异常处理

·异常（**Exceptions**）：程序的执行过程中发生的问题（可以被预知）

·异常处理（**Exception handling**）：解决异常，使程序健全和可容错（允许程序继续执行或通知用户问题）

·**Exception** is a special kind of error that the user **can not estimate（不能被估计）** and **occurs infrequently（很少发生）.**

·错误一旦被处理，就不是错误。

常见的异常

·**系统资源不足**

1）内存不足：不可以动态申请内存空间

2）磁盘不足：不能创建新的文件

【电脑内存不足的其中一种情况是开的程序过多，电脑处理不过来，这种是内存不足？

电脑磁盘不足就是硬盘已经被塞满了东西已经无法再添加】

·**用户操作导致运算关系不正确**

如分母为0、数学运算溢出、数组越界、参数类型不能转换等。

**异常的特点**

·**偶然性**和**可预见性**——异常的存在和出现可以预见，但不会总是发生。

·**严重性**——一旦异常产生，程序可能终止或运行结果未知。

异常处理机制

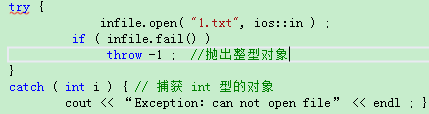
**·throwing an exception（抛出异常信号）**

【就是告诉用户这里出问题了】

**·handling the exception**

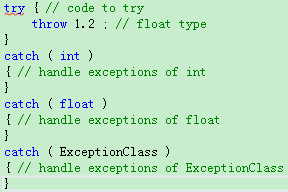
try（检查）, catch（捕捉） and throw（抛出）

try and catch：



一般 try 会和 catch匹配，如果没有catch，try如果异常则直接跳过此块

一个cry会和一个或多个catch匹配，例如

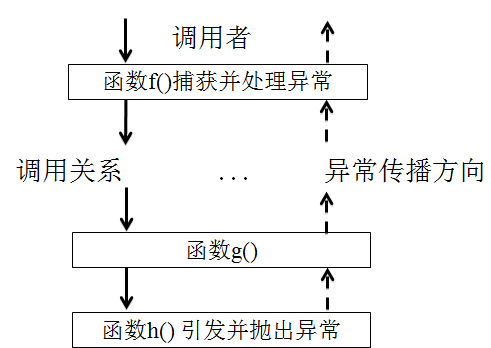


按照抛出的类型与catch匹配，因为仅仅通过类型而不是通过值来匹配的，所以catch块的参数可以没有参数名称，只需要参数类型

catch按照出现的前后顺序被检查，只要匹配catch就处理该异常，就不会再往下

如果throw抛出的异常信息找不到与之匹配的catch块，那么系统就会调用一个系统函数terminate，使程序终止运行

异常处理的思想



【不是看得很懂，大概是一级一级往下匹配，然后抛出的时候再一级一级往上，如果没有匹配的catch就会一直回到程序】

栈展开(stack unwinding)

沿嵌套函数调用链继续向上，直到为异常找到一个catch子句。只要找到能够处理异常的catch子句，就进入该 catch子句，并在该处理代码中继续执行。当catch结束的时候，在紧接在与该 try 块相关的最后一个 catch 子句之后的点继续执行。

【目前理解的是，这个过程跟switch很像，找到匹配的点就执行，执行完了就break】

异常与析构函数

·因异常而退出函数时，编译器保证适当地撤销局部对象

·如果一个块直接分配资源，而且在释放资源之前发生异常，在栈展开期间将不会释放该资源。【就是说只会释放异常的部分的内存，在异常前已经分配好的内存是不会改变的？】

·析构函数应该从不抛出异常

catch异常说明符匹配原则

除下面几种可能的区别之外，异常的类型与 catch 说明符的类型必须完全匹配：

·允许从非 const 到 const 的转换。

·允许从派生类类型到基类类型的转换

·将数组转换为指向数组类型的指针，将函数转换为指向函数类型的适当指针

throw

·followed by an operand representing the type of exception.【跟在一个表示异常类型的操作数后？？】

·可以抛出任何类型

·If the **throw operand** is an object, it is called an exception object. （常用）【此句不是很懂】

