Estructuras de Datos y de la Información

Proyecto de Programación

3ª Parte

Por:
Profesores de EDI

Índice general

1. Introducción

Este documento describe los requerimientos y funcionalidad básica para el desarrollo de `la tercera parte` del proyecto de programación de la asignatura *Estructuras de Datos y de la Información* del curso 2018-19.

En este proyecto continuaremos colaborando con el Ayuntamiento de Cáceres en el desarrollo de una aplicación que procese los datos disponibles en el portal *Open Data Cáceres* [1] relativos a *barrios*, *calles* y *árboles* de la ciudad. La aplicación consistirá en una serie de algoritmos que extraerán información a partir de los datos previamente representados en nuestra aplicación.

Nuestro objetivo será desarrollar una aplicación con las siguientes características generales:

- *Eficiencia*: debido a la gran cantidad de datos y al esperado crecimiento de los mismos en el futuro, debemos encontrar la forma de almacenar y procesar los datos de la forma más eficiente posible. Además, deberemos realizar un análisis de la eficiencia y rendimiento de nuestra aplicación para cumplir este requisito.
- *Extensibilidad*: el diseño de la aplicación en cuanto a entidades y estructuras de datos debe ser extensible, para facilitar el desarrollo de futuras versiones de la aplicación y ampliar su funcionalidad.
- *Modularidad*: la organización de las entidades que forman nuestra aplicación debe ser modular, de forma que las modificaciones y ampliaciones de la misma se puedan realizar de la forma más sencilla posible, afectando a la menor cantidad de código.

Para cumplir con los requisitos anteriores se seguirá una metodología de desarrollo basada en el paradigma de *Programación Orientada a Objetos* y se utilizará el lenguaje de programación *C++*.

Este documento se organiza de la siguiente forma. En la sección 2 se discuten de forma general los pasos que debemos seguir para el desarrollo de la aplicación y se ofrecen detalles de implementación. En la sección 3 se enumeran los algoritmos básicos que debemos implementar y en la sección 4 se discuten los mecanismos de análisis y tests que debemos aplicar sobre nuestra aplicación para evaluarla, y asegurar que cumple los requisitos, y lo hace de forma eficiente. Finalmente, la sección 5 establece los criterios de evaluación de la calidad del desarrollo y su calificación, así como las fechas de entrega.

2. Metodología e Implementación

En esta sección se describe la funcionalidad que implementará nuestra aplicación, y se establecen una serie de pasos generales que pueden servir de guía para el desarrollo de la misma.

En el desarrollo, se partirá de una plantilla de código, previamente disponible, a la que el programador debe ajustarse. Esta estructura estará formada por una clase principal en la que podemos incluir los algoritmos a implementar como nuevas operaciones.

2.1. Datos

Los datos ofrecidos por el Ayuntamiento han sido pre-tratados y están a disposición del programador en el Campus Virtual de la asignatura[1].

Los archivos que contienen los datos son de tipo texto, y se enumeran en el Cuadro 1. Los campos de cada archivo son autoexplicativos, es labor del programador extraer el tipo de cada dato para representarlo adecuadamente.

Nombre	**Descripción**
`Barrio.csv`	Archivo que contiene los *barrios* de Cáceres.
`Via.csv`	Archivo que contiene las *calles* de la ciudad.
`Arbol.csv`	Archivo que contiene los *árboles* plantados en las calles de Cáceres.

Cuadro 1: Ficheros de datos para la aplicación.

Algunas de las calles en el archivo aparecen duplicadas, debido a que pertenecen a distintos barrios. Este es un detalle importante a la hora de elegir una clave adecuada para almacenar y ordernar las calles *sin repeticiones* en nuestras estructuras de datos (véase sección 2.3).

2.2. Implementación de las clases *base*

Se mantienen las clases base implementadas para la segunda entrega (Barrio, Vía y Árbol).

[1]Se deben utilizar los datos en los ficheros proporcionados sin modificaciones. Cualquier modificación supondrá que los resultados de los algoritmos no serán los esperados, y como consecuencia se podrán considerar incorrectos.

