**C++程序设计大型实验**

**实验题目：大整数运算·字串对象版**

**姓名：叶璇**

**学号：201806060825**

**班级：健行1802班**

**时间：2019年6月10日**

1. **实验题目**

本实验为大整数运算·字串对象版

基本描述

有一些整数，其位可能上百位。

在功能上要对这些数做加、减、乘、整除以及取余运算。

在程序组织上要求通过大整数类型，来体现更好的可读性和可维护性，并且对错误采用异常方法处理。

输入描述

输入数据有若干组数据，每组数据由一个运算符（+、-、\*、/、%）和两个整数构成。整数的范围在-10150～10150之间，如果运算符遇到‘@’，则表示运行结束。

输出描述

对于每组数据，输出其运算序号和运算结果，其中的‘/’位整除运算。每个结果单独成行。

如果运算结果超过200位，则应输出“Too Large Number.”；

如果输入整数为空，或者有前导0，则应输出“Illegal Number”；

如果输入符号不是上述的五种运算符之一，则应输出”Illegal Operator.”，并在输入操作上，应跳过后续的两个操作数。

如果除0，则应输出“Divide By Zero.”。

样本输入

/

100 0

/

19 6

\*

12 21

@

样本输出

1 Divide By Zero.

2 3

3 252

1. 运行环境

软件环境

编译器：GCC-6.3.0-1

编辑器：Sublime Text 3.2.1(Build 3207)

硬件环境

电脑：联想ThinkPad S1 2017

CPU：第七代智能英特尔®酷睿™i5-7200U处理器(2.5GHz睿频至3.1GHz,3MB)

操作系统：Windows 10 家庭版

1. 实验课题分析

需求分析

1. 开始进去需要告知能进行什么运算。有一个于菜单的页面。
2. 所做的运算能够处理大数（一般指位数在1000以上）的数据的运算。包含加、减、乘、除和取模的操作。
3. 能够对不同的错误操作进行不同的错误响应。
4. 由两部分组成：需要进行的操作和数据。由用户输入+、-、\*、/、%中的任一一项确定要进行的操作。再由用户输入两个数据，确认需要操作的数据。

结构功能分析

需要一个用于储存符号和数据的类和一个用于处理异常的抽象类。抽象类有4个儿子来处理不同情况的异常。

1. 实验的主要模块、流程图

头文件设计

类strInt：作为处理的载体，储存符号和数据。

//strint.h

#ifndef \_HEADER\_STRINT\_

#define \_HEADER\_STRINT\_

#include<iostream>

using std::string;

class strInt{

private:

string sign;

string num;

void swap(int& x,int& y);

public:

strInt();//默认构造函数值为0

strInt(const string&);

friend strInt mul(const strInt& a,const strInt& b);

friend strInt add(const strInt& a,const strInt& b);

friend strInt sub(const strInt& a,const strInt& b);

friend strInt div(const strInt& a,const strInt& b);

friend strInt mod(const strInt& a,const strInt& b);

string display();

};

#endif

类unexpect：处理异常的抽象类。

类unexpectedNull：用于处理非法数字。

类unexpectedZero：用于处理除数为0的情况。

类unexpectedLarge：用于处理太大的数据。

类unexpectedOp：用于处理奇怪的操作符。

//unexpected.h

#ifndef \_HEADER\_UNEXPECTED\_

#define \_HEADER\_UNEXPECTED\_

#include<string>

using std::string;

class unexpect{

public:

virtual string getWhat()=0;

};

class unexpectedNull:public unexpect{//用于处理非法数字。

public:

string getWhat(){return "Illegal Number.";}

};

class unexpectedZero:public unexpect{//用于处理除数为0的情况。

public:

string getWhat(){return "Divide By Zero.";}

};

class unexpectedLarge:public unexpect{//用于处理太大的数据。

public:

string getWhat(){return "Too Large Number.";}

};

class unexpectedOp:public unexpect{//用于处理奇怪的操作符。

public:

string getWhat(){return "Illegal Operator.";}

};

#endif

功能细节设计

计算模块的功能细节由strint.cpp来实现

1. adding，subing函数作为通用的字符串加减函数存在，也能够在其它所需要的地方调用。
2. add,sub,mul,div,mod作为类的接口，利用友元的特性使得其能在main函数中直接调用。
3. display函数用于返回数值本身（符号+其绝对值）。

//strint.cpp

#include"strint.h"

#include"unexpected.h"

using namespace std;

#include<iostream>

const int BITNUM = 200;

bool cmp(const string& a,const string& b){

return a.length()<b.length()||a.length()==b.length()&& a<b;

}

string adding( const string& a, const string& b ){

int len = (a.length()<b.length() ? b.length() : a.length())+1;

string s = string(len-a.length(), '0')+a;

int tmp = 0;

for(int i=1, bi=b.length()-1, si=s.length()-1; i<=b.length()||tmp; ++i, si--){

if(i<=b.length()) tmp += s[si]-'0' + b[bi--]-'0';

else tmp += s[si]-'0';

s[si] = char(tmp%10+'0');

tmp /= 10;

}

int pos = s.find\_first\_not\_of('0');

return (pos!=string::npos ? s.substr(pos) : s);

}

string subing(const string& a,const string& b){

if(a==b) return "0";

string s(a);

for(int i=1, bi=b.length()-i,si=s.length()-i; i<=b.length(); ++i,si--,bi--)

if((s[si]-=(b[bi]-'0'))<'0')

s[si]+=10, s[si-1]--;

int pos = s.find\_first\_not\_of('0');

return (pos!=string::npos ? s.substr(pos) : s);

}

strInt::strInt(){

num="0";

}

strInt::strInt(const string& a){

if(a.length()==0 || (a[0]=='0'&& a.length()>1)){

throw unexpectedNull();

}

if(a[0]=='-'){ sign="-"; num=a.substr(1); }

else num = a;

}

strInt add( const strInt& a, const strInt& b ){

if(a.num=="0") return b;

if(b.num=="0") return a;

strInt s;

if(a.sign==b.sign){

s.sign = a.sign;

s.num = adding(a.num, b.num);

}

else if( cmp(a.num, b.num) ){

s.sign = b.sign;

s.num = subing(b.num, a.num);

}

else{

s.sign = a.sign;

s.num = subing(a.num, b.num);

}

if( s.num.length() > BITNUM ) throw unexpectedLarge();

return s;

}

strInt sub( const strInt& a, const strInt& b ){

strInt s(b);

s.sign = (s.sign=="-" ? "":"-");

return add( a, s );

}

strInt mul( const strInt& a, const strInt& b ){

if(a.num=="0" || b.num=="0") return string("0");

strInt ss;

ss.sign = (a.sign==b.sign ? "" : "-");

string s(a.num.length()+b.num.length(), '0');

for(int bi=b.num.length()-1; bi>=0; --bi){

if(b.num[bi]=='0') continue;

for(int t=0, ai=a.num.length()-1, si=s.length()-b.num.length()+bi; ai>=0; --ai){

t += (a.num[ai]-'0')\*(b.num[bi]-'0') + s[si]-'0';

s[si--] = char(t%10+'0');

t/=10;

}

}

int pos = s.find\_first\_not\_of('0');

ss.num = (pos!=string::npos ? s.substr(pos) : s);

if( ss.num.length() > BITNUM ) throw unexpectedLarge();

return ss;

}

strInt div( const strInt& a, const strInt& b ){

if(b.num=="0") throw unexpectedZero();

if( cmp(a.num, b.num) ) return string("0");

strInt ss;

ss.sign = (a.sign==b.sign ? "" : "-");

string s, y(a.num.substr(0, b.num.length()-1));

for(int i=b.num.length()-1; i<a.num.length(); ++i){

int cnt=0;

for(y=(y=="0"? string(1,a.num[i]): y+a.num[i]); !cmp( y, b.num ); cnt++)

y = subing( y, b.num );

s += char(cnt+'0');

}

int pos = s.find\_first\_not\_of('0');

ss.num = (pos!=string::npos ? s.substr(pos) : s);

return ss;

}

strInt mod( const strInt& a, const strInt& b ){

if(b.num=="0") throw unexpectedZero();

if(a.num=="0" || b.num=="1" || a.num == b.num) return string("0");

if( cmp( a.num, b.num ) ) return a;

strInt ss;

ss.sign = a.sign;

string s, y(a.num.substr(0, b.num.length()-1));

for(int i=b.num.length()-1; i<a.num.length(); ++i){

int cnt=0;

for(y=(y=="0"? string(1,a.num[i]): y+a.num[i]); !cmp( y, b.num ); cnt++)

y = subing(y, b.num );

s += char(cnt+'0');

