**概念题**

1. **标准输出流( std::cout )和标准错误输出流( std::cerr )两者有什么异同之处？什么时候常用标准输出流？什么时候常用标准错误输出流？**

答：

cout对应于标准输出流，默认情况下是显示器。这是一个被缓冲的输出，可以被重定向，优点是输出文字可以一次输出好多文字，不闪屏。

cerr对应标准错误流，用于显示错误消息。默认情况下被关联到标准输出流，但它不被缓冲，也就说错误消息可以直接发送到显示器，而无需等到缓冲区或者新的换行符时，才被显示。一般情况下不被重定向，优点是及时输出不被影响。

一般情况下输出信息时用用cout ，输出错误信息时用cerr。

1. **C中 scanf 和 printf 有什么不足的地方？C++中 std::cin 和 std::cout 是如何判断输入和输出对象类型的？**

答：

scanf 和 printf缺陷：类型不安全：不是强类型，不利于类型检查，会导致类型相关的运行错误！

std::cin 和 std::cout的优点：不需要额外指定数据的类型，类型由数据本身决定，避免了类型相关的错误！

C++在进行输入/输出时，首先创建某个I/O类的对象，然后，调用该对象类的成员函数进行基于字节流的输入/输出操作。

**三、**

答：

运行没问题，但是最后文件应该关闭一下

1. **什么是程序中的异常？程序中的异常和程序中的错误有什么区别？**

程序的错误通常包括：

语法错误：指程序的书写不符合语言的语法规则。这类错误可由编译程序发现。

逻辑错误（或语义错误）：指程序设计不当造成程序没有完成预期的功能。这类错误可通过对程序进行静态分析和动态测试发现。

运行异常：指程序设计对程序运行环境考虑不周而造成的程序运行错误。导致程序运行异常的情况是可以预料的，但它是无法避免的。为了保证程序的鲁棒性（Robustness），必须在程序中对可能的异常进行预见性处理。

1. **什么时候需要对异常采用就地处理策略？什么时候需要采用异地处理策略？**
2. 如果该该异常影响整个程序，或者是小范围影响，则需要就地处理。
3. 如果该异常影响调用者，则需要进行异地处理。

**六、假设有一个从异常基类派生来的异常类层次结构，则应按什么样的顺序放置catch块？并说明理由。**

答：应按从子孙到祖先的顺序排列catch语句块

**编程题**

**一、题目描述**

**编写一个程序，要求将两个文本文件的内容作为输入，创建一个新文本文件进行输出。该程序将两个输入文件中对应的行拼接起来，并用空格分隔，然后将结果写到输出文件中。如果两个输入文件行数不一致，则将较长文件的余下行直接复制到输出文件中。**

#include<iostream>

#include <fstream>

#include<string>

#include <vector>

using namespace std;

void read\_file(string name,vector<string>&s)

{

ifstream file;

file.open(name, ios::in);

if (!file.is\_open())

{

throw name;

}

else

{

while (!file.eof())

{

string str;

getline(file, str);

s.push\_back(str);

}

file.close();

}

}

void out\_put(string name,const vector<string>&s)

{

ofstream file;

file.open(name, ios::out);

if (!file.is\_open())

{

throw name;

}

else

{

for (size\_t i = 0; i < s.size(); i++)

file << s[i]<<endl;

}

}

int main()

{

vector<string> s1, s2;

try

{

read\_file("test1.txt",s1);

}

catch (string i)

{

cerr << "文件"<<i<<"打开失败" << endl;

}

try

{

read\_file("test2.txt", s2);

}

catch (string i)

{

cerr << "文件" << i << "打开失败" << endl;

}

//开始处理并保存

if (s1.size()>=s2.size())

{

for (size\_t k2 = 0; k2 < s2.size(); k2++)

{

s1[k2] = s1[k2] + " " + s2[k2];

}

try

{

out\_put("test3.txt",s1);

}

catch (string i)

{

cerr << "文件" << i << "打开失败" << endl;

}

}

else

{

for (size\_t k1 = 0; k1 < s1.size(); k1++)

{

s2[k1] = s2[k1] + " " + s1[k1];

}

try

{

out\_put("test3.txt", s2);

}

catch (string i)

{

cerr << "文件" << i << "打开失败" << endl;

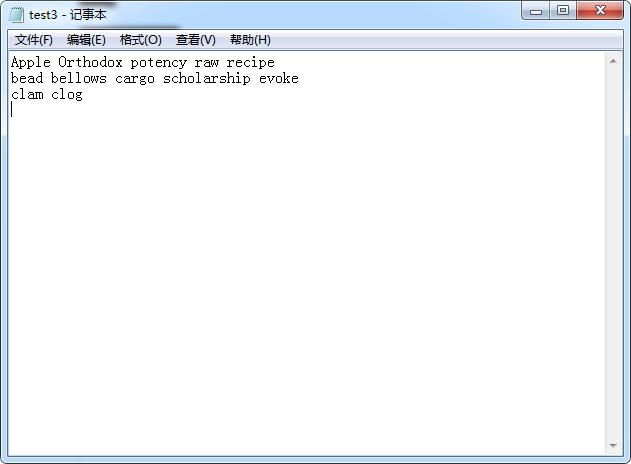
}

}

return 0;

}

**效果验证：**



**二、题目描述**