2.3. Contenedores de datos

La estructura de datos básica que vamos a utilizar en esta tercera entrega es el árbol (si bien se podrán utilizar algunas de las estructuras usadas en la segunda entrega en partes concretas del proyecto). La implementación de estos contenedores es suministrado por los profesores en el Campus Virtual de la asignatura como *Template*, y en ningún caso debe ser modificado.

2.4. Relaciones entre clases

De acuerdo con el párrafo anterior, algunas de las relaciones entre clases serán almacenadas en las estructuras de datos anteriormente indicadas. En la Figura 1 se muestran las distintas relaciones entre las clases base y qué estructuras de datos deben ser utilizadas para implementarlas. Esto implica que, el sistema contendrá un conjunto de barrios (reutilizando la estructura implementada en la anterior entrega). Cada *Barrio* debe contener un puntero con un árbol de las *Vías* que comprende. Y, finalmente, cada *Vía* debe almacenar un puntero a una lista de los árboles plantados en cada vía.

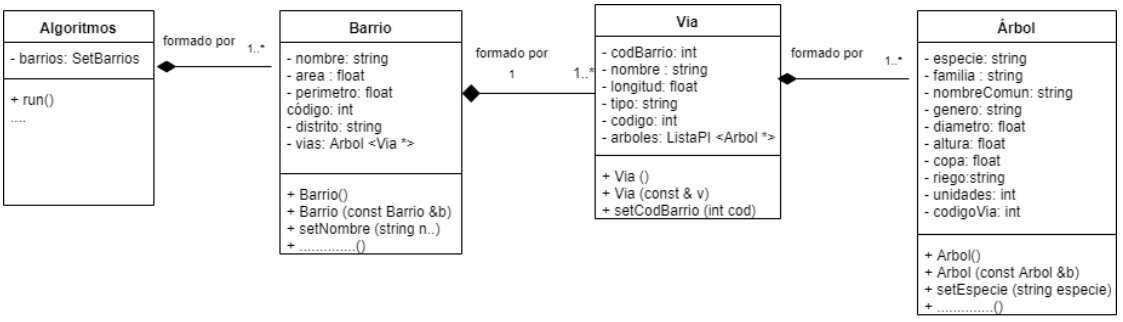

Figura 1: Diseño general del sistema

3. Algoritmos

Los algoritmos a implementar en la tercera entrega del proyecto son:

1. Escribir una operación que genere un fichero con todos los árboles de una determinada vía. La vía se pedirá al usuario. Además, en la documentación se debe comparar la complejidad de esta versión con el mismo algoritmo de la segunda entrega.

2. Escribir una operación que genere un fichero con el nombre de las calles en la que existan árboles de una determinada especie que debe ser aportada por el usuario. En cada línea deberá aparecer el nombre de la vía y el nº de árboles de la especie escogida.
3. Escribir una operación que genere un árbol con todas las vías que tengan árboles de un determinado género aportado por el usuario. Tras devolver el árbol, se debe mostrar por consola la información indicando el barrio donde se encuentra cada vía.
4. Implementar una operación que escriba en un fichero con todas las vías que comiencen por una determinada subcadena de todos los barrios que también comiencen por dichas subcadena, junto con el número de árboles. Solo deben aparecer las vías que tengan árboles.

Todos los algoritmos deben ser realizados sobre las estructuras de datos en memoria, es decir, no están permitidas las implementaciones de los algoritmos directamente sobre los ficheros de datos.

La interfaz de las operaciones debe ser diseñada y documentada correctamente conforme a la descripción.

4. Análisis

La labor de un desarrollador de aplicaciones no es solamente hacer que su código funcione correctamente, sino asegurar que cumple con unos criterios de calidad. Por tanto, el programador tendrá que entregar, además de una versión completamente funcional y completa del código que cumpla con los requisitos expuestos en anteriores secciones, un análisis del mismo, así como algunas propuestas de mejoras futuras y posibles alternativas a la implementación realizada.

La plantilla de documentación a entregar estará disponible en el campus virtual de la asignatura, e incluirá los siguientes puntos:

1. Eficiencia y complejidad: cada algoritmo implementado, y cada operación de un contenedor debe tener asociada una complejidad que depende de la implementación y de la estructura de datos, así como de la estructura de los datos en la aplicación. Se debe proporcionar el análisis de esta complejidad como resultado de la implementación.
2. Alternativas a la implementación actual: se debe analizar alguna alternativa a la implementación y estructura de datos elegida, incluyendo las implicaciones

que tendría en el rendimiento y complejidad dicha alternativa. Además, se debe indicar cómo mejorar el diseño actual con estructuras de datos más avanzadas.

