

ESTRUCTURAS DE DATOS Y DE LA INFORMACIÓN

PROYECTO DE PROGRAMACIÓN 2019/20

1ª 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
5.	Evaluación	6

1. Introducción

Este documento describe los requerimientos y funcionalidad básica para el desarrollo de la primera parte del proyecto de programación de la asignatura *Estructuras de Datos y de la Información* del curso 2019-20.

En este proyecto vamos a colaborar 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, vías y ocio de la ciudad.

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, incluido el contenedor de datos que usaremos en esta parte, la *Lista con Punto de Interés*. 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.

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

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 asignatura².

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>vías</i> de la ciudad.
Bar.csv	Archivo que contiene los <i>bares y restaurantes</i> de la ciudad.

Cuadro 1: Ficheros de datos para la aplicación.

Algunas de las vías en el archivo aparecen duplicadas, debido a que pertenecen a distintos barrios. Este es un detalle importante a la hora de almacenarlas en las estructuras adecuadas (véase sección 2.3). Asimismo, la longitud de algunas vías está en blanco, con lo que se puede tomar como 0 m.

2.2. Implementación de las clases *base*

En la plantilla proporcionada como esqueleto de la aplicación, una vez incorporada a un proyecto en el entorno de programación *Eclipse*, incluiremos tres clases que

²Se 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.

contendrán la descripción de cada uno de los tipos de datos en nuestra aplicación: **Barrio**, **Via** y **Bar**. 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, inicialmente en cada una de las clases base, son (1) *setters/getters* para cada atributo, (2) constructores, (3) destructor, (4) operadores sobrecargados necesarios, y (5) una operación para mostrar la información en pantalla. A estas operaciones básicas se podrán añadir otras si es necesario.

2.3. Contenedores de datos

La estructura de datos básica que vamos a utilizar en nuestra aplicación (sección 1) es la *Lista con Punto de Interés*. La implementación de dicho contenedor será proporcionada por los profesores de la asignatura, y se debe usar su interfaz sin modificaciones.

La clase `ListaPI` se utilizará para almacenar todos los datos en la aplicación. Será necesario utilizarla correctamente siguiendo la sintaxis basada en *templates* de C++. Además, las *Precondiciones* y *Postcondiciones* para cada una de las operaciones se deben respetar por parte del programador³.

En el caso en que se desee ordenar los datos en alguna de las listas de la aplicación, el criterio de ordenación elegido debe quedar claramente definido.

2.4. Relaciones entre clases

Todos los datos se almacenarán en contenedores de tipo *ListaPI*. Sin embargo, la estructura interna de clases y contenedores debe ser la siguiente:

1. Los barrios se almacenarán en una `ListaPI`.
2. Cada barrio contendrá una lista con únicamente las vías que pertenecen a ese barrio.
3. Los bares se almacenarán en una lista independiente. La relación con las vías viene establecida por el código de la vía.

³Se recuerda que las pre/post-condiciones son expresiones booleanas, y no texto descriptivo, y que una vez establecidas deben ser respetadas, es decir, el programador que utiliza la interfaz debe cumplir las precondiciones, y el programador que implementa la interfaz debe asegurar las postcondiciones.

El programador debe tener en cuenta el diseño anterior y sus implicaciones en cuanto a (1) la forma en que se leen y cargan los datos desde los ficheros, (2) la forma en que se implementan las operaciones en cada clase y (3) la forma y complejidad de los algoritmos implementados. **Asímismo, el programador debe implementar las estructuras y operaciones en función de los criterios de calidad establecidos en la sección 1.**

3. Algoritmos

Los algoritmos a implementar en nuestra aplicación son:

1. Mostrar las vías de un determinado barrio.
2. Algoritmo que me devuelva qué vía es la más larga.
3. Algoritmo que devuelva una *Cola* con los bares que sirvan un determinado tipo de comida.
4. (OPCIONAL) Algoritmo que me devuelva el *distrito* que tiene más barrios y cuántos barrios tiene.

Todos los algoritmos deben ser implementados 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 descripción⁴.

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:

⁴Por ejemplo, en la operación que devuelve la vía más larga, la operación devolverá un parámetro de salida, que será presumiblemente un puntero a objetos de la clase *Via*.

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 esta información 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.
3. Propuestas futuras y discusión del diseño: se deben proponer mejoras al diseño implementado, como mejorarlo con estructuras de datos más avanzadas, y qué implicaciones tendrían estos cambios.

5. Evaluación

La evaluación de la implementación tomará en cuenta los criterios generales que se especifican a continuación:

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 (*eficiencia*) 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 cumpla 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 STL⁵ en el desarrollo, ni cualquier otra biblioteca similar.
7. Uso correcto y eficiente de parámetros en la invocación de operaciones.
8. Sobrecarga de operadores para operaciones con objetos, especialmente cuando supongan copias de objetos.

⁵Standard Template Library

9. Uso de constructores y destructores para las clases de forma coherente y adecuada, incluido el constructor por copia.
10. 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 de esta primera entrega del proyecto se llevará a cabo mediante una rúbrica (a modo de lista de comprobación) que contendrá varios items basados en los puntos anteriores. Dicha rúbrica estará a disposición del alumno con antelación suficiente. **El alumno debe cumplir con todos los items de la rúbrica para obtener una calificación de APTO en esta primera entrega.**

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 en formato PDF.
- 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 **domingo, 5 de abril**, 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 entreguen la primera parte del proyecto de programación en la fecha indicada, u obtengan un NO APTO en la misma, tendrán la opción de entregar las dos partes en la convocatoria oficial final de la asignatura. Por tanto, no podrán realizar la segunda entrega del proyecto.