《面向对象程序设计》朋辈课程 · 第二辑

C++异常处理

信息学院 赵家宇

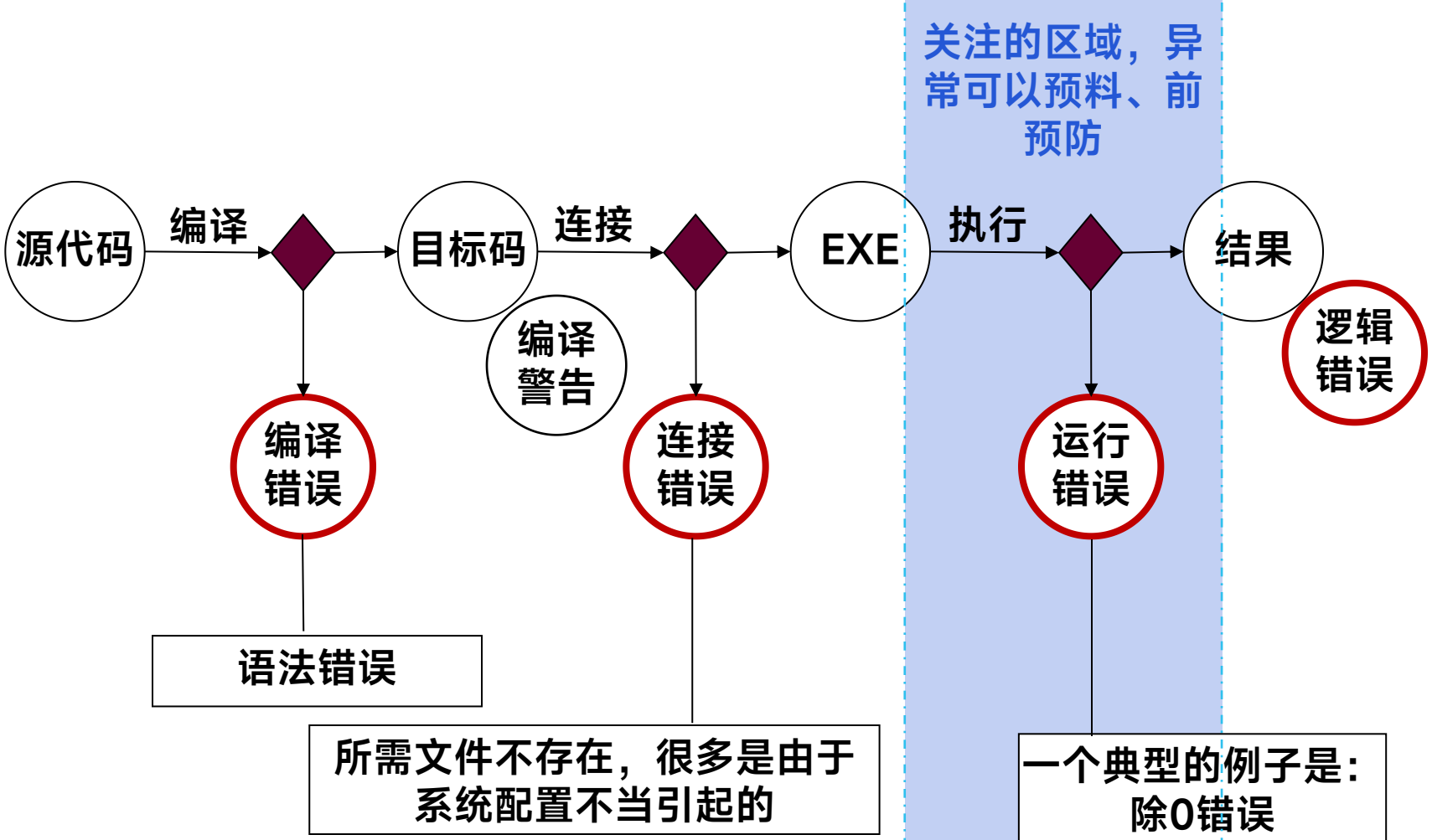
PART 01

引入异常处理

为何引入异常处理

- 程序需要具备特殊情况下处理潜在错误和异常事件的能力，也即编写的程序不仅要保证其在合理输入下的正确性，而且要具有一定的容错能力。
- 这体现在程序不仅在正确的操作条件下要运行正确，而且在出现意外的情况下，也应该能有合理的正确的表现，不能出现灾难性的后果（如数据丢失）。
- 程序员在程序设计时要考虑各种意外的情况，并给予恰当的处理，在C++中通常借助“异常处理”机制来实现这样的目标。