}

ss.num = y;

return ss;

}

string strInt::display(){

return sign+num;

}

控制模块由main.cpp实现

1. main函数先读入一个字符ch，判断是否是结束符号和加减乘除，是否合法。再输入两个字符串，通过两个字符串创建两个strInt实例，通过操作strInt实例来进行运算。
2. 若有异常则立刻抛出提示错误操作的类。

//main.cpp

#include"strint.h"

#include"unexpected.h"

#include"iostream"

#include<fstream>

using namespace std;

int main(){

ifstream cin("in.txt");//输入文件

ofstream cout("out.txt");//输出文件

char ch;string s1,s2;

int cnt=1;

while(cin>>ch&&ch!='@'){

try{

if(!(ch=='+'||ch=='-'||ch=='\*'||ch=='/'||ch=='%'))

throw unexpectedOp();

else{

cin>>s1>>s2;

strInt a(s1);strInt b(s2);

cout<<cnt++<<" ";

switch(ch){

case'+':cout<<add(a,b).display();break;

case'-':cout<<sub(a,b).display();break;

case'\*':cout<<mul(a,b).display();break;

case'/':cout<<div(a,b).display();break;

case'%':cout<<mod(a,b).display();break;

}

cout<<endl;//保证每行换行

}

}

catch(unexpect& n){//处理异常

cout<<n.getWhat()<<endl;

}

}

}

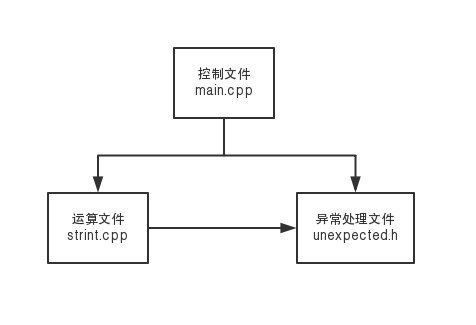
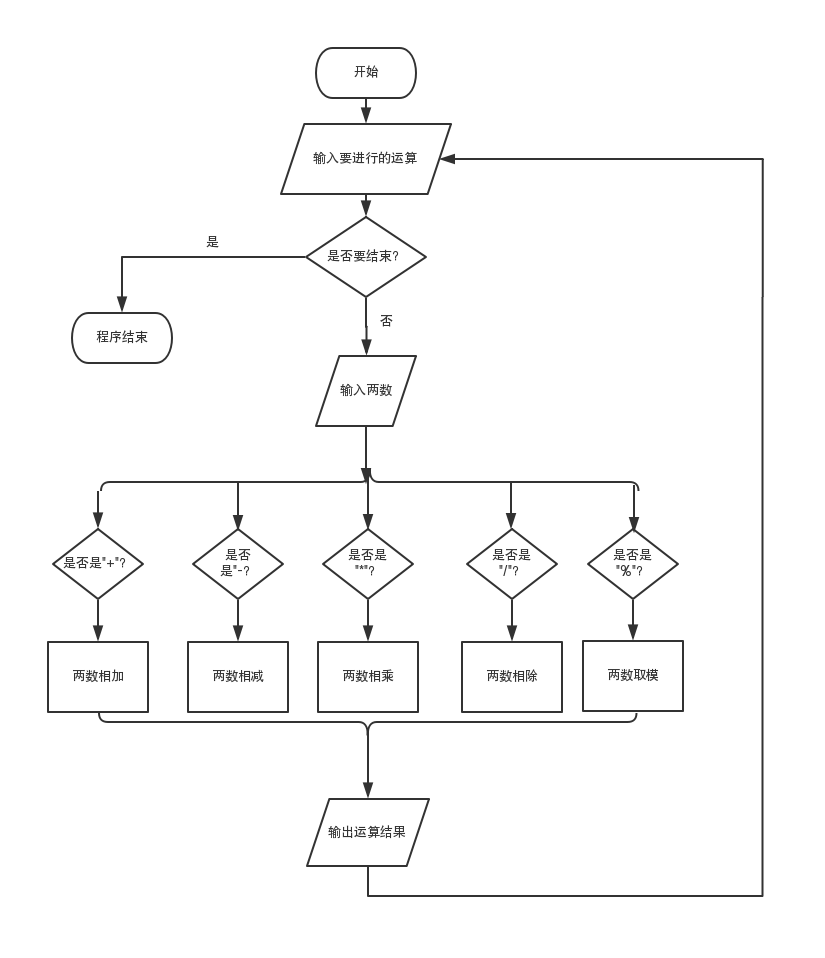


