Un dibujo animado con letras

Descripción generada automáticamente con confianza media Un dibujo de una persona

Descripción generada automáticamente con confianza baja

**Computación Tolerante a Fallas**

***“Otras herramientas para el manejo de errores”***

**Profesor: Michel Emanuel Lopez Franco**

**Sección: D06**

**Horario: Lunes y miércoles 9:00-11:00**

**Fecha de entrega: 29-01-2024**

**Pérez De La Torre Leonardo Octavio | 217429272**

**Desarrollo**

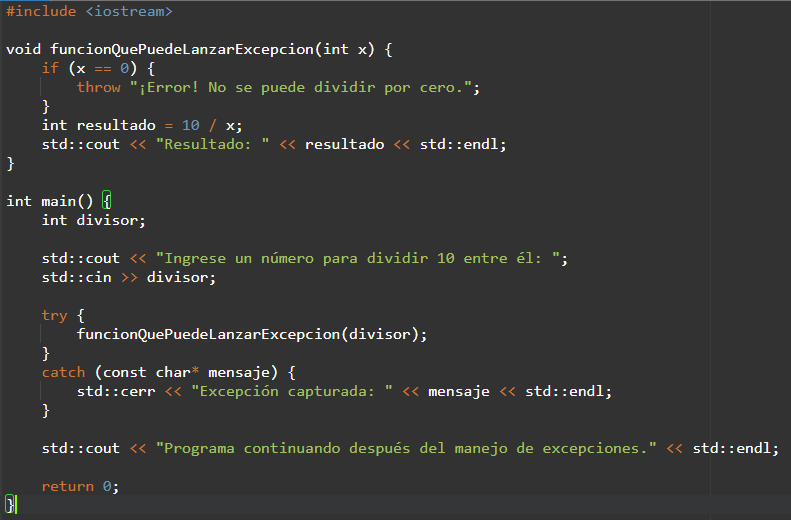
**Bloque Try, Catch y Throw (o su equivalente) en diferentes lenguajes**

Para implementar el control de excepciones en C++, se usan las expresiones **try**, **throw** y **catch**.

En primer lugar, se debe usar un bloque **try** para incluir una o más instrucciones que pueden iniciar una excepción.

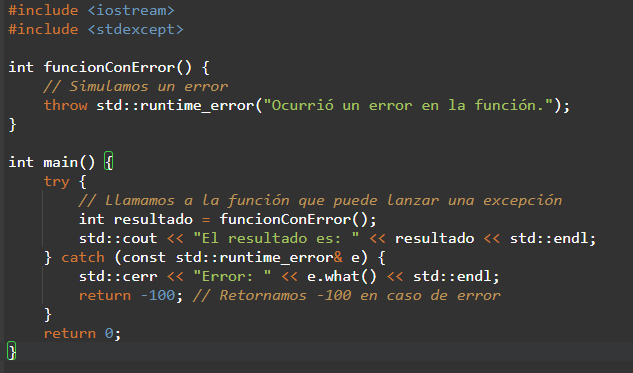
Una expresión **throw** indica que se ha producido una condición excepcional, a menudo un error, en un bloque **try**. Se puede usar un objeto de cualquier tipo como operando de una expresión **throw**. Normalmente, este objeto se emplea para comunicar información sobre el error. En la mayoría de los casos, se recomienda usar la [std::exception](https://learn.microsoft.com/es-es/cpp/standard-library/exception-class?view=msvc-170) clase o una de las clases derivadas definidas en la biblioteca estándar. Si uno de estos no es adecuado, se recomienda derivar su propia clase de excepción de std::exception.

Para controlar las excepciones que se puedan producir, implemente uno o varios bloques **catch** inmediatamente después de un bloque **try**. Cada bloque **catch** especifica el tipo de excepción que puede controlar.



**Retorno de valor para declarar un error**

Otra práctica común para poder determinar un error en una función o sección del código y por lo tanto, poder debuggearlo más fácil, es el retornar un valor en una función en caso de que se produzca una excepción o un fallo.



