学号 E21714049 专业 17 级计算机科学与技术 姓名 梅世祺
实验日期 2019.06.06 教师签字 成绩

实验报告

【实验名称】 异常处理

【实验目的】

1. 熟悉异常的使用场景，理解异常对于大型软件工程的作用及意义。
2. 掌握异常的处理流程。
3. 熟练运用 throw、try 和 catch 处理异常。
4. 掌握标准程序库异常处理。
5. 理解异常类继承与派生的作用。

【实验原理】

C++语言提供对处理异常的内部支持，try, throw 和 catch 语句就是 C++ 语言中用于实现异常处理的机制。有了 C++异常处理，程序可以向更高的执行上下文传递意想不到的事件，从而使程序能更好地从这些异常事件中恢复过来。

【实验内容】

实验一 日期类中的异常

题目：设计一个日期类 Date，包含 year，month，day 三个整型数据成员，实现构造函数、获取和修改年（月、日）的函数，并实现相关异常处理功能。

要求：1. 实现一个异常基类 DateException，包括错误码（code，整型）和错误信息（message，字符串型）两个数据成员和 getCode 及 getMessage 两个成员方法，分别获取错误码和错误信息。

2. 基于基类 DateException，派生 DateYearException，DateMonthException 和 DateDayException 三个子异常类，分别表示年、月、日异常。

3. 在主函数中进行测试，每个异常测试 3 组数据。

4. 注意：年和日的异常时相关的，比如闰年 2 月份的天数为 29 天，非闰年 2 月为 28 天。

原理：

提高(选做)：1.进一步派生 DateYearException(DateMonthException、DateDayException）类，分别细致处理不同类型的年（月、日）异常，例如月份小于零、大于十二等不同类型异常。

2.使用异常处理的方式改进《实验五 运算符重载》时间运算符重载的相关实现代码。

3. 使用异常处理的方式改进其他实验中的相关代码。

实验结果（含源码）：

DateException 基类声明如下：

```
class DateException {
    public:
        DateException(int code, string message) :
            code(code), message(message) {};

        int getCode();
        string getMessage();
    protected:
        int code;
        string message;
};
```

其派生类声明如下：

```
class DateYearException : public DateException {
    public:
        DateYearException(int code, string message)
            : DateException(code, message) {};
};
```

```
class DateMonthException : public DateException {
    public:
        DateMonthException(int code, string message)
            : DateException(code, message) {};
};
```

```
class DateDayException : public DateException {
    public:
        DateDayException(int code, string message)
            : DateException(code, message) {};
};
```

验证时间函数（validateTime）的具体实现如下：

```
bool validateTime(int year, int month, int day) {
    if (isLeapYear(year)) {
        monthMaxValue[2] = 29;
    }

    if (year < 0) {
        DateYearException dateYearException(100, "The year cannot be below zero!");
        throw dateYearException;
    }

    if (month < 1 || month > 12) {
        DateMonthException dateMonthException(200, "The month is out of range!");
        throw dateMonthException;
    }

    if (day < 1) {
        DateDayException dateDayException(300, "The day cannot be below one!");
        throw dateDayException;
    }

    if (day > monthMaxValue[month]) {
        DateDayException dateDayException(301, "The day is out of range!");
        throw dateDayException;
    }

    monthMaxValue[2] = 28;
    return true;
}
```

测试函数如下：

```
void testTime(int year, int month, int day) {
    try {
        cout<<"\033[34mTesting: \033[0m"<<" "<<year<<":"<<month<<":"<<day;
        validateTime(year, month, day);
        cout<<" \033[33mSuccess\033[0m"<<endl;
    } catch (DateYearException exception) {
        cout<<endl<<"\033[31mDateYearException: \033[0m"<<" "<<exception.getMessage()<<" ("<<exception.getCode()<<")"<<endl;
    } catch (DateMonthException exception) {
        cout<<endl<<"\033[31mDateMonthException: \033[0m"<<" "<<exception.getMessage()<<" ("<<exception.getCode()<<")"<<endl;
    } catch (DateDayException exception) {
        cout<<endl<<"\033[31mDateDayException: \033[0m"<<" "<<exception.getMessage()<<" ("<<exception.getCode()<<")"<<endl;
    }
}
```

测试用例如下：

```
testTime(-9, 1, 1);
testTime(0, 1, 1);
testTime(1, 1, 1);

testTime(2010, -9, 28);
testTime(2010, 13, 28);
testTime(2010, 12, 28);

testTime(2001, 2, 29);
testTime(2000, 2, 29);
testTime(2000, 5, -1);
```

```
----------  构建用时:398 ms ---------------
Testing:   -9:1:1
DateYearException:  The year cannot be below zero! (100)
Testing:   0:1:1 Success
Testing:   1:1:1 Success
Testing:   2010:-9:28
DateMonthException:  The month is out of range! (200)
Testing:   2010:13:28
DateMonthException:  The month is out of range! (200)
Testing:   2010:12:28 Success
Testing:   2001:2:29
DateDayException:  The day is out of range! (301)
Testing:   2000:2:29 Success
Testing:   2000:5:-1
DateDayException:  The day cannot be below one! (300)
➜
```

运行结果