Chap07 容错与异常处理

程勇

北京化工大学

信息科学与技术学院计算机系

Sept. 2020

课程提纲

- Chap01 绪论
- Chap02 面向对象程序设计方法
- Chap03 重载与类型转换
- Chap04 继承与派生
- Chap05 多态性
- Chap06 输入输出流
- Chap07 容错与异常处理
- Chap08 模版

本章提纲

- 异常处理简介
- C++异常处理机制
  - throw, try-catch语句
  - 异常匹配规则
- 捕获对象异常
  - 异常处理过程
  - 捕获派生类异常
  - 重新抛出异常
  - 处理所有异常
- 标准异常库

基本概念

- 错误处理
  - 错误处理是编程中的一个重要元素，其质量将决定程序的健壮程度和友好程度；
  - 异常提供了处理错误的另一种方式，其主要优点是导致出错的代码和错误处理代码完全分隔开来；
- 异常
  - 异常(Exception)是临时对象，用于检测程出现的运行时不正常的情况，如存储空间耗尽、数组越界、被0除等；
  - 异常可以使基本类型变量，如int或char*，但通常它是为处理错误而专门定义的类对象；
- 异常处理机制
  - C++提供了一些内置的语言特性来产生(raise)或抛出(throw)异常，用以通知“异常已经发生”，然后由预先安排的程序段来捕获(catch)异常，并对它进行处理；

基本概念(续)

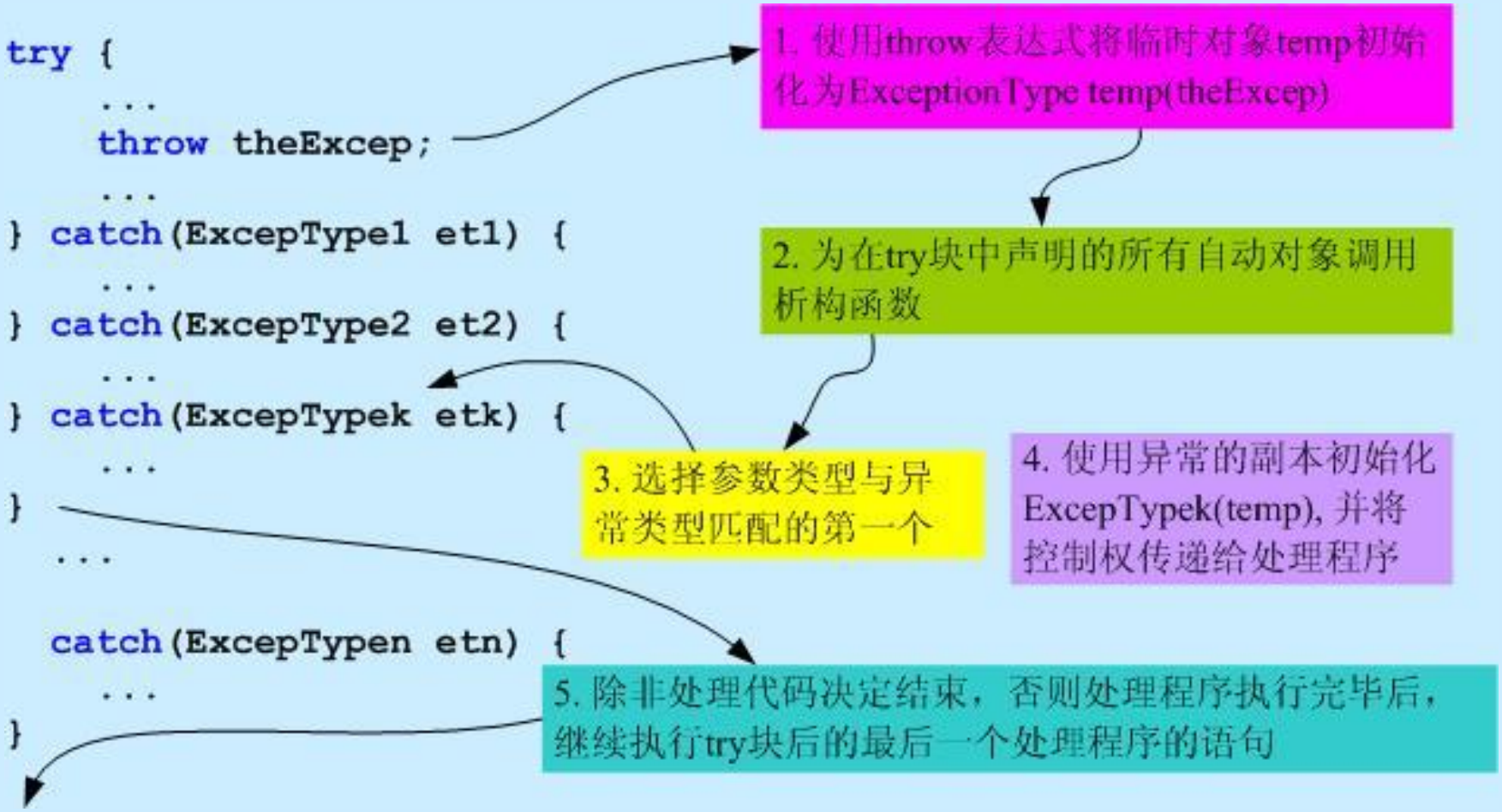
- 首先采用关键字try，构成一个try块(try block)，它包含了抛出异常的语句，当然也可以是包含了这样的调用语句；
- 然后由catch块捕获并处理异常，每个catch块包含处理某种异常的代码；