图 1工程文件组织



控制模块

运算模块

图 2流程图

1. 实验调试、测试、运行记录及分析

测试数据生成

测试数据生成代码由教师给出。

//creatnum.cpp

//=====================================

// 大数运算\_数据生成

//=====================================

#include<fstream>

#include<random>

#include<iostream>

#include<stdlib.h>

#include<cmath>

#define random(x) (rand()%x)

using namespace std;

//-------------------------------------

char op[5]={'+', '-', '\*', '/', '%'};

//-------------------------------------

bool printNum(ostream& cout, int a){

cout<<a;

if(a){

int len = random(100);

for(int i=1; i<=len; ++i)

cout<<random(10);

}

cout<<"\n";

}//------------------------------------

int main(){

ofstream cout("in.txt");

for(int i=1; i<=1000; ++i){

cout<<op[random(5)]<<"\n";

printNum(cout, random(10));

printNum(cout, random(10));

}

cout<<"@\n";

}//====================================

运行记录

生成的测试结果和运行结果详见”in.txt”和”out.txt“文件。

分析

生成的第二组样例：

+

0

63

通过人工计算得63+0=63，正确。

1. 实验总结

优点

面向对象的大数加减运算比起面向过程的大数加减运算可扩展性强，框架架构强。面向对象的大数加减运算是基于面向过程的大数加减运算的，其封装的特性也保证了数据的安全。且面向对象的大数加减增加了异常处理，能够更加优化程序。

不足

面向对象的大数加减需要调试更多，多文件操作有时候会出现未知BUG。且面向对象的大数加减对代码能力要求更高，调试所需要时间更长。

收获及体会

通过本次实验学习了运算C++中运算大数的方法，作为实现常见的底层语言，C/C++的运行速度较快是优点，而其不能对大数进行运算也一直让人诟病，以至于出现了acm中常见知识点：大数。

通过本次实验增强了对面向对象程序设计的理解，也增强了自我调试发现问题和DEBUG的能力，也增强了求助于网络的自我解决问题的能力。

本次实验在进行算法设计上有些不足，有寻求帮助。也发现了自己在算法设计上的短板。

对课程的看法与体会

课程刚开始有介绍面向过程的计算方法，使得在进行面向对象迁移程序时能够更加顺利，由浅入深，知识迁移的方法使得理解起来更加顺利。

第三章 对象化方法

1. **系统模型和类设计**
2. 程序类设计：对象化方法，此时需要设计多个类。首先的是程序类。类中有一个表达式成员，同时有一个不断运行处理数据的run()函数。Run()函数的实现与过程化方法类似。
3. 表达式类设计：将过程化的三个语法单元函数封装在一个表达式类中。表达式类中有单词成员数据，包含三个词法单元函数。同时增加一个getToken()函数，来读取tok,进一步判断处理数据。
4. 单词类设计：在单词类设计中，封装了单词的各类信息，包括数据，标志，等。同时设计了一个初始化函数，有一个readToken()函数，判断单词的类型。
5. 异常类设计：同过程化方法的异常类一样，在此不再赘述，详情请参考上文。
6. **代码运行测试过程**
7. 对象化方法同过程化方法在运行结果方面没有差异，相同的输入数据，输出的数据一样，就是实现的过程不一样，一个是一步一步按过程进行，另一个是将过程，数据封装在类里面，让类与类打交道。编译过程中也没用遇到什么问题。

第四章 对象化系统扩充

1. **系统修改方案**
2. 主程序框架：大整数计算器的设计，主要是将long double换成StrInt 大整数类型进行计算，其余代码不用做太大的修改。
3. 大整数类设计：此部分主要是进行StrInt 大整数类的设计。类中包含大整数类的符号，数值，位数限制，以及一些运算符的重载。详细代码见本部分实验的文件清单。
4. **代码运行测试过程**
5. 由于是前面2个实验代码的整理与小幅度修改，编译过程并么有遇到问题，而且程序运行的结果也与预期的一样，整体来说，实验还算成功。

附录

注：实验全部的源代码在每个文件夹下。输入输出数据也在各个.txt文件中。