uc3m | Universidad Carlos III de Madrid

Informática Industrial I

Grado en Ingeniería en Electrónica Industrial y Automática

Álvaro Castro González

Nicolas Burus

Mohamed Abderrahim

José Carlos Castillo Montoya

Práctica 8

Excepciones y manejo de errores

A lo largo de los últimos años, han colaborado en impartir las prácticas presenciales, y por tanto proporcionar comentarios para mejorar algunos aspectos de este documento los profesores siguientes:

Ioannis Douratsos
Fares Abu-Dakka
Juan Carlos González Victores
Raúl Perula
Avinash Ranganath

Práctica 8 – Excepciones y manejo de errores.

Una excepción es un suceso anormal que rompe la ejecución normal de un sistema. Se puede decir que es una señal provocada por un error que va acompañada de información que identifica el tipo de error. Esta señal provoca la interrupción del hilo de ejecución si no se trata adecuadamente.

En C++, el sistema de manejo de excepciones gira en torno a tres palabras reservadas: `try`, `catch` y `throw`. En términos generales, se puede decir que el conjunto de instrucciones del que se quieren manejar las excepciones se encierra en el bloque `try`; cuando ocurre un error dentro del bloque `try`, se lanza una excepción utilizando la sentencia `throw`; finalmente la excepción es capturada y procesada por la instrucción `catch`. Hay que recalcar que las excepciones no son otra cosa sino una variable de un tipo de dato, y como tal hay que definirlas, declararlas y operar con ellas.

La forma general de la sentencia `try-catch` es la siguiente:

```
try {
      // try block
}
catch (type1 e) {
      // catch block where is processing type1 exceptions
}
catch (type2 e) {
      // catch block where is processing type2 exceptions
}
...
```

Como se puede observar es normal que haya varios `catch` anidados para distintos tipos de excepciones.

El tamaño del bloque `try` puede ser el que se desee: desde unas pocas instrucciones hasta abarcar la función `main` al completo.

Una excepción es capturada por el primer bloque `catch` donde se especifique el mismo tipo de dato que ésta. Si en un bloque `try` no se produce ninguna excepción, entonces no se ejecutará ningún bloque `catch`.

Para lanzar una excepción se emplea la orden `throw exception` donde `exception` será un dato de cualquier tipo. Para que esta excepción sea capturada tiene que ser lanzada desde dentro de un bloque `try`, bien sea directamente o indirectamente a través de llamadas a funciones donde se lance. Si una excepción no es manejada, está provocará la terminación anormal del programa.

Observar el siguiente ejemplo de uso:

```
#include <iostream>

using namespace std;

int main(int argc, char *argv[]) {
      cout << "Learning exceptions in C++" << endl;
      try {
            cout << “Inside try block” << endl;
```

```
            throw 99;
            cout << “This is not executed” << endl;
      }
      catch(int i) {
            cout << "Catched exception with value: " << i << endl;
      }
      cout << "END" << endl;

      return 0;
}
```

Darse cuenta de que una vez se lanza una excepción, el bloque `try` termina de inmediato y la ejecución pasa al bloque `catch` correspondiente.

Generalmente dentro del bloque `catch` se intenta solucionar el posible problema ocasionado o, si no es posible, terminar la ejecución del programa de una forma ordenada.

Si una excepción no se maneja dentro de la función que la generó, la excepción se reenvía hacia arriba a la función que invocó primero a está, y así sucesivamente hasta que se encuentra un manejador que la trate o hasta que provoque la terminación anormal de la aplicación.

Ejercicio

Crear un programa donde exista una función que realice la división de dos números que se le pasan como parámetros y devuelva el resultado. En el caso de que el divisor sea cero se tendrá que generar una excepción que será capturada en la función `main`.

```
#include <iostream>
#include <cstring> //para el strcpy

using namespace std;

class MyException {
      public:
            char mesage[80];
            int number;
            MyException() {
                  *mesage = 0; number = 0;
            }
            MyException(const char *msg, int n) {
                  strcpy(this->mesage, msg);
                  this->number = n;
            }
};

int main(int argc, char *argv[]) {
      int n;
      try {
            cout << "Insert a positive number: " << endl;
            cin >> n;
            if(n < 0) {
                  throw MyException("Negative number!", n);
```

```
            }
      }
      catch(MyException e) {
            cerr << "ERROR: " << e.mesage << endl;
            cerr << "value = " << e.number << endl;
      }

      return 0;
}
```

Una excepción puede ser de cualquier tipo, incluyendo clases creadas por el usuario. De hecho, normalmente la mayoría de las excepciones son de algún tipo de clase, puesto que esto permite incluir información extra sobre el error para poder tratarlo mejor o informar con más detalle.

El orden de los bloques `catch` puede tener mucha relevancia cuando se está tratando con excepciones de una clase padre y clases hijas. El bloque `catch` encargado de manejar las excepciones del tipo de la clase padre también puede capturar excepciones de sus clases hijas. Por lo tanto, si se quieren capturar excepciones de clases hijas y padres, se deben poner primero las encargadas de las hijas. Dicho de otra forma, los manejadores de las excepciones de una clase padre capturarán todas las excepciones.

1. Capturando todas las excepciones

Para capturar todo tipo de excepciones y darles el mismo tratamiento se emplea el siguiente método:

```
catch (...) {
      // processing exception
}
```

Un uso muy extendido de esto es emplearlo como un manejador por defecto de excepciones. Situando este bloque `catch` en último lugar de una serie de estos permitirá hacer un tratamiento generalizado de las excepciones que no se quieren manejar explícitamente. De este modo se evita una terminación anormal del programa.

2. Funciones para envío de excepciones

Es posible restringir las excepciones que una función es capaz de enviar fuera de ella misma añadiendo la cláusula `throw` seguida de los tipos de excepción permitidos entre paréntesis:

```
type_returned funtion_name(parameters_list) throw (type_list) {
      // function body
}
```

3. Realizando excepciones

Una excepción puede ser relanzada solamente desde dentro de su manejador llamando a `throw` sin ninguna excepción. Esto provoca que la excepción sea pasada a otro `try-catch` más externo. Para terminar recordarte que la clave en el manejo de excepciones es proporcionar una manera ordenada de manejar los errores y esto implica corregir la situación que ha provocado el error siempre que sea posible.

Ejercicio

Modificar el ejercicio anterior de la división por 0 para que ahora la excepción generada dentro de la función que realiza la división se trate ahí y luego la relance para que sea tratada en el `main` *también.*

Ejercicios

Continuar la implementación del simulador de ordenador. Crear una jerarquía de clases de excepciones que permitan controlar los posibles errores que surjan de una manera controlada, clara y elegante.

Para ello, crear una clase padre/base `ComputerException` que tendrá solamente un campo de tipo `string mesage` y que todas las demás clases de excepciones heredarán de ella. Ahora, crear las clases `InputException`, `ProcessorException` y `OutputException` que manejarán los respectivos errores a esos dispositivos. Por ejemplo, si se tiene un tipo `Keyboard` que es un fichero del que se van a leer las entradas, `KeyboardException` manejará los posibles problemas que pudieran aparecer a la hora de abrir el fichero, leerlo, etc.

Los errores que surjan deberán ser capturados y el programa deberá ser capaz de recuperarse del error siempre que sea posible (si no se puede abrir un fichero para leerlo porque no existe, habrá que volver a pedir la ruta del fichero). En el caso de que no se pueda recuperar de un error, el programa terminará de forma ordenada e indicando con un mensaje claro en que parte del computador se ha dado el error.

Un posible diseño general podría ser el siguiente: