

ESTRUCTURAS DE DATOS Y DE LA INFORMACIÓN

PROYECTO DE PROGRAMACIÓN

2ª Parte

Por:
Profesores de EDI



Índice general

1.	Introducción	2
2.	Metodología e Implementación	3
2.1.	Datos	3
2.2.	Implementación de las clases <i>base</i>	3
2.3.	Contenedores de datos	4
2.4.	Relaciones entre clases	4
3.	Algoritmos	5
4.	Análisis	5
4.1.	Calidad del código	6
5.	Evaluación	6

1. Introducción

Este documento describe los requerimientos y funcionalidad básica para el desarrollo de la **segunda 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 asignatura¹.

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 <i>barrios</i> de Cáceres.
Via.csv	Archivo que contiene las <i>calles</i> de la ciudad.
Arbol.csv	Archivo que contiene los <i>árboles</i> 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 las calles *sin repeticiones* en nuestras estructuras de datos (véase sección 2.3).

2.2. Implementación de las clases *base*

Adicionalmente a las clases base implementadas en la primera entrega del proyecto (Via y Barrio) debemos incluir una tercera clase que describa el comportamiento

¹Se 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.

de los *árboles* plantados en Cáceres. La elección de los atributos de cada clase y su representación (tipos de datos) es responsabilidad del programador, y se deben ajustar a los datos en los archivos. Las operaciones a implementar son (1) *setters/getters* para cada atributo, (2) constructores, (3) destructor y una operación para (4) mostrar la información en pantalla. A estas operaciones básicas se podrán añadir otras si es necesario.

2.3. Contenedores de datos

Las estructuras de datos básica que vamos a utilizar en esta segunda entrega son la pila, cola y la lista con puntos de interés. 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 concreto *Ilist*. En la Figura 1 se muestran las distintas relaciones entre las clases base y qué estructura de datos debe ser utilizada para implementarla. 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 una lista de las *Vías* que comprende. Y, finalmente, cada *Vía* debe almacenar un puntero a una lista de los árboles plantados en cada vía.

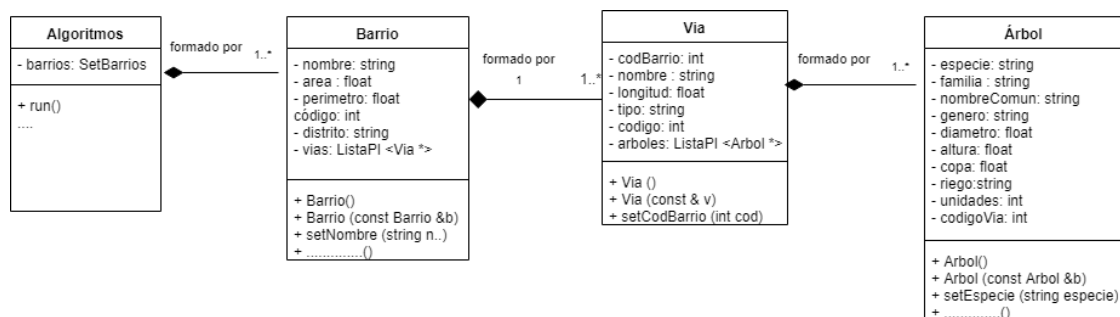


Figura 1: Diseño general del sistema

NOTA: Como se ha comentado anteriormente, algunas de las calles aparecen duplicadas dependiendo del número de barrios al que pertenecen. Para simplificar la

asignación de árboles a vías, se debe asumir que un árbol va a estar almacenado y enlazado solamente por la primera ocurrencia de la vía que sea encontrada.

3. Algoritmos

Los algoritmos a implementar en la segunda 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.
2. Escribir una operación que genere una cola con prioridad con los árboles de un determinado género susceptibles de realizar una poda de aclareo de ramas. La cola con prioridad estará ordenada en base a los siguientes criterios: primero, la altura (siempre y cuando la altura sea mayor que 5m) y, segundo, el tamaño de la copa (se podarán los árboles cuya copa sea mayor que 3m). Tras devolver la cola, se debe mostrar la información indicando el nombre de la calle donde se encuentra cada árbol.
3. Implementar una operación que escriba en un fichero el total de árboles por cada una de las especies.
4. Generar un listado en pantalla que muestre el numero de árboles que tiene cada barrio, con el formato: **nombre del barrio: número de 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. Por ejemplo, en la operación que devuelve la cola con prioridad (algoritmo 2), la operación devolverá como parámetro dicha cola que serán presumiblemente punteros a objetos de la clase `Árbol`.

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 STL² en el desarrollo, ni cualquier otra biblioteca similar.

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 `ApellidosAlumno1_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.

²Standard Template Library

La fecha límite de entrega del proyecto es el día **30 de abril 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 parte del proyecto de programación en la fecha indicada, u obtengan un suspenso en la misma, no pueden entregar esta parte 2, y tendrán la opción de entregar las tres partes en la convocatoria oficial final de la asignatura.

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).