题9.5

**Complex class**

题目要求：

建立一个复数类，要求含有两个数据成员，包括复数的实部与虚部，要求有默认参数的构造函数，同时有若干个成员函数包括以下功能：复数相加，复数相减，打印复数。

解题思路：

类中存储两个数据成员（Real，Imag分别表示实部与虚部），相加相减只需对实部虚部分别相加减，其中add()，sub()函数分别为加减函数，print()函数用于打印复数。

源代码：

// 9.5.cpp

# include <iostream>

# include "Complex.h"

using namespace std;

int main()

{

Complex a(1,2);

Complex b(2,5);

Complex c=a.add(b);

Complex d=a.sub(b);

c.print();

d.print();

system("pause");

return 0;

}

// Complex.h

# include <iostream>

#ifndef Complex\_h

#define Complex\_h

using namespace std;

class Complex{

double Real,Imag;

public:

Complex(double real=0,double imag=0);

Complex add(Complex );

Complex sub(Complex );

void print();

};

#endif

// Complex.cpp

# include <iostream>

# include "Complex.h"

using namespace std;

Complex::Complex(double real,double imag)

{

Real=real;

Imag=imag;

}

Complex Complex::add(Complex a)

{

Complex c(Real+a.Real,Imag+a.Imag);

return c;

}

Complex Complex::sub(Complex a)

{

Complex c(Real-a.Real,Imag-a.Imag);

return c;

}

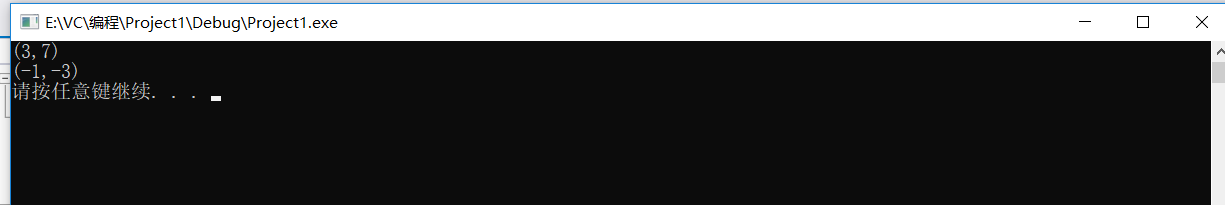
void Complex::print()

{

cout<<"("<<Real<<","<<Imag<<")"<<endl;

}

运行结果：



题9.6

**Rational class**

题目要求：

建立一个自然数类，要求包括两个数据成员numerator和denominator，有默认参数的构造函数，同时，如果fraction可以约分要进行约分，建立若干成员函数，包括以下功能：自然数相加、相减、相乘、相除，以及打印（按a/b的形式进行打印，同时打印对应的浮点数形式）。

解题思路：

类中存储两个数据成员（Num，Den分别表示分子与分母），相加减使用add(),sub()函数，先进行通分，再加减，最后再使用reduce()函数进行约分；乘除算法使用mul(),div()函数，直接进行乘除，最后约分；打印使用print()函数，先打印分数形式，再使用(double)强制转换结果为浮点数打印。

注：代码中的biggest()函数用于计算最大公约数，使用辗转相除法计算最大公约数。

源代码：

//9.6.cpp

# include <iostream>

# include "Rational.h"

using namespace std;

int main()

{

Rational a(1, 4);

Rational b(2, 7);

Rational c = a.add(b);

Rational d = a.sub(b);

Rational e = a.mul(b);

Rational f = a.div(b);

c.print();

d.print();

e.print();

f.print();

system("pause");

return 0;

}

//Rational.h

# include <iostream>

using namespace std;

class Rational {

int Num, Den;

public:

Rational(int num = 0, int den = 1);

Rational add(Rational);

Rational sub(Rational);

Rational mul(Rational);

Rational div(Rational);

void print();

void reduce();

int biggest(int, int);

};

//Rational.cpp

# include <iostream>

# include "Rational.h"

using namespace std;

Rational::Rational(int num, int den)

{

Num = num;

Den = den;

reduce();

}

Rational Rational::add(Rational a)

{

Rational c;

c.Num = Num \* a.Den + Den \* a.Num;

c.Den = Den \* a.Den;

c.reduce();

return c;

}

Rational Rational::sub(Rational a)

{

Rational c;

c.Num = Num \* a.Den - Den \* a.Num;

c.Den = Den \* a.Den;

c.reduce();

return c;

}

Rational Rational::mul(Rational a)

{

Rational c;

c.Num = Num \* a.Num;

c.Den = Den \* a.Den;

c.reduce();

return c;