程序错误的不同种类


关注的区域，异常可以预料、前预防
源代码
编译
目标码
连接
EXE
执行
结果
编译警告
编译错误
连接错误
运行错误
逻辑错误
语法错误
所需文件不存在，很多是由于系统配置不当引起的
一个典型的例子是：除0错误

程序运行错误的现象

- 运行错误：在运行过程中会出现异常，得不到正确的运行结果，甚至导致程序不正常终止，或出现死机现象，例如：
  - 在一系列计算过程中，出现除数为0的情况；
  - 内存空间不够，无法实现指定的操作；
  - 无法打开输入文件，因为无法读取数据；
  - 输入数据时数据类型有错；
- 显然，我们可以使用C语言中的if…else…语句来解决大部分问题，但是这样的程序编写不仅结构冗余，且仍然存在出错的可能。

异常处理的任务

- 程序异常处理的任务：在设计程序时，应当事先分析程序运行时可能出现的各种意外的情况，并且分别制定出相应的处理方法；其主要为了解决“程序运行时若情况出现异常，由于程序本身不能处理，程序只能终止运行”的问题；
- 如果在程序中设置了异常处理机制，则在运行情况出现异常时(异常检测)，由于程序本身规定了处理的方法，于是程序的流程(控制权)就转到异常处理代码段处理(错误处理).从而可避免程序的中止。

异常的处理手段和方法

- C++提供了异常处理机制。异常处理机制提供了处理错误的另一方式，此类方式更适合大系统。其主要优点包括:
- 1、可以将功能代码与错误处理代码隔离开;
- 2、在子函数中发现异常，可以将异常提交给上层函数处理，这样上层函数有机会把已经完成的工作做妥善处理。

PART 02

异常处理方法

异常处理方法

从代码的角度看，C++处理异常的机制：

- 抛出throw：当出现异常时发出一个异常信息
- 检查try：把需要检查的语句放在try块中
- 捕捉catch：捕捉到异常信息，捕捉到后就处理

异常处理方法

- try块：异常检测，并对检测到的异常进行处理。try块以try关键字开始，并以一个或多个catch子句结束。在try块中执行的代码所抛出的异常在被检测到后，通常会被其中一个catch子句处理；
- throw表达式：使用这种表达式来说明遇见了不可处理的错误（抛出异常）；由标准库定义的一组异常类：用来在throw和相应的catch之间传递有关的错误信息。

处理异常—throw表达式

- 一般形式：`throw 表达式;`
- 功能：某程序发现自己不能处理的异常，就可以使用throw抛出异常；
- 程序执行throw语句时，函数将终止执行，程序流程将转向异常处理部分；
- 如果程序中有多处要抛出异常，应该用不同的表达式类型来互相区别，表达式的值不用来区别不同的异常。

处理异常—catch表达式

- 一般式：`catch (double)`

  圆括号中，一般写异常信息的类型名；此时catch只检查捕获信息的类型，不检查值；

  若需要检测不同的异常信息，应当由throw抛出不同类型的信息，由catch匹配不同类型。

- 接收信息式：`catch (double d)`

  如果抛出的异常信息是指定类型的，则catch会将异常信息进行拷贝以便后续处理。

- 一网打尽式：`catch (…)`

  可以捕捉任何类型的异常信息，一般放在try-catch结构最后，相当于“其他”。若放第一，则屏蔽后面的其他所有catch语句。

处理异常—try/catch结构

```
try{    //复合语句：可能出现异常的程序代码
        //这些语句有可能直接抛出异常，或间接抛出异常。
    }
catch( 类型说明符1 )
     {                    } //处理异常类型1的代码
catch( 类型说明符2 )
     {                    } //处理异常类型2的代码
```