- 异常处理一般格式

```
try {
        …
} catch (ExceptionType1 et1) {
        …
} catch (ExceptionType2 et2) {
        …
} catch (ExceptionType3 et3) {
        …
}
```

基本概念(续)

- 异常处理过程



基本概念(续)

- 注意事项
  - 编译器在匹配异常类型和catch参数时，会忽略关键字const；
  - 如果try块中的异常没有由catch块捕获，则会调用头文件<exception>中声明的库函数terminate()，该函数再调用默认的中断处理函数，然后再调用头文件<cstdlib>中声明的abort()函数，abort()函数会立即中断整个程序。在某些情况下，可以使用set_terminate()来替换默认的中断处理程序； set_terminate()函数接受terminate_handler类型的参数；terminate_handler是一个指针(typedef void(*terminate_handler)())，可以指向任何没有参数也没有返回值的函数；
  - 可以在一个try块中嵌套另一个try块，两个try块都可以调用函数，内层try块抛出异常时，其内层的catch块首先处理它，如果不能匹配，则由外层的catch块来捕获该异常；
  - 应该在对程序结构和操作最方便的地方处理异常，极端情况下，由main()函数处理程序中的所有异常；

捕获对象异常

- 类对象作为异常
  - 作为异常的类对象可以是任意类的对象，由于异常主要用于错误处理，因此一般其核心是与处理程序交流错误信息，并且常定义一个特殊的类来表示程序中的某一类型的错误；
- 参数匹配情形
  - 参数类型与异常类型完全匹配，忽略const；
  - 参数类型是异常类型的直接或间接基类，或是异常类的直接或间接基类的引用，忽略const；
  - 参数和异常都是指针，异常类型可以自动转换为参数类型，忽略const；
- 注意事项
  - 异常对象的类型必须是可以复制的，如果对象的类副本构造函数是私有的，该对象就不能用作异常；
  - 由于异常对象总是在抛出时复制，所以在catch块中总是使用引用参数；

捕获对象异常(续)

- 重新抛出异常
  - 在处理一个异常时，可以重新抛出该异常，让外层try块的处理程序来捕获它；

```
throw trouble;
```

  - 重新抛出的异常不是复制的；
- 捕获所有异常
  - 可以使用…作为catch块的参数说明来捕获所有异常；
- 指定函数可能抛出的异常
  - 在函数头添加异常说明，可以指定函数可能抛出的异常集；

