面向对象程序设计项目报告

分数计算器系统的设计与实现

姓　　名：乔翱

课　程　名　称：C/C++程序设计

指　导　教　师：董俊

2019 年 5 月

分数计算器设计

乔翱

（燕山大学 信息科学与工程学院）

摘 要： 本报告具体、详细地记述了我在做此三级项目——分数计算器的整个心路过程，此项目报告具体包括四个方面，分别是对项目的分析、项目设计、测试、以及结论。具体介绍了我在做这个项目前的分析，我的设计思路，以及完成后的反思和改进等方面。这个报告反映了这个分数计算器项目从最初的雏形到完善的一个过程。

1 项目分析

1.1 用户需求分析

分数计算器是为了实现对分数的一系列操作，用户可以通过菜单选择执行的功能，实现对分数的一系列操作，包括对分数的加、减、乘、除运算，实现对分数的化简运算，实现对分数的乘方运算等，用户还可以查看过去的计算记录（只有本次操作的历史计算记录）。

设计此分数计算器就是为了方便用户可以实现对分数的一系列操作，方便用户进行分数的计算等对分数的操作，由于此程序是用控制台做的，所以没有优美的界面，但是也可以极大的满足客户的需求，用户可以通过输入所想要执行功能前的序号，去执行某项功能，用户使用起来十分的方便。

1.2 系统设计思路

整体的思路是把这个系统分成两个类，然后具体实现其中的各个功能，一步步先实现各个小功能，最后合起来实现整个分数计算器的全部功能。

具体步骤如下：

1、 先思考需要两个类，一个整数类，一个分数类，分数类继承整数类；

2、 思考分数的加减乘除算法，如何可以实现分数之间的运算；

3、 思考运算符重载怎么样实现，包括流提取和流插入的运算符重载，以及+、-、*、/的运算符重载；

4、 通过文件读写功能实现历史纪录功能；

5、 通过写代码实现各个功能，美化各个界面，使用户可以更方便的操作，使用户可以有更好的体验；

6、 最后完成后，找人体验这个系统，收集用户的使用意见，去完善并且改进这个系统。

1.3 系统模块划分

该系统设计两个类，一个整数类，一个分数类，分数类继承整数类；

1、 项目设计输入输出功能：在整数类中对流提取和流插入运算符进行重载实现分数的输入和输出；

2、 计算功能：设计成员函数分别实现分数的加减乘除，并且对相应的运算符+-*/进行重载；

3、 其他的对分数的操作功能：通过设计成员函数进行函数的化简，也就是对一个函数的约分，由于函数的分母不能为零；

4、 异常处理功能：要注意设计异常处理功能，不能让分数的分母为零，并且设计成员函数实现分数的乘方运算；

5、 历史纪录功能：以及通过文件读写实现对分数计算历史记录的写入以及读取；

6、 主菜单：在主界面要有一个菜单，实现用户可以通过功能前的编号实现对该功能的执行。

2 项目设计

2.1 各个子模块的设计方法

主要实现分数类的成员函数；分数类的核心代码如下：

```
class Fractions:public Integer{
public:
```

```
    Fractions(int a=0,int b=1){above=a;below=b;}
    ~Fractions(){}
     friend ostream & operator<<(ostream &out,const Fractions
&f);
     friend istream & operator>>(istream &input, Fractions &f);
    Fractions add(Fractions);//加法计算
    Fractions subtract(Fractions);//减法计算
    Fractions multiply(Fractions);//乘法计算
    Fractions divide(Fractions);//除法计算
    Fractions power(int);//乘方计算
    Fractions Fractions::operator +( Fractions &f);
    Fractions Fractions::operator -( Fractions &f);
    Fractions Fractions::operator *( Fractions &f);
    Fractions Fractions::operator /( Fractions &f);
    void reduction();//约分
    void makeCommond(Fractions&);//通分
    int getAbove(){return above;}
    int getBelow(){return below;}
   private:
    int above;
    int below;
   };
```

通过实现其中成员函数的功能，实现对分数的一系列操作，完成各个子模块的设计，最终实现整个分数计算器的全部功能。

2.2 设计核心技术

1、分数的加减乘除：

通过对分数分子和分母的处理，对分数进行通分和约分，再进行一系列计算操作，核心代码如下：

约分：

```
void  Fractions::reduction(){
int a,b,temp,divisor;
a=abs(above);
b=abs(below);
while(a%b){//辗转相除法求最大公约数
 temp=a;
 a=b;
 b=temp%b;
}
divisor=b;
above/=divisor;
below/=divisor;
}
```

通分：

```
void Fractions::makeCommond( Fractions &f){//通分
int temp;
reduction();
f.reduction();
above*=f.below;
f.above*=below;
temp=below*f.below;
below=f.below=temp;
```

```
}
```

