UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIAS

Materia

Computación Tolerante a Fallas

Alumno:

Estrella Vanessa Palacios Rodríguez

Tratamiento de excepciones

El problema de la seguridad es uno de los clásicos quebraderos de cabeza de la programación. Los diversos lenguajes han tenido siempre que lidiar con el mismo problema: ¿Qué hacer cuando se presenta una circunstancia verdaderamente imprevista? (por ejemplo, un error). El asunto es especialmente importante si se trata de lenguajes para escribir programas de "Misión crítica"; digamos por ejemplo controlar los ordenadores de una central nuclear o de un sistema de control de tráfico aéreo, digamos que en el lenguaje de los programadores C++ estas "circunstancias imprevistas" reciben el nombre de excepciones, por lo que el sistema que implementa C++ para resolver estos problemas recibe el nombre de manejador de excepciones. Así pues, las excepciones son condiciones excepcionales que pueden ocurrir dentro del programa durante su ejecución. Por ejemplo, que ocurra una división por cero, se agote la memoria disponible, Etc. que requieren recursos especiales para su control.

Manejo de excepciones en C++

El manejo de excepciones C++ se basa en un mecanismo cuyo funcionamiento tiene tres etapas básicas:

1: Se intenta ejecutar un bloque de código y se decide qué hacer si se produce una circunstancia excepcional durante su ejecución.

2: Se produce la circunstancia: se "lanza" una excepción (en caso contrario el programa sigue su curso normal).

3: La ejecución del programa es desviada a un sitio específico donde la excepción es "capturada" y se decide que hacer al respecto.

¿Pero qué es eso de "lanzar" y "capturar" una excepción"? En general la frase se usa con un doble sentido: Por un lado, es un mecanismo de salto que transfiere la ejecución desde un punto (que "lanza" la excepción) a otro dispuesto de antemano para tal fin (que "captura" la excepción). A este último se le denomina manejador o "handler" de la excepción. Además del salto -como un goto-, en el punto de lanzamiento de la excepción se crea un objeto, a modo de mensajero, que es capturado por el "handler" (como una función que recibe un argumento). El objeto puede ser cualquiera, pero lo normal es que pertenezca a una clase especial definida al efecto, que contiene la información necesaria para que el receptor sepa qué ha pasado; cual es la naturaleza de la circunstancia excepcional que ha "lanzado" la excepción.

Para las tres etapas anteriores existen tres palabras clave específicas: try, throw y catch. El detalle del proceso es como sigue.

Intento (try):

En síntesis, podemos decir que el programa se prepara para cierta acción, decimos que "lo intenta". Para ello se especifica un bloque de código cuya ejecución se va a intentar ("try-block") utilizando la palabra clave try.

```
try {    // bloque de código-intento
  ...
}
```

El juego consiste en indicar al programa que, si existe un error durante el "intento", entonces debe lanzar una excepción y transferir el control de ejecución al punto donde exista un manejador de excepciones ("handler") que coincida con el tipo lanzado. Si no se produce ninguna excepción, el programa sigue su curso normal.

Se lanza una excepción (throw):

Si se detecta una circunstancia excepcional dentro del bloque-intento, se lanza una excepción mediante la ejecución de una sentencia throw. Por ejemplo:

```
if (condicion) throw "overflow";
```

Es importante advertir que, salvo los casos en que la excepción es lanzada por las propias librerías C++, estas no se lanzan espontáneamente. Es el programador el que debe utilizar una sentencia (generalmente condicional) para, en su caso, lanzar la excepción.

El lenguaje C++ especifica que todas las excepciones deben ser lanzadas desde el interior de un bloque-intento y permite que sean de cualquier tipo. Como se ha apuntado antes, generalmente son un objeto (instancia de una clase) que contiene información. Este objeto es creado y lanzado en el punto de la sentencia throw y capturado donde está la sentencia catch.

La excepción es capturada en un punto específico del programa (catch):

Esta parte del programa se denomina manejador ("handler"); se dice que el "handler" captura la excepción. El handler es un bloque de código diseñado para manejar la excepción precedido por la palabra catch. El lenguaje C++ requiere que exista al menos un manejador inmediatamente después de un bloque try. Es decir, se requiere el siguiente esquema:

```
try {          // bloque de código que se intenta
  ...
}
catch (...) { // bloque manejador de posibles excepciones
  ...
}
...            // continua la ejecución normal
```

El "handler" es el sitio donde continua el programa en caso de que ocurra la circunstancia excepcional (generalmente un error) y donde se decide qué hacer. A este respecto, las estrategias pueden ser muy variadas (no es lo mismo el programa de control de un reactor nuclear que un humilde programa de contabilidad). En último extremo, en caso de errores absolutamente irrecuperables, la opción adoptada suele consistir en mostrar un mensaje explicando el error. Puede incluir el consabido "Avise al proveedor del programa" o bien generar un fichero texto (por ejemplo: error.txt) con la información pertinente, que se guarda en disco con objeto de que pueda ser posteriormente analizado y corregido en sucesivas versiones de la aplicación.

