面向对象程序设计(OOP)

C++(第5章：异常处理)

福州大学·软件学院·软件工程系
王灿辉(wangcanhui@fzu.edu.cn)

第5章：异常处理

概述：实际案例

- 第1个案例：我怎么有两种性别？

  用户抱怨：系统经常异常终止、数据不正确、数据不一致

- 第2个案例：已经完成的子系统？

  用正确的数据执行可以得到正确的结果，错误的数据将导致不可预料的结果。

第5章：异常处理

概述：案例的启示

- 必须**检测**数据(尤其是用户输入的数据)的正确性、完整性和一致性、**捕获**出现的异常并进行适当的**处理**。
- 异常的处理应贯穿整个软件开发过程，而不是在编完程序后再增加(程序注解也是如此要求)。

第5章：异常处理

概述：健壮性(不处理错误)

- 程序的“健壮性”对使用者极为重要！！
- 编写一个计算两个数相除的函数：

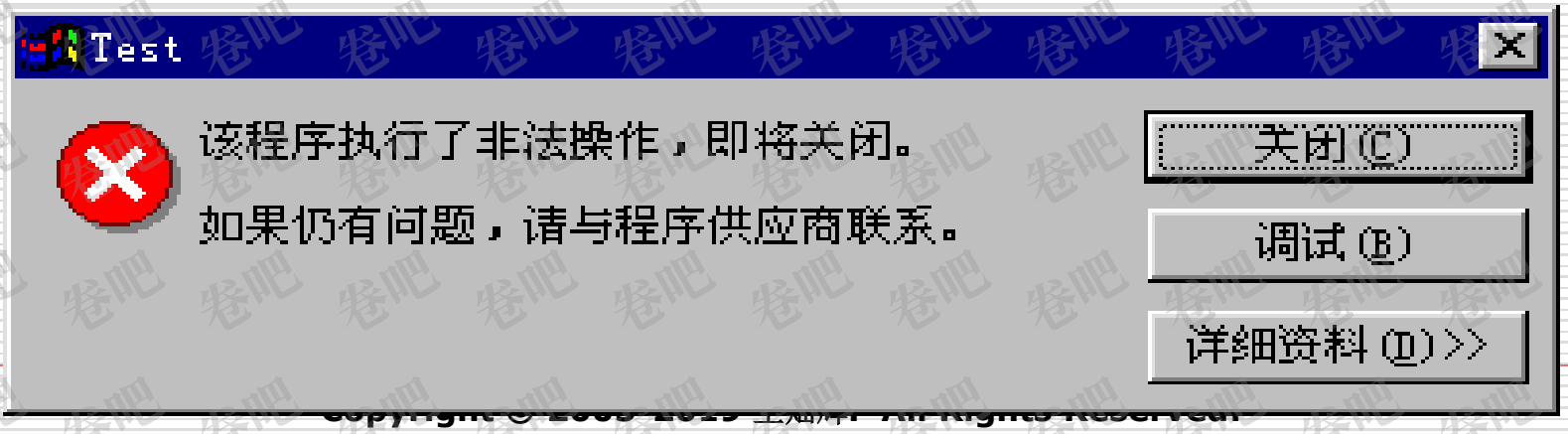
```
int division(int x,int y)
{
   return x/y; //y=0时怎么办？
}
```

- 对各种意外情况应能够给出恰当的处理。

第5章：异常处理

概述：健壮性(不处理错误)

```
int division(int x,int y)
   {return x/y;}
void main() { //输出2后出现下述提示信息
   cout<<division(12,5)<<endl;
   cout<<division(12,0)<<endl;
}
```



第5章：异常处理

概述：健壮性(如何处理？)

- 编写一个计算两个数相除的函数：

```
int division(int x,int y)
{
   if (y==0) return ?;
   else      return x/y;
}
```

- 发现一个异常(错误)，怎么办？

第5章：异常处理

概述：异常处理的基本思想

- 许多异常是可以预料的，但无法避免，如：用户输入错误、内存溢出、打印机未连接好、文件不存在、无U盘等。
- 处理方法：最好能暂停当前工作，允许用户排除错误继续执行程序，或至少给出适当的提示信息。
- 异常：发现、检测、捕获和处理异常。