加法运算：

```
Fractions Fractions::add(Fractions f){
Fractions p;
makeCommond(f);
p.above=above+f.above;
p.below=below;
p.reduction();
return p;
}
```

减法运算：

```
Fractions Fractions::subtract(Fractions f){
 Fractions p;
makeCommond(f);
p.above=above-f.above;
p.below=below;
p.reduction();
return p;
}
```

乘法运算：

```
Fractions Fractions::multiply(Fractions f){
 Fractions p;
makeCommond(f);
p.above=above*f.above;
p.below=below*f.below;
p.reduction();
return p;
```

```
}
```

除法运算：

```
Fractions Fractions::divide(Fractions f){
 Fractions p;
makeCommond(f);
p.above=above*f.below;
p.below=below*f.above;
p.reduction();
return p;
}
```

2、运算符的重载

包括对流提取和流插入运算符重载实现分数的输入和输出功能，以及对+、-、*、/运算符的重载，核心代码如下：

输出，对流提取运算符的重载：

```
ostream & operator<<(ostream &out,const Fractions &f){//输出，对流提取运算符的重载
 if(f.below==1)
        out<<f.above<<endl;
 else
 out<<f.above<<"/"<<f.below;
 return out;
}
```

输入，对流插入运算符的重载：

```
istream & operator>>(istream & input, Fractions &f){//输入，对流插入运算符的重载
 cout<<"请依次输入分数的分母和分子，以空格隔开（eg：1 2）";
 input>>f.above;
        input>>f.below;
```

```
        if(f.below==0)
                throw f;
 return input;
}
```

对+运算符的重载：

```
Fractions Fractions::operator +( Fractions &f){
Fractions p;
makeCommond(f);
p.above=above+f.above;
p.below=below;
p.reduction();
return p;
};
```

对-运算符的重载：

```
Fractions Fractions::operator -( Fractions &f){
 Fractions p;
makeCommond(f);
p.above=above-f.above;
p.below=below;
p.reduction();
return p;
};
```

对*运算符的重载：

```
Fractions Fractions::operator *( Fractions &f){
 Fractions p;
makeCommond(f);
p.above=above*f.above;
```

```
p.below=below*f.below;
p.reduction();
return p;
}
```

对/运算符的重载：

```
Fractions Fractions::operator /( Fractions &f){
Fractions p;
makeCommond(f);
p.above=above*f.below;
p.below=below*f.above;
p.reduction();
return p;
}
```

2、异常处理

若分数的分母输入为零，进行异常处理功能，核心代码如下：

```
if(f.below==0)
                throw f;
try{cin>>f1;}
        catch(Fractions &p){
                cout<<"分数的分母不能为 "<<p.getBelow()<<" 请重新输入。"<<endl;
                cin>>f1;
                }
```

3、历史记录

利用 C++中的文件读写操作实现历史计算记录的写入与读取，核心代码如下：

历史计算记录的写入：

```
ofstream outFile;
outFile.open("C:\\Users\\qa\\Desktop\\C++\\Fractions.txt",ios::app);
```

```
outFile <<f1<<x<<f2<<"="<<f3<<endl;
        outFile.close();
```

历史记录的读取：

```
cout<<"欢迎您查看历史记录（本历史记录只包含当前操作的计算记录）
"<<endl;
        string str;
        ifstream fin;
        fin.open("C:\\Users\\qa\\Desktop\\C++\\Fractions.txt",ios::in
) ;
        stringstream buf;
        buf<<fin.rdbuf();
        str=buf.str();
        cout<<str<<endl;
        fin.close();
```

历史记录的清除：

```
void cleartxt(string filename)
{
ofstream text;
text.open(filename,ios::out);
text.close();
}
```

截图：

4、主菜单

通过输入对应功能前的数字，实现对应功能,核心代码如下：

```
void Menu(int a){
        if(a==1){
                string p,a,b,c,d;
Fractions f1;Fractions f2;Fractions f3;
char x;
try{cin>>f1;}
                catch(Fractions &p){
```

```
                    cout<<"分数的分母不能为 "<<p.getBelow()<<" 请重
新输入。"<<endl;
                cin>>f1;
                }
                inttostr(f1.getAbove(),a);
inttostr(f1.getBelow(),b);
p+=a+"/"+b;
cout<<"请输入运算符号：";
cin>>x;
while(x!='=')
{
try{cin>>f2;}
                catch(Fractions &p){
                    cout<<"分数的分母不能为 "<<p.getBelow()<<" 请重
新输入。"<<endl;
                cin>>f2;
                }
if(x=='+')
{
f3=f2+f1;
f1.reduction();
f2.reduction();
inttostr(f2.getAbove(),c);
inttostr(f2.getBelow(),d);
p+="+"+c+"/"+d;
}
else if(x=='-')
{
f3=f1-f2;
f1.reduction();
f2.reduction();
inttostr(f2.getAbove(),c);
inttostr(f2.getBelow(),d);
p+="-"+c+"/"+d;
}if(x=='*')
{
f3=f2*f1;
f1.reduction();
f2.reduction();
inttostr(f2.getAbove(),c);
inttostr(f2.getBelow(),d);
```

