Universidad de Córdoba
Escuela Politécnica Superior de Córdoba

ESTRUCTURAS DE DATOS

GRADO EN INGENIERÍA INFORMÁTICA
Segundo curso. Segundo cuatrimestre.

Curso académico 2016 – 2017

TERCERA PRÁCTICA

ASIGNATURA IMPLEMENTADA MEDIANTE UN ÁRBOL BINARIO ENLAZADO Y ORDENADO

- **Objetivo**
  - Implementar el tipo abstracto de datos **Asignatura** utilizando un árbol binario enlazado y ordenado 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**
  - NodoEstudianteInterfaz
    - Clase abstracta codificada por el profesor
  - **NodoEnlazadoArbolBinarioEstudiante**
    - **Código que debe codificar el estudiante**
  - 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
    - Clase codificada por le profesor

  - Clase **Asignatura**
    - El estudiante debe codificar los métodos que “no” son “inline”
    - Esta clase hereda de la clase **AsignaturaInterfaz**
    - Atributos
      - Código de la asignatura
      - Nombre de la asignatura

        - Un atributo de la clase **NodoEnlazadoArbolBinarioEstudiante**
    - 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
            - consultarEstudiantes:
                - Escribe los datos de los estudiantes haciendo del recorrido “en orden” del árbol
            - Cargar una Asignatura desde un fichero de texto
            - Grabar una Asignatura en un fichero de texto

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

- Clase **NodoEnlazadoArbolBinarioEstudiante**
    - El estudiante debe codificar los métodos que “no” son “inline”
    - Clase que hereda de forma pública de la clase abstracta **NodoEstudianteInterfaz**
    - Atributos
        - Campo informativo:
            - Puntero a un dato de tipo Estudiante
        - Enlace al hijo izquierdo:
            - Puntero del tipo NodoEnlazadoArbolBinarioEstudiante
        - Enlace al hijo derecho:
            - Puntero del tipo NodoEnlazadoArbolBinarioEstudiante
        - Enlace al padre:

Puntero del tipo NodoEnlazadoArbolBinarioEstudiante

- Constructores
- Destructor
- Observadores
  - isEmpty
  - getItem
  - hasLeftTree
  - hasRightTree
  - hasParent
  - getLeftTree
  - getRightTree
  - getParent
  - nItems
  - exists
  - getEstudiante: función sobrecargadad
  - getMinor
- Modificadores
  - setItem
  - setLeftTree
  - setRightTree
  - setParent
  - setEstudiante
  - insert
  - remove
- Escritura
  - inorder();
  - Etc.

- Clase **EstudianteInterfaz**
  - Clase abstracta que especifica los métodos "virtuales puros" para acceder y modificar los atributos de un estudiante
  - Clase proporciona por el profesor.

- Clase **Estudiante**
  - Clase que hereda de forma pública de la clase **EstudianteInterfaz**
  - Clase proporcionada por el profesor.

- **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

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

- **NodoEnlazadoArbolBinarioEstudiante.hpp**
  - Definición de la clase **NodoEnlazadoArbolBinarioEstudiante** que hereda de la clase **nodoEstudianteInterfaz**
  - Fichero proporcionado por el profesor

        - Importante: el estudiante debe revisar este código

    - **NodoEnlazadoArbolBinarioEstudiante.cpp**
        - Código auxiliar de la clase NodoEnlazadoArbolBinarioEstudiante
        - **Importante: el estudiante debe completar 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 3**
    - Duración de la práctica 3: tres sesiones de dos horas cada una.
    - Plazo máximo de entrega
        - Grupos de los lunes: 24 de abril de 2017
        - Grupos de los martes: 25 de abril de 2017
        - Grupo del jueves: 28 de abril de 2017
    - Se deberá subir un fichero comprimido denominado “practica-3-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