处理异常—try/catch结构

- try/catch语句是一对，中间不能有语句分割；如果使用try，则后面必须紧跟着catch。
- 无论是try或是catch，{}都是不能省略的。例如：

```
try {
      quotient=divide(num1,num2);
      cout<<“商是：”<<quotient<<endl;
} //这是一个分析可能出现异常后，在这里布点
catch(char *exceptionString){
      cout<<exceptionString;
}
```

处理异常—try/throw/catch复合结构

- try块中的复合语句是代码的保护段，用来框定那些有可能产生异常的代码语句；这些语句有可能使用throw语句直接抛出异常，或通过调用其他带有throw的函数来间接抛出异常；
- catch块必须紧接着try块，用于存放将异常处理的语句；其中的类型说明符可以是任何有效的数据类型，包括C++的类；
- 当异常被抛出后，便依次检查catch块，若某个块的异常类型声明与被抛出的异常类型一致，则执行该段异常处理程序。

处理异常—try/throw/catch复合结构

```
try{
    函数f();

    throw([异常信息]);
}
```

```
catch(){
    …
}
```

```
函数f{…}(

    throw([异常信息]);

)
```

异常处理示例

```
try
  { …                          //包含可能会有异常的代码
    if (test>5)
       throw “Test is greater than 5”; //抛出异常
    …                          //若没有抛出异常,代码将继续执行
  }
catch(const char* message)
  {  …                          //处理异常的相关代码
      cout<<message<<endl;
  }
```

思考：求三角形面积

```
double trg(double a,double b,double c){
        double area, s=(a+b+c)/2;
        area=sqrt(s*(s-a)*(s-b)*(s-c));
        return area;
}

int main(){
    double a, b, c;
    cin >> a >> b >> c;
    while (a>0 && b>0 && c>0){
        cout<<trg(a, b, c)<<endl;
        cin >> a >> b >> c;
    }
    return 0;
}
```

- 哪些代码需要检查?
  - 把需要检查的语句放在try块中
- 哪个部分会出现异常?
  - 当出现异常时throw发出一个异常信息
- 对异常如何进行操作?
  - catch捕捉异常信息，捕捉到后就处理

思考：求三角形面积

```
double trg(double a,double b,double c){
    double s=(a+b+c)/2;
    if (a+b<=c||b+c<=a||c+a<=b) throw a;
    return sqrt(s*(s-a)*(s-b)*(s-c));
}
int main(){
    double a,b,c;
    cin>>a>>b>>c;
    try{
        while (a>0 && b>0 && c>0){
            cout<<trg(a,b,c)<<endl;
            cin>>a>>b>>c;}
    }
    catch(double){
        cout<<"Not a triangle"<<endl;
    }
}
```

- 哪些代码需要检查?
  - 把需要检查的语句放在try块中
- 哪个部分会出现异常?
  - 当出现异常时throw发出一个异常信息
- 对异常如何进行操作?
  - catch捕捉异常信息，捕捉到后就处理

异常处理：拓展

- 在try块包含可能抛出异常的代码，指的是逻辑上的包含关系。例如在try块中调用一个函数，在该函数中抛出的任何异常都会被该try块的某个catch块捕获。
- 其具体逻辑为：若在该函数中抛出的异常没有被函数本身捕获和处理(无catch()或类型不匹配)，则该异常就会传递到上一级的调用函数，以此类推；如果异常到达没有catch处理程序的一级，且仍未被捕获，就调用中断处理程序，结束程序。

异常处理示例

```
try
{ …
   fun1();
   …
   fun2();
   …
}
catch(…)
{ …
}
```

```
void fun1()
{
… //这里可能抛出异常fun3();
…
}
```

```
void fun3()
{
…//这里可能抛出异常
}
```

```
void fun2()
{
…//这里可能抛出异常
}
```

备注：

1．这里的fun1()和fun2()和fun3()中，均无对应的catch()；

2．黑色线表示正常返回；彩色线表示异常返回。

异常处理：拓展

- 若在try块中抛出的异常没有由catch块处理，处理的过程如下：
  - 调用标准库函数terminate();
  - terminate()将调用预先定义的默认中断处理函数;
  - 默认中断函数再调用标准库函数abort(),从而立即中断整个程序。
- 通过这样的设计，能有效地将函数功能处理代码与错误处理代码分离。

感谢倾听

给个好评！