第5章：异常处理

概述：处理用户输入错误

- 强行进行纠正。
- 报告错误，要求重新输入，直到一定的次数或直到输入正确为止！
- 报告错误，终止程序的运行。
- ……

第5章：异常处理

概述：谁处理异常？

- 调用函数和/或被调用函数处理？
- 处理方法：检测并报告，或检测并处理？
- 分工合作，严格规定！

第5章：异常处理—处理方法

概述：直接处理错误

- 编写一个计算两个数相除的函数：

```
int division(int x, int y) {
    if (y==0) {
        cout << “exception of deviding
zero.\n” ;
        exit(-1);  //结束所有程序，返回OS
    }//经典但不好的处理方法！
    else return x/y;
}//函数直接处理异常有时并不合适！
```

第5章：异常处理—处理方法

概述：返回错误

- 编写一个计算两个数相除的函数：

```
bool division(int x,int y,int &r)
{//经典的处理方法！
   if (y==0) return false;
   else{r=x/y;return true;}
}//由函数返回错误，调用者据此处理！
```

- 但调用程序可以选择忽略被返回的错误

第5章：异常处理—处理方法

概述：调用统一的错误处理函数

- 编写一个计算两个数相除的函数：

```
int division(int x,int y)
{
  if (y==0)
     {return error( “Divide by 0.” );}
  else return x/y;
}//error为统一的错误处理函数！
```

第5章：异常处理—处理方法

C++新增的处理方法：抛出异常

- 编写一个计算两个数相除的函数：

```
int division(int x,int y) {
   if (!y) throw x;
   else    return x/y;
}//发现、检测出异常后抛出异常(类)
```


- 异常不必在发生错误的地方被处理，这可以方便库函数的编写！

第5章：异常处理

概述：C++异常处理的基本思想

- 一般情况下由类专门检查各种可能出现的错误(异常)，并用**throw语句**抛出异常；而类的使用者则用**try语句**捕获由throw抛出的异常，并用**catch语句**提供具体的异常处理程序。

第5章：异常处理

没有捕获异常

- 调用除法函数(抛出异常)的主函数：

```
void main(){ //没有处理抛出的异常！
    cout<<division(12,5)<<endl;
    cout<<division(12,0)<<endl;
}//输出2后
  //出现下述提示信息
```

第5章：异常处理

没有捕获异常

- 调用除法函数(抛出异常)的主函数：

```
void main()  {//没有处理函数抛出的异常！
    cout<<division(12,5)<<endl;
    cout<<division(12,0)<<endl;
    cout<<division(12,7)<<endl;
} //在DOS方式下输出2后,报告：“abnormal
    program termination”
```

第5章：异常处理

概述：C++的异常处理方法

- 编写一个测试除法函数的主函数:用try语句捕获由throw抛出的异常，并用catch语句提供具体的异常处理程序。
- 完整的源程序
- catch语句处理的错误类型必须和函数抛出的错误类型一致，实例
- 可以用catch(…)匹配捕获的所有类型的错误(类似switch的default语句)，实例

