概念题

一. 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？

- **同**：都是C++ I/O类库中预先定义的I/O对象，均可以直接用来在控制台中进行输出操作。都是`ostream`类的对象。通常默认的输出设备都是显示器。
- **异**：
  - `cout`通常用于计算机输出程序正常运行的结果。`cerr`通常用于计算机输出程序错误信息的结果。
  - `cout`经过缓冲后输出，而`cerr`输出无缓冲，立刻输出。
  - `cout`可进行输出重定向，`cerr`不受输出重定向的影响。
- **什么时候使用标准错误输出流**：当需要立刻输出信息或需要输出程序错误信息的时候。

二. 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？

- **不足**：类型不安全，不是强类型的，不利于类型检查，当格式串描述与数据类型不一致时可能会导致类型相关的运行错误。
- **如何判断对象类型**：通过重载`<<`，`>>`操作符及其函数名重载，将输入和输出对象作为参数，进而判断输入和输出的对象类型。

三.

- 应在`while`循环前先读入`str`，再判断`fail()`，否则当**文件末尾没有空行**时，会**多读入一个空行**，应如下所示：

```
string str;
file >> str;
while(!file.fail()){
    cout << str << endl;
    file >> str;
}
```

四. 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？

- **什么是异常**：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。
- **区别**：
  - 错误是指在环境正常的情况下，程序结构不正确。错误通常是**无法预料**的，但**可以避免**的。
  - 异常是指在环境不正常的情况下，结果不正确，往往是由于程序设计者对程序运行环境的一些特殊情况考虑不足所造成的。异常通常是**可以预料**的，但**无法避免**的。

五. 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？

- **就地处理策略**：当发生异常则不可以继续向后执行时，常采用就地策略，调用C++标准库中的函数exit或abort终止程序执行。
- **异地处理策略**：当发现异常时，在发现地（如在被调用的函数中）有时不知道如何处理这个异常，或者不能很好地处理这个异常，要常采用异地处理策略，由程序的其它地方（如函数的调用者）来处理。

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。

- **顺序**：先catch派生类的异常，再catch基类的异常。
- **理由**：派生类的异常和基类的异常都会被基类异常的catch块捕获。为了明确异常的具体类型，应先catch派生类的异常。

编程题

一.

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

#define print(x) cout << x << endl
#define input(x) cin >> x

int main(int argc, char const *argv[]){
    ifstream fin1, fin2;
    ofstream fout;
    fin1.open("test1.txt", ios::in);
    fin2.open("test2.txt", ios::in);
    fout.open("result.txt", ios::out);

    if(!fin1.is_open() || !fin2.is_open() || !fout.is_open()){
        cerr << "open error!\n";
        return 0;
    }

    string str1, str2, result;
    getline(fin1, str1);
    getline(fin2, str2);
    while (!fin1.fail() && !fin2.fail()){           //读入最小长度
        result = result + str1 + " " + str2 + "\n";
        getline(fin1, str1);
        getline(fin2, str2);
    }
    while (!fin1.fail()){                           //fin1可能未读完
        result = result + str1 + "\n";
        getline(fin1, str1);
    }
    while(!fin2.fail()){                            //fin2可能未读完
        result = result + str2 + "\n";
        getline(fin2, str2);
    }
```

```
    fin1.close();
    fin2.close();
    fout << result;
    fout.close();

    return 0;
}
```

二.

```
#include<iostream>
#include<fstream>
#include<string>
#include<string.h>
#include<algorithm>

using namespace std;

class MyException: public exception{            //继承exceptions类
private:
    string msg;
public:
    MyException(string m){
        msg = m;
    }
    ~MyException() throw (){}                   //throw()表示承诺函数不会抛出异常

    const char* what() const throw (){
        return msg.c_str();
    }
};


class User{
private:
    char uID[11]; // 用户ID
    char uPwd[16]; // 用户密码

public:
    User(){}
    void createUsers(const char* id, const char* pwd);      //创建用户
    void verifyUsers(const char* id, const char* pwd);      //验证用户
};

void User::createUsers(const char* id, const char* pwd){
    try{
        if(strlen(id) > 10){                                //ID长度不合法
            throw MyException("ID不合法，长度不合法!");
        }
        if((id[0] > 'Z' || id[0] < 'A') && id[0] != '_'){   //ID首字母不合法
            throw MyException("ID不合法，首字母不合法!");
        }
        if(strlen(pwd) >= 16 || strlen(pwd) < 6){           //密码长度不合法
            throw MyException("PWD不合法，长度不合法!");
        }
```

```
        string pwd1(pwd);
        string pwd2(pwd);
        string pwd3(pwd);
        transform(pwd1.begin(), pwd1.end(), pwd2.begin(), ::toupper);   //转为全
大写
        transform(pwd1.begin(), pwd1.end(), pwd3.begin(), ::tolower);   //转为全
小写
        if(pwd1 == pwd2 || pwd1 == pwd3){                               //PWD大小
写不合法
            throw MyException("PWD不合法，大小写不合法!");
        }
        strcpy(uID, id);
        strcpy(uPwd, pwd);
        cout << "ID, PWD均合法!" << endl;

        ofstream fout;
        fout.open("user.dat", ios::app | ios::out | ios::binary);
        if(!fout.is_open()){
            throw MyException("文件打开错误!");
            return;
        }
        fout.write((char*)this, sizeof(*this));
        cout << "用户存储成功!" << endl;
    }
    catch(const exception& e){
        cerr << e.what() << '\n';
    }

}

void User::verifyUsers(const char* id, const char* pwd){
    try{
        if(strlen(id) == 0){
            throw MyException("未输入用户名!");
        }else if(strlen(pwd) == 0){
            throw MyException("未输入密码!");
        }
        ifstream fin;
        fin.open("user.dat", ios::in | ios::binary);
        if(!fin.is_open()){
            throw MyException("文件打开错误!");
            return;
        }
        User user;
        fin.read((char*)&user, sizeof(user));

        bool idFlag = false;                                            //是否有该ID

        while (!fin.fail()){
            if(strcmp(id, user.uID) == 0 && strcmp(pwd, user.uPwd) == 0){
                cout << "验证通过!" << endl;
                fin.close();
                return;
            }else if(strcmp(id, user.uID) == 0){
                idFlag = true;
            }
            fin.read((char*)&user, sizeof(user));
        }
```

```
        if(idFlag){
            throw MyException("密码错误!");
        }else{
            throw MyException("无此用户!");
        }
    }catch(const exception& e){
        cerr << e.what() << '\n';
    }
    return;
}

int main(int argc, char const *argv[]){
    User user;
    user.createUsers("nancy", "123456Nancy");           //ID头部不对
    user.createUsers("DavidDDDDDDD", "Password111");    //ID长度不对
    user.createUsers("Jack", "qwe123ASD1111111111111111");  //密码长度不对
    user.createUsers("Poul", "poul123");                //密码不包含大小写

    user.createUsers("Nancy", "123456Nancy");
    user.createUsers("David", "Password111");
    user.createUsers("Jack", "qwe123ASD");
    user.createUsers("Poul", "Poul123");

    User user2;
    user2.verifyUsers("Nancy", "123456Nancy");      //通过
    user2.verifyUsers("David", "Password");         //密码错误
    user2.verifyUsers("Poul2", "Poul123");          //无此用户
    user2.verifyUsers("Mike", "");                  //未输入密码
    user2.verifyUsers("", "123");                   //未输入用户名
    return 0;
}
```