      - El enlace al nodo anterior
      - El enlace al nodo posterior

- Clase **EstudianteInterfaz**
  - Clase abstracta que especifica los métodos “virtuales puros” para acceder y modificar los atributos de un estudiante
    - Consulta
      - Nombre
      - Apellidos
      - Teoría
      - Práctica
    - Modificación
      - Nombre
      - Apellidos
      - Teoría
      - Práctica

- Clase **Estudiante**
  - Clase que hereda de forma pública de la clase **EstudianteInterfaz**
  - Atributos
    - Nombre
    - Apellidos
    - Teoría
    - Práctica
  - Constructor
    - Parametrizado con valores por defecto
    - de copia
  - Métodos
    - Observadores
      - Nombre
      - Apellidos
      - Teoría
      - Práctica
      - Además se podrá calcular y consultar la **nota final**, que será calculada como la media aritmética de teoría y prácticas
    - Modificadores
      - Nombre
      - Apellidos
      - Teoría
      - Práctica
    - Operador
      - de asignación "="
      - de igualdad "=="
      - de comparación "<": se comparan dos estudiantes lexicográficamente según sus apellidos y nombres.
    - Funciones de lectura y escritura
      - Leer un Estudiante
      - Escribir un Estudiante
    - Funciones **amigas** para sobrecargar los operadores de flujo ">>" y "<<"
      - **friend** istream &operator>>(istream &stream, **Estudiante** &e);
      - **friend** ostream &operator<<(ostream &stream, **Estudiante** const &e);

- **Descripción de los ficheros**
  - **Introducción**
    - Se proporciona un fichero comprimido denominado "practica-2.zip" con los ficheros que se describen a continuación
    - El estudiante debe completar o revisar el código de los ficheros que se indican en cada caso.

  - **makefile**
    - Facilita la compilación de los ficheros, la generación de la documentación con doxygen y el borrado de ficheros que ya no sean necesarios (*.o, *~, etc.)
    - El estudiante puede mejorar este fichero.

- **Doxyfile**
  - Fichero de configuración para generar la documentación con doxygen
  - El estudiante puede modificar este fichero para mejorar su documentación.

- **principalAsignatura.cpp**
  - Programa principal utilizado como ejemplo para comprobar el funcionamiento de la clase Asignatura
  - Fichero proporcionado por el profesor
  - Utiliza macros de pantalla para mejorar la visualización de la información.
  - El estudiante puede ampliar y mejorar este programa de ejemplo.

- **macros.hpp**
  - Fichero con macros para mejorar la visualización de la información en la pantalla.

- **funcionesAuxiliares.hpp**
  - Incluye los prototipos de funciones auxiliares utilizadas en el programa principal del fichero **principalAsignatura**.cpp
  - Fichero proporcionado por el profesor
  - Importante: el estudiante debe revisar este código
- **funcionesAuxiliares.cpp**
  - Código complementario de las funciones auxiliares utilizadas en el programa principal
  - Importante: el estudiante debe completar este fichero.

- **asignaturaInterfaz.hpp**
  - Definición de la clase abstracta AsignaturaInterfaz
  - Fichero proporcionado por el profesor
  - Importante: el estudiante debe revisar este código
- **asignatura.hpp**
  - Definición de la clase Asignatura que hereda de la clase AsignaturaInterfaz
  - Importante: el estudiante debe completar este código
- **asignatura.cpp**
  - Código auxiliar de la clase Asignatura
  - Importante: el estudiante debe completar este código

- **listaOrdenadaEstudiantesInterfaz.hpp**
  - Definición de la clase abstracta **ListaOrdenadaEstudiantesInterfaz**
  - Fichero proporcionado por el profesor
  - Importante: el estudiante debe revisar este código
- **listaDoblementeEnlazadaOrdenadaEstudiantes.hpp**
  - Definición de la clase **ListaDoblementeEnlazadaOrdenadaEstudiantes**
  - Importante: el estudiante debe completar este código

- **nodoEstudianteInterfaz.hpp**
  - Definición de la clase abstracta **nodoEstudianteInterfaz**
  - Fichero proporcionado por el profesor

        - Importante: el estudiante debe revisar este código
    - **nodoDoblementeEnlazadoEstudiante.hpp**
        - Definición de la clase **NodoDoblementeEnlazadoEstudiante** que hereda de la clase **nodoEstudianteInterfaz**
        - Fichero proporcionado por el profesor
        - Importante: el estudiante debe revisar este código

    - **EstudianteInterfaz.hpp**
        - Definición de la clase abstracta **EstudianteInterfaz**
        - Fichero proporcionado por el profesor
        - Importante: el estudiante debe revisar este código
    - **Estudiante.hpp**
        - Definición de la clase **Estudiante** que hereda de la clase Vertice2DInterfaz
        - Proporcionado por el profesor
        - Importante: el estudiante debe revisar este código
    - **Estudiante.cpp**
        - Código complementario de las funciones de la clase **Estudiante**
        - Proporcionado por el profesor
        - Importante: el estudiante debe revisar este código
    - **Fichero con una asignatura de ejemplo**
        - datos.txt
        - Proporcionado por el profesor

- **Observaciones sobre la entrega de la práctica número 2**
    - Duración de la práctica 2: tres sesiones de dos horas cada una.
    - Plazo máximo de entrega
        - Grupos de los lunes: 27 de marzo de 2017
        - Grupos de los martes: 28 de marzo de 2017
        - Grupo del jueves: 30 de marzo de 2017
    - Se deberá subir un fichero comprimido denominado “practica-2-usuario.zip”, donde “usuario” es el login de cada estudiante.
    - El fichero comprimido contendrá
        - makefile
        - Doxyfile
        - ficheros hpp
        - ficheros cpp
        - ficheros txt de ejemplo

    - Nota:
        - Se debe usar el espacio de nombres de la asignatura: **ed**

- **Evaluación**
    - La calificación de la práctica se basará
        - en la calidad del trabajo realizado
        - y en su defensa **presencial**.
    - Se valorará

- El correcto funcionamiento del programa principal propuesto como ejemplo:
  - Véase el fichero principalAsignatura.cpp
- La ampliación y mejora del menú del programa principal para añadir más opciones.
- La documentación del código con doxygen
- La claridad del código
- El uso de macros de pantalla para mejorar la visualización de la información