Hemos dicho que try es una sentencia que en cierta forma es capaz de especificar el flujo de ejecución del programa; en el fondo el mecanismo de excepciones de C++ funciona como una especie de sentencia if ... then .... else, que tendría la forma:

If { este bloque se ejecuta correctamente }
then

seguir la ejecución normal de programa

else // tres acciones sucesivas.

a. Crear un objeto con información del suceso (excepción)
b. Transferir el control de ejecución al "handler" correspondiente
c. Recibir el objeto para su análisis y decisión de la acción a seguir

en este caso la sintaxis utilizada es la siguiente:

```
try {          // bloque de código que se intenta
  ...
}
catch (...) { // captura de excepciones
  ...
}
...            // continua la ejecución normal
```

El diseño del mecanismo de excepciones C++, someramente expuesto, tiene la ventaja de permitir resolver una situación muy frecuente: el bloque en que se detecta el error no sabe qué hacer en tal caso (cuando se presenta el error o excepción); la acción depende en realidad de un nivel anterior, el módulo que invocó la operación.

Precauciones:

La respuesta más honesta es que el sistema perfecto e invulnerable no existe. Aunque el sistema de excepciones de C++ es una formidable herramienta para controlar imprevistos, que permite hasta cierto punto, controlar imprevistos dentro de imprevistos. A pesar de ello, nada puede sustituir a una programación cuidadosa. El propio Stroustrup advierte: "Aunque las excepciones se pueden usar para sistematizar el manejo de errores, cuando se adopta este esquema, debe prestarse atención para que cuando se lance una excepción no cause más problemas de los que pretende resolver. Es decir, se debe prestar atención a la seguridad de las excepciones.

Tipos de excepciones:

Durante la ejecución de un programa pueden existir dos tipos de circunstancias excepcionales: síncronas y asíncronas. Las primeras son las que ocurren dentro del programa. Por ejemplo, que se agote la memoria o cualquier otro tipo de error. Son a estas a las que nos hemos estado refiriendo. Las excepciones asíncronas son las que tienen su origen fuera del programa, a nivel del Sistema Operativo. Por ejemplo, que se pulsen las teclas Ctrl+C. Generalmente las implementaciones C++ solo consideran las excepciones síncronas, de forma que no se pueden capturar con ellas excepciones tales como la pulsación de una tecla. Dicho con otras palabras: solo pueden manejar las excepciones lanzadas con la sentencia throw.

El "handler" no puede devolver el control al punto de origen del error mediante una sentencia return. En este contexto, un return en el bloque catch supone salir de la función que contiene dicho bloque.

Secuencia de ejecución:

Como puede verse, la filosofía C++ respecto al manejo de excepciones no consiste en corregir el error y volver al punto de partida. Por el contrario, cuando se genera una excepción el control sale del bloque-intento try que lanzó la excepción (incluso de la función), y pasa al bloque catch cuyo manejador corresponde con la excepción lanzada (si es que existe).

A su vez el bloque catch puede hacer varias cosas:

Relanzar la misma excepción.

Saltar a una etiqueta.

Terminar su ejecución normalmente (alcanzar la llave } de cierre).

Puede ocurrir que el bloque-catch lance a su vez una excepción. Lo que nos conduce a excepciones anidadas. Esto puede ocurrir, por ejemplo, cuando en el proceso de limpieza de pila ("Stack unwinding") que tienen lugar tras una excepción, un destructor lanza una excepción

Constructores y destructores en el manejo de excepciones:

Cuando en el lanzamiento de excepciones se utilizan objetos por valor, throw llama al constructor copia. Este constructor inicializa un objeto temporal en el punto de lanzamiento. Cada vez que desde dentro de un bloque-try se lanza una excepción y el control sale fuera de bloque, tiene lugar un proceso de búsqueda y desmontaje descendente en la pila hasta encontrar el manejador ("Catcher") correspondiente. Durante este proceso, denominado "Stack unwinding", todos los objetos de duración automática que se crearon hasta el momento de ocurrir la excepción, son destruidos de forma controlada mediante llamadas a sus destructores. Observe que no se menciona para nada la destrucción de objetos persistentes que se hubiesen creado entre el inicio del bloque try y el punto de lanzamiento de la excepción. Esto origina una difícil convivencia entre el mecanismo de excepciones y el operador new. Para resolver los problemas potenciales deben adoptarse precauciones especiales.

Establecer opciones de manejo de excepciones

Los compiladores suelen disponer de comandos de compilación para determinar el tratamiento que seguirá el manejo de excepciones. A título de ejemplo se muestran algunos:

Opción	Compilador	Descripción
-x-	Borland C++	C++ Deshabilitar el manejo de excepciones C++ (habilitado por defecto).
-xd	Borland C++	Habilita limpieza total.
-xp	Borland C++	C++ Habilita información sobre las excepciones.
-fno-exceptions	GNU c++	Deshabilitar el mecanismo de excepciones.

(https://www.zator.com/Cpp/E1_6.htm, s.f.)