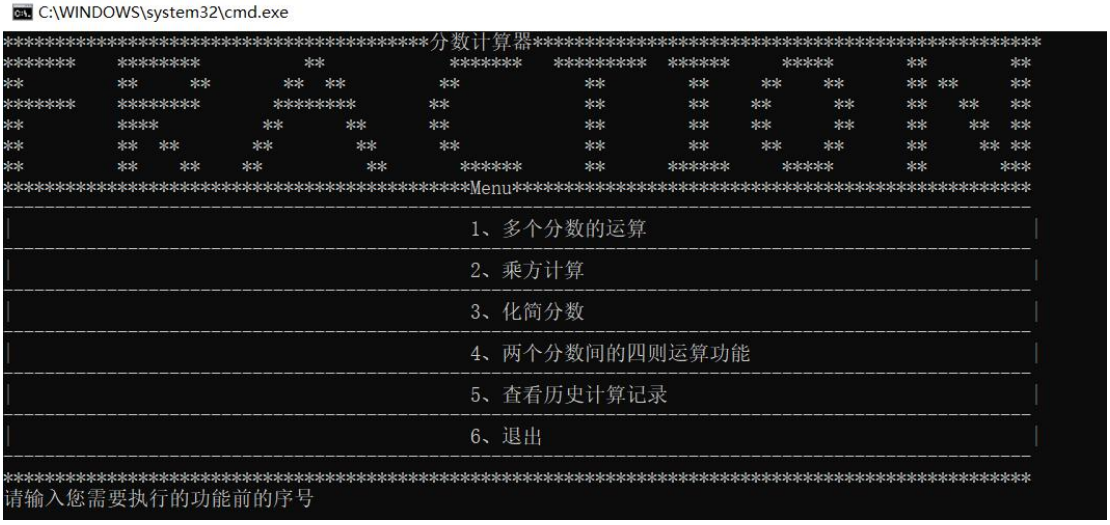
```
p+="*"+c+"/"+d;
}if(x=='/')
{
f3=f2+f1;
f1.reduction();
f2.reduction();
inttostr(f2.getAbove(),c);
inttostr(f2.getBelow(),d);
p+="/"+c+"/"+d;
}
        cout<<"请输入运算符号：";
cin>>x;
f1=f3;
}
ofstream outFile;
outFile.open("C:\\Users\\qa\\Desktop\\C++\\Fractions.txt",ios::app);
outFile <<p<<"="<<f3<<endl;
        outFile.close();
cout<<p<<"="<<f3<<endl;
        }
        else if(a==2){
                Fractions f1;
                Fractions f2;
                int a;
                cout<<"请输入需要乘方的分数：  ";
                try{cin>>f1;}
                catch(Fractions &p){
                        cout<<"分数的分母不能为 "<<p.getBelow()<<" 请重
新输入。"<<endl;
                cin>>f1;
                }
                cout<<"请输入幂：";cin>>a;
                cout<<"乘方后的分数为 ： "<<f1.power(a)<<endl;
                ofstream outFile;
outFile.open("C:\\Users\\qa\\Desktop\\C++\\Fractions.txt",ios::app);
outFile <<f1<<"的"<<a<<"次方为 "<<f1.power(a)<<endl;
        outFile.close();
        }
        else if(a==3){
        Fractions f1;
        Fractions f2;
```

```
        cout<<"请输入需要化简的分数： ";
        try{cin>>f1;}
                catch(Fractions &p){
                        cout<<"分数的分母不能为 "<<p.getBelow()<<" 请重
新输入。"<<endl;
                cin>>f1;
                }
        f2=f1;
        f1.reduction();
        if(f2.getAbove()==f1.getAbove()&&f2.getBelow()==f1.getBelow())
{
        cout<<"您输入的分数已经是最简的了，无需化简。"<<endl;
        }
        else{
        cout<<f2<<"化简后为 "<<f1<<endl;
        }
        ofstream outFile;
outFile.open("C:\\Users\\qa\\Desktop\\C++\\Fractions.txt",ios::app);
outFile <<f1<<"化简为"<<f2<<endl;
        outFile.close();
        }
        else if(a==4){
Fractions f1;Fractions f2;Fractions f3;
char x;
try{cin>>f1;}
                catch(Fractions &p){
                        cout<<"分数的分母不能为 "<<p.getBelow()<<" 请重
新输入。"<<endl;
                cin>>f1;
                }
cout<<"请输入运算符号：";
cin>>x;
try{cin>>f2;}
                catch(Fractions &p){
                        cout<<"分数的分母不能为 "<<p.getBelow()<<" 请重
新输入。"<<endl;
                cin>>f2;
                }
if(x=='+')
{
f3=f2+f1;
```

```
f1.reduction();
f2.reduction();
cout<<f1<<"+"<<f2<<"="<<f3<<endl;
}
else if(x=='-')
{
f3=f1-f2;
f1.reduction();
f2.reduction();
cout<<f1<<"-"<<f2<<"="<<f3<<endl;
}if(x=='*')
{
f3=f2*f1;
f1.reduction();
f2.reduction();
cout<<f1<<"*"<<f2<<"="<<f3<<endl;
}if(x=='/')
{
f3=f2+f1;
f1.reduction();
f2.reduction();
cout<<f1<<"/"<<f2<<"="<<f3<<endl;
}
ofstream outFile;
outFile.open("C:\\Users\\qa\\Desktop\\C++\\Fractions.txt",ios::app);
outFile <<f1<<x<<f2<<"="<<f3<<endl;
       outFile.close();
       }
       else if(a==5)
       {
       cout<<"欢迎您查看历史记录（本历史记录只包含当前操作的计算记
录）"<<endl;
       string str;
       ifstream fin;
       fin.open("C:\\Users\\qa\\Desktop\\C++\\Fractions.txt",ios::in)
;
       stringstream buf;
       buf<<fin.rdbuf();
       str=buf.str();
       cout<<str<<endl;
       fin.close();
```

```
    }
    }//菜单功能
```

