# Práctica 7. Excepciones, depuración y documentación



## 1. Descripción

Son varios los objetivos a conseguir con esta práctica. En primer lugar, identificar situaciones críticas del sistema que estamos desarrollando y controlarlas mediante el manejo de excepciones. En segundo lugar, hacer uso del depurador GDB para hacer un correcto seguimiento de la ejecución del sistema y detectar fallos lógicos. En tercer lugar, documentar formalmente el sistema con la herramienta Doxygen.

En esta práctica cobra especial importancia la descripción de los casos de uso. Al final de esta práctica deberías tener un sistema software bien depurado, con el manejo de las excepciones allá donde sea necesario, y analizando paso a paso la ejecución del programa. Y todo ello debe quedar reflejado mediante el registro de los casos de uso. Esto es lo que le da solidez a un desarrollo software. Recuerda siempre probar los casos extremos.

Y no menos importante es la documentación formal de un sistema software para que este adquiera un nivel profesional. Recuerda que no hace falta detallar absolutamente todo, pero sí las cabeceras de las clases y las de aquellos métodos más relevantes, así como los atributos más importantes de una clase. También es importante destacar aquellas sutilezas que consideres que incorpora tu código. Toma como ejemplo la documentación de cualquiera de las librerías software vistas en teoría para hacerte una idea del aspecto final que debería tomar la documentación de tu sistema.

# Ejercicio 1. Manejo de excepciones

Recuerda todos los conceptos relativos al manejo de exceptiones que hemos visto en clase e introduce, al menos, uno de cada uno de los siguientes conceptos:

- Clase propia de excepción.
- Bloque simple `try-catch`.
- Bloque `try-catch` con varios manejadores `catch`.
- Instrucción `throw`.
- Relanzamiento de excepción.
- Excepción `bad_alloc` al usar `new`.

# Ejercicio 2. Uso del depurador GDB

Usando el depurador GDB y con la ayuda de los puntos de ruptura o *breakpoints*, realiza el seguimiento de, al menos, los siguientes ítems:

- Alguna estructura de almacenamiento (e.g. un vector) que uses en tu sistema.
- Algún atributo compartido entre clases que pertenezcan a una jerarquía de herencia.
- Algún bloque de repetición (e.g. un while).
- Algún mecanismo de construcción de objeto que sea instancia de clase hija/nieta en una jerarquía de herencia.

Para documentar este seguimiento, realiza capturas de pantalla de la ventana del Terminal, mostrando el contenido de lo que vaya vertiendo el depurador durante la ejecución del sistema. Haz uso de los distintos comandos que ofrece el depurador para mostrar la información que vayas considerando de interés en cada momento.

# Ejercicio 3. Documentación del sistema con Doxygen

Por último, una vez tengas bien depurado tu sistema, deberás documentarlo debidamente para que la herramienta Doxygen genere la documentación final en HTML y LaTeX.

La documentación deberá contener, al menos, lo siguiente:

- Cabeceras de las clases.
- Cabeceras de los métodos más relevantes (e.g. los métodos *getters* y *setters* no haría falta).
- Algunos atributos que consideres de interés.
- Algún bloque de código que consideres importante en el funcionamiento del sistema.

El resultado de esta documentación deberá albergarse en una carpeta denominada `doxygen-doc` en el sistema raíz del programa. Y esta contendrá, a su vez, otras dos carpetas: `html` y `latex`, que contendrán la documentación en estos dos lenguajes.