·在函数的声明中列出可能抛出的异常类型



如果没有特别说明，如，则表示可以抛出一切类型

还有一种写法为

Exception Class

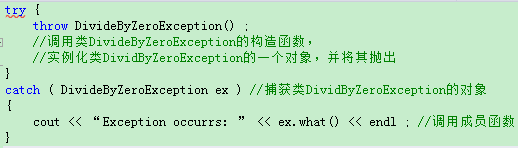
·一般C++库里自带

·it is common to define a class whose objects can carry the kind of information you want thrown to the catch-block【它的对象可以带着你想要的信息抛出？？】

·you can have a different type to identify each possible kind of exceptional situation.

【有一个不同的类型来确定每一个可能的一种特殊情况？？】

异常类

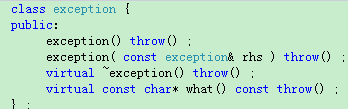


·抛出对象可以传出更多的信息

·catch的时候是传引用，即下面即使有多个catch，都只会生成一个类的对象，这个效率会比较高

标准C++异常类

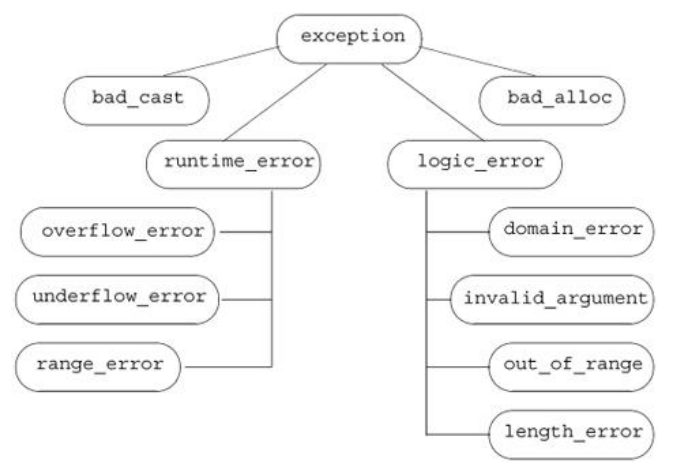
一个基类：exception　（所有Ｃ＋＋异常的基类）



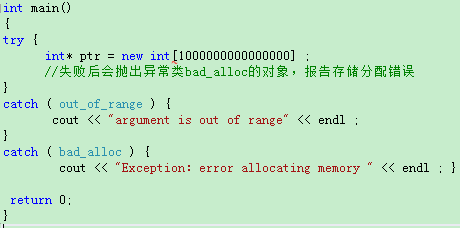
有throw()即不抛出任何类型的异常的声明。

【为什么要加这个？】

异常类层次图：



例子：



【VS不让我运行……】

·加上这个头文件之后，在new的时候如果出错会自动抛出异常类的对象（PS. new是运算符而不是关键词，可以重载nes）

异常处理机制的缺点：

·带来了额外的**时间开销**和**空间开销**（很多时候用if判断就可以了）

**·**容易过度使用，即滥用。如滥用**抛出异常**来代替**错误返回**，会使程序像运行大量goto语句一样混乱，产生不可预知的出口，这违反了面向对象的原则——**结构化**和**封装**的原则。

建议与需要注意的地方：

·分清错误和异常，消除错误，控制异常。（能不用异常处理的就不用，尽量用其他方法捕捉到错误并处理）

·**异常处理**可以使**出错处理程序**与**“通常”代码**分离开来，使代码更简洁更灵活，提高程序必不可少的健壮性

·如果抛出的异常一直没有被程序捕获(catch)，则会一直上传到操作系统，导致整个程序的终止

·一般在异常抛出后，资源可以正常被释放，但是如果在构造函数中抛出了异常，其析构函数不会被调用，处理方法是：**在抛出前要记得删除申请的资源**。【主动delete？】

·catch块的参数推荐采用地址传递而不是值传递，不仅可以提高效率，还可以利用对象的多态性。【多态性？】

·为了加强程序的可读性，方便地知道所使用的函数会抛出哪些异常，应该在函数的声明中列出这个函数可能抛出的所有异常类型