第5章：异常处理

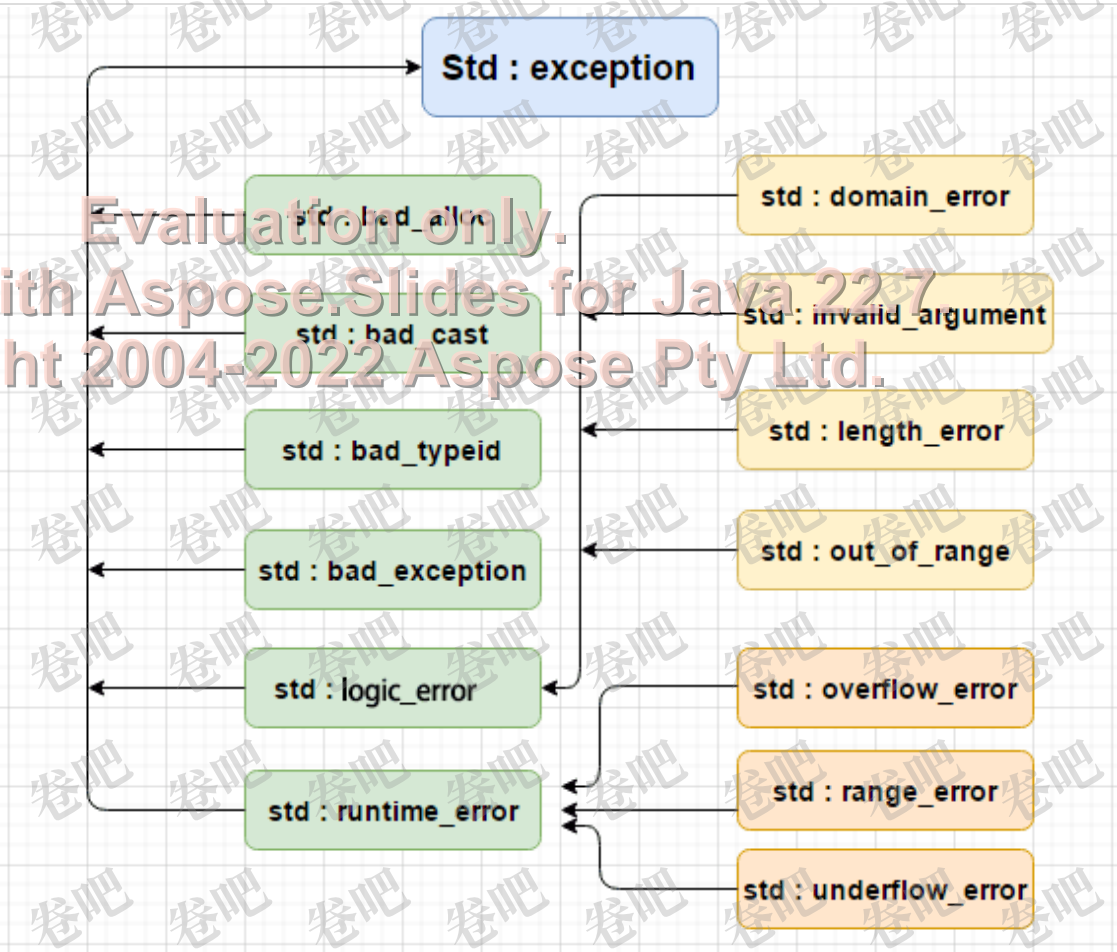
异常抛出、捕获和处理

- throw 表达式
- try块{//含throw语句或使用了throw语句的函数}
  catch块(异常类 [对象形参]){处理throw语句抛出的异常}…
- 如果try块没有发生异常，程序将跳过try块后所跟的所有catch语句继续执行
- 异常抛出、捕获和处理的完整实例

第5章：异常处理

异常抛出、捕获和处理

- 可以使用C++标准中的异常类来创建异常处理对象，抛出异常。



第5章：异常处理

异常抛出、捕获和处理

- 头文件<stdexcept>，根类exception
- **#include <stdexcept>**
- **throw runtime_error("除零错！");**
- **catch (runtime_error &ex)**
- **{cout << ex.what() << endl;}**
- 使用C++异常类的实例

第5章：异常处理

异常抛出、捕获和处理

- 对于那些不能使用C++标准异常类的异常，可以自定义异常类（其和一个普通类没有什么区别），但通常应该（不是必须的，但却是一个好的习惯）派生自exception类或该类的某个派生类。

- 自定义异常类的实例

第5章：异常处理

C++的处理：利用异常

- 异常处理并不仅限于显示错误，我们也可以利用异常，把程序从异常状态恢复到正常状态（这正是C++中称为异常处理而不称为错误处理的主要原因）
- 利用异常的完整的源程序