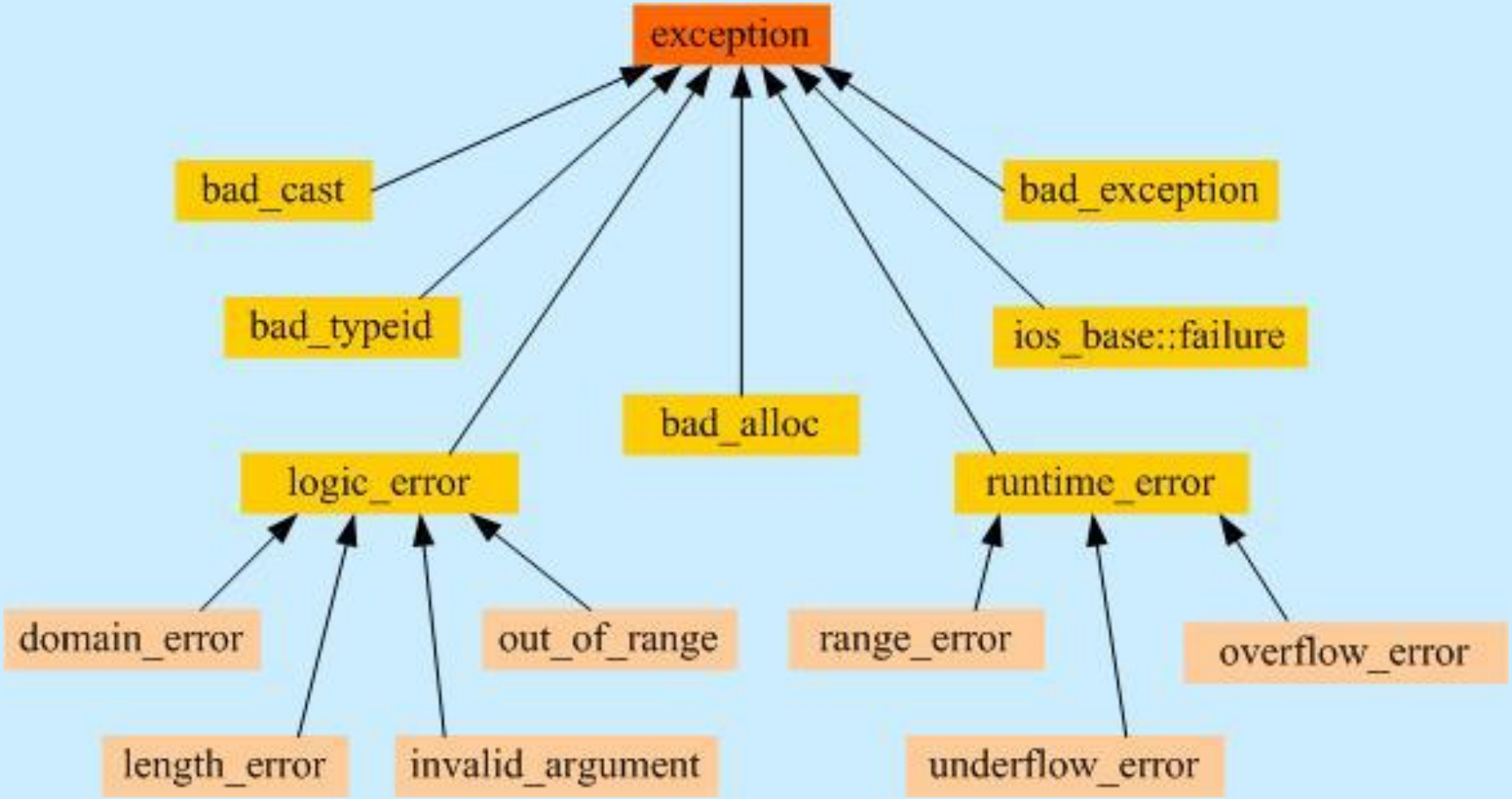
```
void dothat(int arg) throw (Trouble, MoreTrouble){ }
```

  - 如果使用typedef来定义函数指针的类型，不能包含异常说明，但使用指针声明来使用该类型时就必须包含它；

```
typedef void(*FunPtrType)(int);
FunPtrType pfun throw(Trouble, MoreTrouble);
```

标准异常库

- 标准异常类


exception
bad_cast
bad_exception
bad_typeid
ios_base::failure
bad_alloc
logic_error
runtime_error
domain_error
out_of_range
range_error
overflow_error
length_error
invalid_argument
underflow_error

本章小结

- 异常处理简介
- C++异常处理机制
- 捕获对象异常
- 标准异常库

- 课程网站
  - http://www.jiaowu.buct.edu.cn