UNIVERSIDADE DA CORUÑA

DISEÑO DE LENGUAJES DE PROGRAMACIÓN
GRADO DE INGENIERÍA INFORMÁTICA: MENCIÓN DE COMPUTACIÓN
CURSO 2017-18

PRÁCTICA 1:
ANÁLISIS DE LAS CARACTERÍSTICAS DE UN CONJUNTO DE LENGUAJES DE PROGRAMACIÓN A PARTIR DE UN CASO PRÁCTICO

1. OBJETIVO

En esta primera práctica de la asignatura se realizará una aproximación inicial, un tanto informal e indirecta, al estudio de las características de los lenguajes de programación. Para ello, dado un problema de programación y una serie de lenguajes, el estudiante deberá implementarlo en cada uno de ellos, analizando las ventajas, desventajas, limitaciones, etc. que plantea cada uno de dichos lenguajes de cara a la implementación del problema propuesto, sea tanto a nivel de estructuras de datos como de estructuras de control, entrada/salida, etc.

2. EL PROBLEMA

El problema elegido es el de la **implementación de Árboles Binarios de Búsqueda (ABB)**. Como recordará el alumno por asignaturas anteriores, un ABB es un árbol binario que o bien es vacío o bien se cumple que para todo nodo T del árbol:

1. El nodo T tiene asociada una clave (supondremos de entrada que se trata de un entero).
2. Los nodos de su subárbol izquierdo (si existe) contienen valores menores que la clave asociada al nodo T.
3. Los nodos de su subárbol derecho (si existe) contienen valores mayores que la clave asociada al nodo T.

Adviértase que se ha optado por no permitir valores repetidos.

Asimismo, se adjuntan al presente enunciado los siguientes ficheros, los cuales servirán al alumno de referencia para la realización de esta práctica:

- `repre_parentizada.pdf`: explicación de cómo se puede representar un árbol en forma de texto parentizado.
- `abb.pas`: *unit* en Pascal que implementa ABBs. Incluye tanto la implementación recursiva como la iterativa.
- `programa_abb.pas`: programa "principal" en Pascal que ilustra el funcionamiento de la unit anterior permitiendo además representar el árbol en formato parentizado –véase `preordenParentizado()`.

3. REALIZACIÓN DE LA PRÁCTICA

De acuerdo a lo expuesto anteriormente, la práctica consiste en **Adaptar_la implementación aportada del TAD ABB para Pascal (`abb.pas`) a una serie de lenguajes**. Dicha implementación deberá cumplir estos requisitos:

- Debe tratarse de una adaptación de dicho código original, no de una implementación partiendo de cero. Por tanto se han de respetar siempre en lo posible las decisiones de diseño e implementación del código Pascal para el resto de los lenguajes, aún cuando esto no permita aprovechar las bondades y características de éstos. Sólo se permitirán las adaptaciones que, forzadas por las características del nuevo lenguaje, respeten el espíritu del diseño e implementación original. Por ejemplo: Pascal no gestiona automáticamente la memoria dinámica y estamos desarrollando una nueva implementación en un lenguaje *L* que permite tanto la gestión automática como manual, deberemos seguir empleando la gestión manual.

  A la hora de dichas adaptaciones, toda decisión de diseño e implementación resultante que no sea obvia/trivial deberá recogerse en la memoria correspondiente a ese lenguaje (véase más adelante) y ser convenientemente justificada. Si resultase del todo imposible la implementación del problema en el nuevo lenguaje respetando mínimamente el diseño e implementación originales, deberá explicarse también el porqué.

- Se hará empleando cada lenguaje a un nivel básico, es decir, no se permite el empleo de librerías especializadas que aborden el tratamiento de árboles, ni de implementaciones de los mismos ya proporcionados por el lenguaje o el entorno de programación.

- Deberán emplearse buenas prácticas de programación, incluyendo una documentación adecuada del código.

- Siempre que sea posible la implementación se hará en una unit, librería, paquete o equivalente, de forma similar a como se hizo también en el caso de Pascal.

1. En la valoración de la práctica se tendrá especialmente en cuenta la heterogeneidad de los lenguajes de programación empleados. Es requisito mínimo para aprobar que se utilicen al menos tres lenguajes de programación. A modo de orientación, podemos citar, entre muchos otros, los siguientes: Pascal, C, Fortran, Java, Python, C++, C#, Ruby, Visual Basic, Matlab, Go, Perl, PHP, OCaml, Haskell, Erlang, Lisp, Basic, Cobol, Scratch, ensamblador, etc. En caso de que el alumno tenga dudas al respecto de la adecuación de un posible lenguaje, consulte previamente al profesor.

2. De cara a su utilización y fácil corrección, deberá **implementar además un programa principal** que, bien automáticamente, bien mediante un pequeño menú o funcionalidad equivalente, permita:
   a) Insertar un valor en el árbol.
   b) Eliminar un valor del árbol.
   c) Buscar un valor en el árbol, mostrando el subárbol correspondiente por pantalla.
   d) Mostrar el árbol por pantalla en formato parentizado.