第5章：异常处理

异常处理：没有或错误捕获

- 当catch块捕获的异常类型与try块中抛出的异常类型不同时，该catch块将不会被执行，因此也就不会捕获到异常！
- 错误捕获和处理的实例

第5章：异常处理

函数嵌套调用时的异常处理

- 如果当前函数没有用try块(和catch)捕获异常或catch没有捕获到该异常，则系统继续将该异常抛给调用它的函数！直到遇到一个能捕获此异常的函数或OS。即没有处理异常的函数遇到异常相当于执行了再次抛出功能(执行throw语句)。
- 函数嵌套调用时的异常处理实例

第5章：异常处理

更改系统终止处理的缺省行为

- 如果直到最外层的main()程序也不能处理某函数抛出的异常对象，这时系统会自动调用一个函数：void terminate();该函数缺省情况下将调用abort()来终止程序的运行，我们可以通过调用函数set_terminate来改变terminate()的缺省行为。
- 完整的源程序

第5章：异常处理

利用异常类记录异常的状态

- 可以把异常发生时的一些异常状态记录在异常对象中。当抛出异常对象之后，处理程序可以根据这些状态信息按具体的错误进行处理。格式形如：

  catch（类名::异常类 异常对象）{//…}

- 利用异常类记录异常状态的实例

第5章：异常处理

多个catch语句

- 一个类可以抛出多个异常，一个try块也可以对应多个catch语句，但最多只执行一个catch语句(不需要break)。顺序检测到匹配的第1个catch语句并执行，然后自动跳过后面的所有catch语句。
- 多个catch语句的实例

第5章：异常处理

捕获所有异常

- 可以用catch(…){//……;}语句捕获所有类型的异常，不过该语句必须放在所有catch语句的最后面(否则其他语句就没有机会执行了)。
- 演示实例

第5章：异常处理

异常处理的分工和合作

- 当有多个异常时，可以由某个函数处理部分异常，而其它函数处理另外的异常，这样方便了异常的组织。
- 异常处理的分工和合作实例

第5章：异常处理

再次抛出异常

- 有时处理程序在捕获到一个异常对象之后，仍不知道如何处理，为此它要把这个异常对象再次抛出，希望其他函数能更好地处理它。
- 再次抛出异常的演示实例

第5章：异常处理

捕获基类和派生类的异常

- 基类的catch语句也将匹配从该基类派生的任何类。因此，如果想捕获基类类型和派生类类型的异常，则应在catch序列中将派生类置于前面，否则将永远执行不到，编译器会给出警告。
- 完整的源程序

第5章：异常处理

概述：异常与析构

- C++在异常抛出前会自动为所有异常发生之前构造的局部对象调用析构函数
- 当找到一个匹配的catch子句时，如果其异常类型声明是值参数则复制被抛出的异常对象。如果为引用调用则不发生复制。
- 完整的演示源程序

第5章：异常处理

概述：异常接口

- 可以在函数定义中限定函数能够抛出的异常种类，缺省允许抛出任何类型的异常，throw()子句限制函数不能抛出异常。目前该子句在VC下可以使用但不起限制作用(仍然可以抛出任何异常)。
- 完整的源程序

第5章：异常处理

处理内存申请的异常

- 可以通过调用函数set_new_handler()来改变内存申请异常处理的缺省行为。

```
void out_of_store(void) {
   //……
   throw bad_alloc();
}
void main() {
   set_new_handler(out_of_store);
   //……
}
```

第5章：异常处理

异常嵌套

- 异常可以嵌套，也就是在catch语句处理异常时，catch语句本身也可以捕捉异常并做相应处理。不过由于这种嵌套形式过于复杂，难于理解，因而并不建议使用。

第5章：异常处理

异常的组织

- 对多种异常可采用各种形式进行组织：用枚举组织异常、用派生类组织异常、利用虚函数来组织异常、用多继承来组织异常、用模板组织异常。
- 详细参见其他书籍[麦中凡 编著，C++程序设计语言教程(语言基础)，北京航空航天大学出版社，1995.]

本章内容讲授到此结束！

福州大学·软件学院·计算机教研室
王灿辉(wangcanhui@fzu.edu.cn)