Prácticas de Estructuras de Datos
Grado en Ingeniería en Informática
Curso 2022/2023

Práctica 1. Introducción a los contenedores (I)

Sesiones de prácticas: 1

Introducción

Entre las entidades de una aplicación es habitual encontrarse con relaciones de multiplicidad superior a 1 que deben gestionarse mediante alguna estructura de datos que permita almacenar una colección de elementos. A estas colecciones se les suele denominar, de forma genérica, Contenedores, y suelen disponer de funcionalidades similares. En esta práctica, que dividiremos en dos partes, implementaremos y usaremos algunos contenedores básicos para familiarizarnos con sus funcionalidades esperadas así como con algunas técnicas de diseño para facilitar la gestión de sus datos.

Este curso vamos a realizar una aplicación para gestionar Imágenes que sus propietarios quieren compartir con otros. Estas imágenes son etiquetadas automáticamente por un sistema de reconocimiento de imágenes con elementos que se han localizado en ellas. Para ello, trabajaremos con clases de objetos relacionados con esta temática, una de ellas son las Imágenes que publican los usuarios del sistema.

Objetivos

- Implementar una clase que represente un **contenedor básico** que se encargue de almacenar y recuperar elementos de forma segura de un vector de C. El contenedor utilizará **memoria dinámica** para almacenar los datos en el vector de C **y excepciones** para notificar situaciones anómalas, e.g. acceso a una posición no válida. Elaborar un programa de prueba para comprobar su correcto funcionamiento.
- (Opcional) Saber cómo instalar[1] el entorno de desarrollo en C++ CLion en el equipo personal de trabajo.

Descripción de la EEDD

Implementar la clase `ContenedorImagenes` como un contenedor que disponga de algunas funcionalidades básicas para almacenar y recuperar Imágenes. Los métodos a implementar serán los siguientes:

- Constructor por defecto `ContenedorImagenes()`, iniciando el tamaño de posiciones disponibles a 100

[1] En Windows seguir el siguiente tutorial http://tiny.cc/poouja_clion a partir del minuto 7:15. El proceso en Linux y Mac es muy similar.

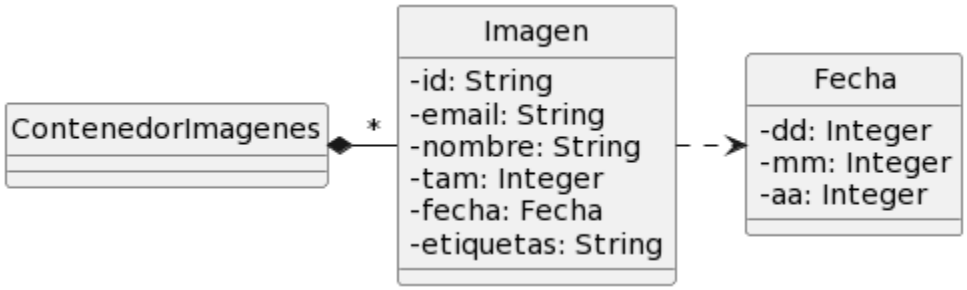
- Constructor dando un tamaño máximo determinado, `ContenedorImagenes (unsigned int tamMax)`.
- Constructor copia `ContenedorImagenes (const ContenedorImagenes & origen)`.
- Constructor de copia parcial `ContenedorImagenes (const ContenedorImagenes & origen, unsigned int posicionInicial, unsigned int numElementos)`. En este constructor hay que tener en cuenta que el vector que se genera debe tener un tamaño máximo de numElementos.
- Operador de asignación (=).
- Asignar una foto a una posición, `ContenedorImagenes::asigna(const Imagen& dato, unsigned int pos)`. Lanzará una excepción `std::out_of_range` si la posición no es válida.
- Recuperar la Imagen de una posición, `Imagen ContenedorImagenes::recupera(unsigned int pos)`. Lanzará una excepción de forma similar a la del apartado anterior si la posición no es válida
- Ordenar el vector de menor a mayor. Se puede utilizar la función sort[2] de <algorithm>: `void ordenar()`.
- Ordenar el vector de mayor a menor con la función `void ordenarRev()`.
- `unsigned int tam()` para obtener el número de posiciones existentes en el ContenedorImagenes.
- El destructor correspondiente.

Programa de prueba: Gestión de un contenedor de Imágenes

En esta primera práctica almacenaremos los datos del fichero adjunto en un contenedor de imágenes. Para ello se deberá leer el fichero adjunto que almacena la información de una Imagen por línea. Los valores de los atributos están puestos en orden según el siguiente UML. El código para la lectura de dicho fichero y la clase Fecha se encuentran adjuntos a la práctica.

La clase Imagen contendrá únicamente los constructores, operadores y getter y setter necesarios para la práctica. El criterio de ordenación de Imagen es por id. Las etiquetas de una imagen son una secuencia de palabras separadas por comas que representan elementos identificados en la misma, e.g. "coche, perro, casa". El propietario de la imagen viene determinado por su email.

[2] Para trabajar con objetos, la función std::sort() requiere que se sobrecargue el operador relacional < de la clase o pasar una función de comparación booleana como tercer parámetro que, dadas dos referencias a objetos, devuelva verdadero si el primer parámetro debe ir antes que el segundo o falso en caso contrario. +info



La prueba implementada en la función main () consistirá en:

- Instanciar un contenedor de imágenes con 10000 posiciones y almacenar en él las imágenes contenidas en el fichero adjunto. Una vez leído el fichero, mostrar el nombre, la fecha y etiquetas de las 50 primeras imágenes del contenedor
- Ordenar el contenedor al revés, es decir, de mayor a menor y mostrar la información de todas sus imágenes
- Ordenar el contenedor de menor a mayor y mostrar los identificadores de las primeras 20 imágenes
- Una vez ordenado el vector, buscar imágenes con algún identificador que se conozca que existe y otro que no, mostrando su posición en el contenedor. Tener en cuenta que al buscar una imagen esta puede no existir.
- Los responsables de la aplicación consideran interesante que los usuarios puedan localizar las imágenes comprendidas entre un periodo de tiempo. Recuperar las 20 primeras imágenes (podría haber menos) del usuario magdalen_upton99@gmail.com durante el 2020 y almacenarlas en un nuevo contenedor de imágenes, mostrando finalmente su contenido por pantalla.
- Aquellos que hagáis las prácticas por parejas tendréis que estudiar los tiempos de los cuatro apartados anteriores. Para el resto es opcional.

Estilo y requerimientos del código:

1. El código debe ser claro, tener un estilo definido y estar perfectamente indentado, para ello se pueden seguir algunos de los estilos preestablecidos para el lenguaje C++ (http://geosoft.no/development/cppstyle.html).
2. Deben comprobarse todas los posibles errores y situaciones de riesgo que puedan ocurrir (desbordamientos de memoria, parámetros con valores no válidos, etc.) y lanzar las excepciones correspondientes, siempre que tenga sentido. Leer el tutorial de excepciones disponible en la pestaña “Material Complementario” en el repositorio de la asignatura en Platea.
3. Se valorará positivamente la calidad general del código: claridad, estilo, ausencia de redundancias, etc.