3. **Redactar para cada lenguaje un informe** que incluya:
   a) El compilador que se utilizó, su versión y el sistema operativo empleado.
   b) Una breve descripción, a modo de contextualización, de las características generales de dicho lenguaje (p.ej. Paradigma al que pertenece, si permite o no trabajar con memoria dinámica, en tal caso su gestión, etc.).
   c) Un análisis detallado de las bondades, desventajas, limitaciones, carencias y cualquier característica peculiar o llamativa de dicho lenguaje en relación a la implementación de nuestro problema, tanto a nivel de estructuras de datos como de estructuras de control, gestión de memoria, operaciones de entrada/salida, etc.
   d) Las justificaciones respecto a las decisiones de diseño e implementación tomadas así

como respecto a las posibles modificaciones que hubiera que introducir para cada lenguaje.

e) En el caso de aquellas implementaciones que, por tener que adecuarse a la *implementación original*, no pudieron echar mano de alguna característica positiva de su lenguaje, deberemos explicar cómo se implementó, por qué no resulta adecuada dicha implementación en el contexto de este lenguaje, cómo deberíamos haberla implementado aprovechando sus bondades (no es necesario incluir código, sólo explicarlo) y qué ganaríamos con ello.

5. ENTREGA, DEFENSA Y EVALUACIÓN

- Las prácticas son OBLIGATORIAS y se realizarán en grupos de DOS PERSONAS.

- La ENTREGA DE TODAS las prácticas dentro de los plazos acordados y la SUPERACIÓN de la parte práctica de la asignatura (es decir, obteniendo entre todas prácticas al menos el 50% de la puntuación de dicha parte) son requisitos IMPRESCINDIBLES para aprobar la asignatura.

  Por tanto, el retraso, la no entrega o la copia de cualquier práctica impedirán automáticamente la superación de la misma y, consecuentemente, impedirán la superación de la asignatura.

- La nota individual de cada una de las prácticas se mantiene para la oportunidad de JULIO.

- **Forma de entrega:** Las prácticas deberán depositarse en el directorio habilitado por el CeCaFi a tal efecto. **Ambos miembros** del grupo deberán entregar la práctica en sus respectivos directorios. Dentro del directorio de entrega del usuario éste depositará:

  1. Un único **documento PDF** que incluya:
     a) Portada con los datos de los integrantes del grupo (nombre, login y email) y un listado de los lenguajes elegidos.
     b) Los informes elaborados para cada uno de dichos lenguajes.

  2. **Un fichero comprimido por cada lenguaje** empleado, cuyo nombre será el de dicho lenguaje (o lo más parecido posible, p.ej.: cplusplus.tgz) Dicho fichero contendrá un directorio que a su vez contendrá:
     a) El código fuente correspondiente, tanto de la implementación del TAD como del programa principal. De ser posible se incluirá en la cabecera de dichos ficheros los nombres, *logins* y *emails* de los miembros del grupo.
     b) Instrucciones detalladas para su compilación (un *Makefile* sería también válido) y ejecución.

- Esta práctica supone el 30% de la nota de la parte práctica de la asignatura.

- En caso de que para un lenguaje dado no se presente su correspondiente informe (o este sea claramente deficiente), dicho lenguaje no computará de cara a la calificación de la práctica, independientemente de la implementación.

- El retraso o no entrega de la práctica supondrán automáticamente la calificación de “no presentada”, imposibilitando también la superación de la parte práctica y, consecuentemente, la superación de la asignatura.

- De cara a la **valoración** de cada solución entregada se tendrán en cuenta, entre otros, los siguientes aspectos:
  - Cumplimiento de las condiciones especificadas en el enunciado.
  - Corrección de la implementación realizada.
  - Eficacia: que se cumplan las especificaciones del problema, implementando correctamente toda la funcionalidad requerida.
  - Esfuerzo y diversidad. Como ya se ha dicho, no es lo mismo, por ejemplo, elegir lenguajes muy similares entre sí, donde las diferencias son poco más que léxicas, a explorar lenguajes de paradigmas muy diferentes unos de otros. Por ejemplo, si se ha implementado ya el problema en Java y se traslada a C++, es obvio que, dadas las similitudes entre ambos, esta nueva implementación en C++ será menos valorada.
  - Dificultad de la adaptación del diseño e implementación originales al nuevo lenguaje.
  - Uso de buenas prácticas de programación: tales como control de errores, claridad y documentación del código, etc.
  - Claridad y calidad de los contenidos del informe.
  - Dado que la mejora de un idioma extranjero figura entre las competencias de la asignatura, se valorará también el esfuerzo de escribir en inglés el código y los informes de cada lenguaje de programación.

- El profesor podrá requerir que la práctica sea defendida, posteriormente a su entrega, mediante el procedimiento que este indique. Si es el caso, dicha defensa será obligatoria y la nota obtenida será individual para cada miembro del grupo. En caso de no haber defendido la práctica dentro de los plazos estipulados a tal efecto, los alumnos en cuestión obtendrán la calificación de “No Presentado” para dicha práctica, con las consecuencias ya antes citadas.

- Fecha límite de entrega: **viernes 6 de octubre**