截图：



3 项目测试

通过多次测试，发现游戏存在的问题，解决问题，改进程序。

测试要点：

1. 测试是否可以通过在主菜单操作跳转，方便的实现相应的功能，并且测试是否能结束这个系统的使用

2. 重点测试是否能实现分数的四则运算，运算结果是否正确，可以用手机计算器检验运算结果是否正确；

3. 测试文件读写功能是否正常，是否可以读取并写入，正常显示历史计算，给用户一种直观的历史计算记录的呈。

4. 测试当输入分母为零时，是否会提示错误，异常处理功能是否可以正常运行。

4 结论

4.1 心得感受

在整个做三级项目的过程中，遇到很多问题，我于是就去一点一点地学习，去问同学，问老师，上网查资料，自己去钻研，解决一系列问题。在整个做分数计算器系统的过程中，我学习到了很多实用又新型的知识，让我了解到我们大学生的知识面不应局限于课堂，我们应该充分利用课余时间广泛涉猎，力争做一名新世纪的复合型人才。

最后到真正做好这个分数计算器的时候，有着很大的成就感，想想其实也没有那么难做。通过做这个分数计算器系统，我收获颇丰，我开阔了眼界，勇敢的接受了挑战，我更清楚的体会到不断充实自己的知识是多么重要，我们不能局限于自己的专业知识，还应该广泛涉猎。通过实践，我也锻炼了自己行动实施的能力，提高了自己的代码水平。

通过做三级项目发现其实没有什么是特别困难的，只要你努力去做，就一定能做好。我开始做的时候也是毫无头绪，但是通过自己一点点的学习，一步一步把三级项目做好。

通过做三级项目，我更加喜欢代码，提升了对代码的兴趣，提升了对编程的兴趣。

4.2 主要发现

在做这个分数计算器系统时，发现自己还有很多不足的地方，不是上课听懂了，你就能做出来一个程序，必须要加强练习，加强动手能力，自己多去打代码，多去实践，才能提高自己的能力，在接下来的学习生活中，要去多学多想，充实自己，提高自己的各方面能力，遇到不会的要多问多学，只有自己去主动地学习，才能学到知识，才能提升自己的水平

4.3 下一步开展的工作

再次研读C++课本，每天抽时间打一打代码，只有你坚持去做，多练习，最后才能做好。只有你自己去实践操作了，你才能发现你自己的不足，发现你自己的问题，发现问题后要及时解决，遇到不会的问题不跳过，去网上搜一搜，去书上找一找，去问问老师。弥补自己的不足和漏洞，并且去完善自己做的分数计算器的一些功能，改进自己设计的程序中存在的问题，进一步完善自己设计的分数计算器系统。

参考文献：

[1] 郑莉 董渊 何江舟 编 《C++语言程序设计》，清华大学出版社，2010.