4.1. Calidad del código

Para realizar una correcta comprobación de la calidad del código se deberán realizar tests unitarios y de integración que permitan probar tanto los métodos concretos de cada clase básica implementada (*Barrio*, *Vía* y *Árbol*) como de las relaciones entre clases.

Nota: Las únicas clases que no deben ser probadas son las proporcionadas por los profesores de la asignatura y que implementan las estructuras de datos a utilizar.

5. Evaluación

La evaluación de la implementación tomará en cuenta varios puntos, de los cuales algunos son de obligado cumplimiento, mientras otros son recomendaciones para mejorar la calidad de la implementación. Los siguientes puntos no van en detrimento de lo especificado en el resto del documento.

Obligatorios:

1. Uso de punteros en la implementación, tanto en variables como en estructuras de datos, evitando así los problemas derivados de mover datos en memoria y mantener distintas copias (consistencia).
2. Diseño correcto de interfaces, incluidas la documentación (pre/post-condiciones y descripción) y evaluación de la complejidad de las operaciones.
3. Organización correcta de los datos en memoria de forma que la implementación de los algoritmos cumplan con los requisitos de eficiencia y uso de memoria.
4. Corrección en el resultado de la ejecución de los algoritmos.
5. Utilización correcta de memoria, en cuanto a utilización de la necesaria y reserva/liberación de forma correcta.
6. El lenguaje de programación será C++ y se utilizará el entorno de programación Eclipse. No está permitido utilizar la STL[2] en el desarrollo, ni cualquier otra biblioteca similar.

[2]Standard Template Library

Recomendados:

1. Uso correcto y eficiente de parámetros en la invocación de operaciones.
2. Sobrecarga de operadores para operaciones con objetos, especialmente cuando supongan copias de objetos.
3. Uso de constructores y destructores para las clases de forma coherente y adecuada.
4. Siempre que sea posible, no se duplicarán objetos en la aplicación, y se evitarán copias y duplicados de los mismos.

La evaluación final y calificación del proyecto se llevará a cabo mediante una rúbrica que contendrá los puntos anteriores y que estará a disposición del alumno. Además, las siguientes normas se deben seguir en la realización del proyecto:

- El proyecto debe realizarse en grupos de dos personas pertenecientes a grupos del mismo profesor.
- La entrega del proyecto consistirá en un fichero `zip` cuyo nombre seguirá el formato `ÄpellidosAlumno1_ApellidosAlumno2.zip¨`.
- El fichero debe contener: todo el código fuente generado, los test implementados para probar las funcionalidades y la documentación.
- Cualquier intento de copia, detección de entrega un código que no es propio, o de que uno de los alumnos de la pareja no ha trabajado en el desarrollo, será motivo de suspenso (0) en la asignatura (para los dos alumnos), y puesta en conocimiento de la autoridad del centro.

La fecha límite de entrega del proyecto es el día **9 de junio de 2019**, a través de la correspondiente tarea abierta en el campus virtual de la asignatura. No se tendrán en cuenta entregas realizadas por cualquier otro medio que no sea el establecido, ni con posterioridad a esa fecha. Los alumnos que no entregaron la primera/segunda parte del proyecto de programación en la fecha indicada, u obtuvieron un suspenso en la misma, no pueden entregar esta parte, y tendrán la opción de entregar las tres partes en la convocatoria oficial final de la asignatura.

NOTA: Los alumnos que no hayan superado el proyecto por evaluación continua tienen que entregar tres proyectos independientes (correspondientes a cada una de las entregas), completamente funcionales siguiendo cada una de las especificaciones de las partes, usando la tarea final que estará disponible en el Campus Virtual.

Bibliografía

[1] Ayuntamiento de Cáceres, Open Data Cáceres. Disponible en: http://opendata.caceres.es

[2] B. Stroustrup, *Programming principles and practice using C++*, 2nd ed. Pearson, 2014. Capítulo 19 (sección 3).