}

Rational Rational::div(Rational a)

{

Rational c;

c.Num = Num \* a.Den;

c.Den = Den \* a.Num;

c.reduce();

return c;

}

void Rational::print()

{

cout << Num << "/" << Den << endl;

cout << (double)Num / Den << endl;

}

int Rational::biggest(int a,int b)

{

a = abs(a); b = abs(b);

if (a < b)

{

int temp = a;

a = b;

b = temp;

}

if (b == 0)

return a;

else

return biggest(b, a%b);

}

void Rational::reduce()

{

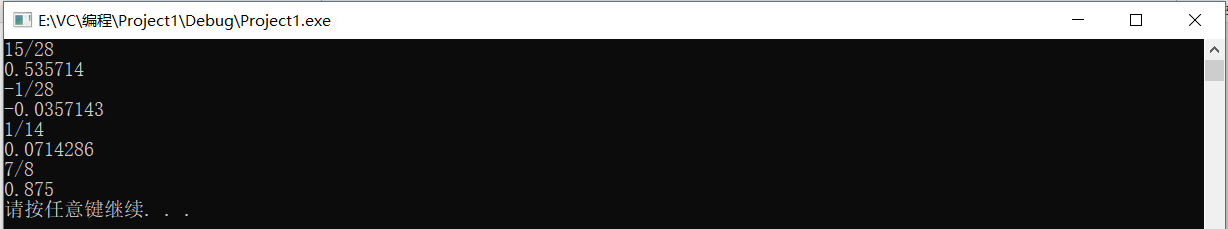
int gcd = biggest(Num, Den);

Num = Num / gcd;

Den = Den / gcd;

}

运行结果：



题9.7

**Enhancing Class Time**

题目要求：

对Figs.9.8—9.9的Time类进行改进，增加一个tick()函数，每次增加一秒钟，通过一个循环，每次循环打印标准时间来检验tick()函数是否正确，检验应包括一下几个要点：增加到下一分钟，下一小时，以及第二天（11.59.59 PM~0.00.00 AM）。

主要思路：

在tick()函数中调用setSecond()函数，并以Second+1为参数，使得每次增加一秒钟，同时，在setSecond(),setMinute(),setHour()中增加if(){}条件判断句，当秒钟或分钟达到60 时，给Second或Minute赋值为0，并调用setMinute(Minute+1)或setHour(Hour+1)，当Hour=24时，给Hour赋值为0。

源代码：

//9.7.cpp

# include <iostream>

# include "Time.h"

using namespace std;

int main()

{

Time time1(15, 28, 56);

for (int i = 0; i < 5; i++)

{

time1.tick();

time1.printStandard();

}

cout << "\n\n";

Time time2(15, 59, 56);

for (int i = 0; i < 5; i++)

{

time2.tick();

time2.printStandard();

}

cout << "\n\n";

Time time3(23, 59, 56);

for (int i = 0; i < 5; i++)

{

time3.tick();

time3.printStandard();

}

return 0;

}

//Time.h

#ifndef Time\_h

#define Time\_h

class Time {

public:

Time(int = 0, int = 0, int = 0); //default constructor

//set functions

void setTime(int, int, int);

void setHour(int);

void setMinute(int);

void setSecond(int);

//get functions

int getHour();

int getMinute();

int getSecond();

void printUnversal();

void printStandard();

void tick();

private:

int hour;

int minute;

int second;

};

#endif

//Time.cpp

# include <iostream>

# include <iomanip>

# include <stdexcept>

# include "Time.h"

using namespace std;

Time::Time(int hour, int minute, int second)

{

setTime(hour, minute, second);

}

void Time::setTime(int h, int m, int s)

{

setHour(h);

setMinute(m);

setSecond(s);

}

void Time::setHour(int h)

{

if (h == 24)

{

h = 0;

}

if (h >= 0 && h < 24)

hour = h;

else

throw invalid\_argument("hour must be 0~23");

}

void Time::setMinute(int m)

{

if (m == 60)

{

m = 0;

Time::setHour(hour + 1);

}

if (m >= 0 && m < 60)

minute = m;

else

throw invalid\_argument("minute must be 0~59");

}

void Time::setSecond(int s)

{

if (s == 60)

{

s = 0;

Time::setMinute(minute + 1);

}

if (s >= 0 && s < 60)

second = s;

else

throw invalid\_argument("second must be 0~59");

}

int Time::getHour()

{

return hour;

}

int Time::getMinute()

{

return minute;

}

int Time::getSecond()

{

return second;

}

void Time::printUnversal()