**Uso de “logs” para tener registro de errores**

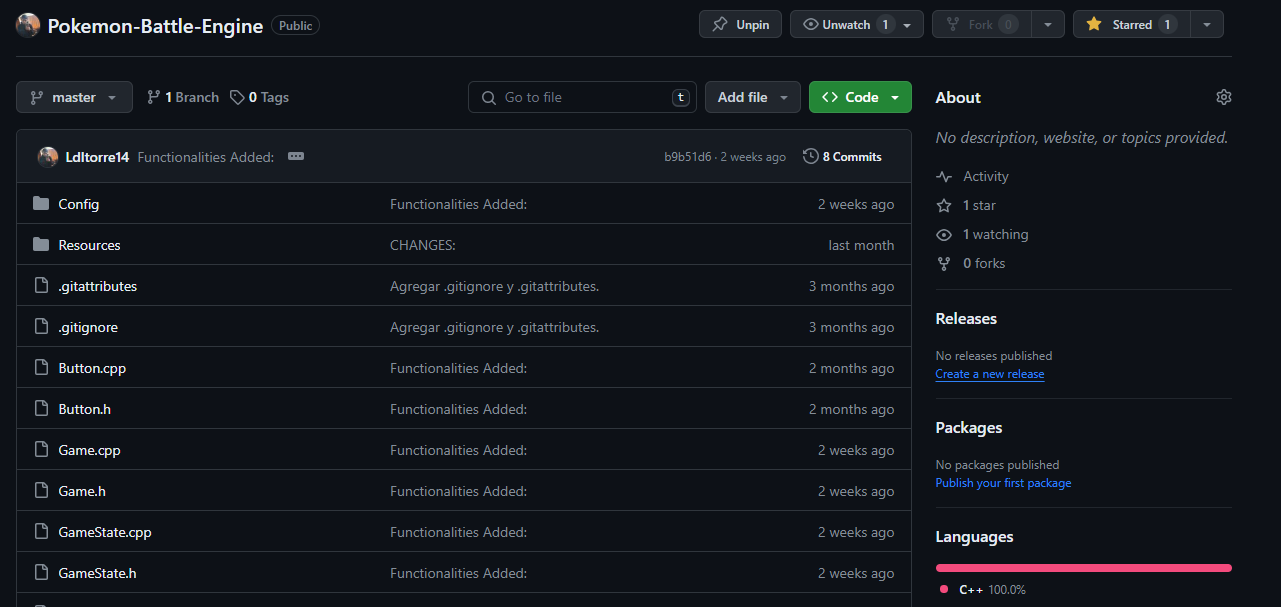
Una práctica común y altamente recomendada, es mantener un archivo normalmente en formato .log, que mantenga un registro de los errores que vaya presentando un software o código según la versión del código o los commits que pueda tener en cada actualización; si bien, esto no corrige el error del todo, en un equipo de desarrollo en una empresa o un software serio, es una buena práctica para poder tener un registro de los fallos y de esta manera poder priorizar la depuración de uno u otro error.

**Software de Control de versiones**

Hacer uso de software de control de versiones como Git/GitHub, es una práctica universalmente adoptada por la industria del desarrollo de software.

Este control de versiones nos permite poder almacenar nuestro código en repositorios y en base a este, poder crear branches o ramas de puntos en los que, en base a una implementación, nuestro código podría funcionar o no funcionar, en general es una buena práctica porque nos da un registro y control absoluto de cada ocasión en la que hemos actualizado nuestro código

(Nota, otra buena práctica en relación a esto, es el utilizar buenos comentarios que puedan describir el commit que hacemos en nuestro código con bastante precisión esto ayuda no solo a nosotros, si no a otras personas que vean nuestro software y deseen contribuir)



**Conclusión**

Conocer los conceptos iniciales y fundamentales para poder introducir a los Sistemas tolerantes a fallas, es un paso fundamental para poder avanzar a conceptos más avanzados y poder aprender sobre la creación de estos mismos sistemas.

**Conclusión**

En todo momento que programamos, nos vamos a encontrar con errores/bugs en nuestro código o incluso en nuestro software, es probablemente el único suceso que esta 100% garantizado cuando programamos, es por eso que, el tener herramientas que nos ayuden a manejar estos errores, son skills bastantes buenas a tomar en cuenta cuando intentamos programar.

**Bibliografía**

* Instrucciones try, throw y catch en C++, (12/10/2023). Microsoft. Consultado en: <https://learn.microsoft.com/es-es/cpp/cpp/try-throw-and-catch-statements-cpp?view=msvc-170>