Assignment 13

191220022 丁一凡

一、 概念题

1.1 程序中的错误通常包括哪几种？它们分别是由什么原因造成的?

①语法错误，程序的书写不符合语言的语法规则

如 `x = y` (C程序语句末尾没有';')

②逻辑错误，程序的设计不当造成程序没有完成预期的功能

计算矩阵乘积时，写错下标

③运行异常，程序设计时对程序运行环境考虑不周而造成的程序运行错误

如内存不足

1.2 异常处理的两种策略是什么？它们分别是怎么做的？为什么不能在析构函数中调用exit?

两种策略分别是：

①异常的就地处理：

在发现错误的地方就地处理

②异常的异地处理：

发现异常的函数和处理异常的函数可以不是同一个函数

调用exit时，会进行关闭被程序调用全局对象和static存储类局部对象的析构函数，如果在这些对象的析构函数中调用exit会导致无限递归调用exit

1.3 如果不用C++的异常处理机制，应该如何处理在构造函数中发现的异常?

可以在类中添加对应的变量，在构造函数中对其设置对应的值，之后通过判断该变量的值发现异常并进行相应的处理

1.4 如果catch语句不能对异常完全处理，需要调用链中的上层函数进行处理应该怎么办？什么时候需要对catch中的异常对象声明为引用?

当内层的try语句的执行出现了异常，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捉该异常对象的catch语句，则逐步向外层进行查找。如果到了函数调用链的最顶端（函数main）也没有能捕捉该异常对象的catch语句，则调用系统的terminate函数进行标准异常处理

当需要对throw抛出的变量进行操作时，需要对catch中的异常对象声明为引用

二、 编程题

2.1

```
int main()
{
    double result = 0;
    bool flag = true;
    while(flag)
    {
        try
        {
            cout << "请输入两个数：  ";
            int a, b;
            cin >> a >> b;
            result = divide(a, b);
            cout << a << "除以" << b << "的商为：" << result;
            flag = false;
        }
        catch(int)
        {
            flag = true;
            cout << "除数不能为0！" << endl;
        }

    }
}
```

2.2

```
class Matrix
{
    vector<vector<int>> p_data; //表示矩阵数据
    int row, col; //表示矩阵的行数和列数
public:

    Matrix(int r, int c)    //构造函数
    {
        row = r;
        col = c;
        for (int i = 0; i < row; i++)
        {
            vector<int> temp(col);
```

```
            p_data.push_back(temp);
        }
    }

    ~Matrix() //析构函数
    {
        p_data.clear();
    }

    vector<int>& operator[] (int i) //重载[], 对于Matrix对象m, 能够通过m[i][j]访问第i+1行、第j+1列元素
    {
        return p_data[i];
    }
    Matrix operator + (const Matrix& m) const //重载+, 完成矩阵加法
    {
        if(this->row != m.row || this->col != m.col)
            throw 0;
        Matrix temp(row, col);
        for (int i = 0; i < row; i++)
        {
            for (int j = 0; j < col; j++)
            {
                temp[i][j] = p_data[i][j] + m.p_data[i][j];
            }
        }
        return temp;
    }
    Matrix operator * (const Matrix& m) const //重载*, 完成矩阵乘法
    {
        if(this->col != m.row)
            throw 0;
        Matrix temp(row, m.col);
        for (int i = 0; i < row; i++)
        {
            for (int j = 0; j < m.col; j++)
            {
                for (int k = 0; k < col; k++)
                {
                    temp.p_data[i][j] = p_data[i][k] * m.p_data[k][j] + temp.p_data[i][j];
                }
            }
        }
        return temp;
    }
};
void f()
{
    Matrix result;
    bool flag = true;
    while(flag)
    {
        try
        {
            cout << "请输入第一个矩阵的行、列： ";
            int a, b;
            cin >> a >> b;
```

```
            Matrix m1(a, b);
            cout << "请输入矩阵内容：";
            for(int i =0; i < a; i++)
            {
                for(int j = 0; j < b; j++)
                    cin >> m1[i][j];
            }
            cout << "请输入对应操作：";
            string oper;
            cin >> oper;
            cout << "请输入第二个矩阵的行、列：  ";
            cin >> a >> b;
            Matrix m2(a, b);
            cout << "请输入矩阵内容：";
            for(int i =0; i < a; i++)
            {
                for(int j = 0; j < b; j++)
                    cin >> m2[i][j];
            }
            if(oper == "+")
                result = m1 + m2;
            else
                result = m1 * m2;
            if(oper == "+")
            {
                cout << "结果为:" << endl;
                result.print();
                flag = false;
            }
            else
            {
                cout << "结果为:" << endl;
                result.print();
                flag = false;
            }
        }
        catch(int)
        {
            flag = true;
            cout << "矩阵输入不合法! " << endl;
        }

    }
}
```