田永铭-221900180 作业十

一、概念题

1. 通过函数返回值实现的异常异地处理有什么不足?

- 通过函数的返回值返回异常情况会导致正常返回值和异常返回值交织在一起，有时无法区分。
- 通过指针/引用类型的参数返回异常情况，需要引入额外的参数，给函数的使用带来负担。
- 通过全局变量返回异常情况会导致使用者会忽视这个全局变量的问题。(不知道它的存在)
- 程序的可读性差！程序的正常处理代码与异常处理代码混杂在一起。

2. 什么是异常处理的嵌套?

try 语句是可以嵌套的:

- 在 try 语句块的语句序列执行过程中还可以包含 try 语句块。

■ 当在内层的 try 语句的执行中产生了异常，则首先在内层 try 语句块之后的 catch 语句序列中查找与之匹配的处理，如果内层不存在能捕获相应异常的 catch，则逐步向外层进行查找。

■ 如果抛掷的异常对象在程序的函数调用链上没有给出捕获，则调用系统的 terminate 函数进行标准的异常处理。默认情况下，terminate 函数将会去调用 abort 函数。

解释下列程序中抛出的异常被处理的位置?

1. g() 函数中的 try 块内:
   - 当调用 h() 函数时，如果在 h() 函数中抛出了 EA 类型的异常，它会被 g() 函数中的 catch (EA ea) 捕获，并在相应的 catch 块中处理。
2. f() 函数中的 try 块内：f 通过调用 g 调用 h
   - 如果 h 抛出的异常是 EA 类型，它会被 g 中 catch (EA ea) 捕获，并在相应的 catch 块中处理。
   - 如果抛出的异常是其他类型的 Exception (例如 EB 或 Exception(2))，它将被 catch (Exception e) 捕获，并在相应的 catch 块中处理。
   - 如果抛出的异常是 int 类型的值 (例如 throw 8)，它将被 catch (int) 捕获，并在相应的 catch 块中处理。
3. main() 负责调用 f 函数

3. 程序中的断言有什么作用?

为了确认程序运行到这些地方时状态是否正确。

- 帮助理解程序和形式化验证。
- 在程序开发阶段，帮助开发者发现程序的错误和进行错误定位。

编程题:

1. 示例程序中给出了一种除零异常的处理方式，请将程序修改为: 遇到除零异常时不终止，一直到获得正确的数 据为止。

修改后的代码：//标红处为核心修改

```
#include <iostream>
using namespace std;

int divide(int x, int y)
{
	if (y == 0) throw 0;
	return x/y;
}

void f()
{
int a,b;
	try
	{
		cout << "请输入两个数：";
		cin >> a >> b;
		int r = divide(a,b);
		cout << a << "除以" << b << "的商为：" << r << endl;
	}
	catch(int)
	{
		cout << "除数不能为 0，  请重新输入两个数：";
		cin >> a >> b;
		lnt r = divide(a,b);
		cout << a << "除以" << b << "的商为：" << r << endl;
	}
}

int main()
{
tryagain:
	try
	{
		f();
	}
	catch(int)
	{
		cout << "除数不能为 0，  请重新输入两个数：" << endl;
		goto tryagain;
	}
	return 0;
}
```