{

cout << setfill('0') << setw(2) << getHour() << ":" << setw(2) << getMinute() << ":" << setw(2) << getSecond() << endl;

}

void Time::printStandard()

{

cout << (( getHour() == 12) ? 12 : getHour() % 12)

<< ":" << setfill('0') << setw(2) << getMinute() << ":" << setw(2) << getSecond() << (hour < 12 ? "AM" : "PM") << endl;

}

void Time::tick()

{

Time::setSecond(second + 1);

}

运行结果：



题9.14

**HugeInteger Class**

题目要求：

创建一个HugeInteger类，能够存储一个四十位的“巨数”，提供成员函数input()，output()，add()，subtract()。为了对比两个“巨数”，提供bool型函数isEqualTo()，isNotEqualTo(),isGreaterThan()，isLessThan()，isGreaterThanOrEqualTo() 和isLessThanOrEqualTo()，同时提供一个isZero()函数。

主要思路：

用40位的int型数组（digits[40]）存储巨数，在初始化时，用户只需输入巨数，程序将以字符串的形式读入，在初始化再将其转化成对应的每一位数字，这一操作简化了用户的输入。在加减法中依次对每一位进行加减，结果大于等于10的，则数组前一个元素+1；若相减出现负数的则前一位-1，此位+10。

注：巨数在digits[]里的存储如下图所示：

digits[39]

3

digits[38]

2

digits[37]

1

digits[2]

0

digits[1]

0

digits[0]

0

**……**

将巨数存储在如图所示的数组中，并且由后往前分别为个位，十位，百位……（此图表示的数字为123），当巨数没有达到40位时，自动对缺少的位次赋值为0。

源代码：

//9.14.cpp

# include <iostream>

# include "HugeInteger.h"

using namespace std;

int main()

{

HugeInteger hugeinteger1("100000000000000000000");

HugeInteger hugeinteger2("200000000000000000000000000");

HugeInteger hugeinteger3 = hugeinteger1.add(hugeinteger2);

HugeInteger hugeinteger4 = hugeinteger1.subtract(hugeinteger2);

hugeinteger3.output();

hugeinteger4.output();

return 0;

}

//HugeInteger.h

# include <iostream>

using namespace std;

class HugeInteger {

public:

HugeInteger(string a = "0");

void input(string );

void output();

HugeInteger add(HugeInteger);

HugeInteger subtract(HugeInteger);

bool isEqualTo(HugeInteger);

bool isNotEqualTo(HugeInteger);

bool isGreaterThan(HugeInteger);

bool isLessThan(HugeInteger);

bool isGreaterThanorEqualThan(HugeInteger);

bool isLessThanorEqualThan(HugeInteger);

bool isZero(HugeInteger);

private:

int digits[40];

bool t;

};

//HugeInteger.cpp

# include "HugeInteger.h"

using namespace std;

HugeInteger::HugeInteger(string a)

{

int Long = a.length();

int left = 39;

for (int i = Long-1; i >=0; i--,left--)

{

digits[left] = (int)a[i] - 48;

}

for (int i = left; i >= 0; i--)

digits[i] = 0;

}

void HugeInteger::input(string a)

{

int Long = a.length();

int left = 39;

for (int i = Long - 1; i >= 0; i--, left--)

{

digits[left] = (int)a[i] - 48;

}

for (int i = left; i >= 0; i--)

digits[i] = 0;

}

void HugeInteger::output()

{

bool k = 0;

if (digits[0] != 0)

k = 1;

if (!t)

cout << "-";

for (int i = 0; i < 40 ; i++)

{

if (digits[i] != 0)

k = 1;

if(k)

cout << digits[i];

}

cout << endl;

}

HugeInteger HugeInteger::add(HugeInteger a)

{

HugeInteger c;

for (int i = 39; i > 0; i--)

{

c.digits[i] = digits[i] + a.digits[i];

if (c.digits[i] > 9)

{

c.digits[i] = c.digits[i] % 10;

c.digits[i - 1]++;

}

}

c.digits[0] += digits[0] + a.digits[0];

if (c.digits[0] >= 10)

{

cout << "Error adding !! Too big result !!!" << endl;

return c;

}

return c;

}

HugeInteger HugeInteger::subtract(HugeInteger a)

{

HugeInteger c;

for (int i = 39; i >= 0; i--)

c.digits[i] = digits[i] - a.digits[i];

if (isLessThan(a))

{

for (int i = 0; i < 40; i++)

c.digits[i] = -c.digits[i];

for (int i = 39; i > 0; i--)

{

if (c.digits[i] < 0)

{

c.digits[i] += 10;

c.digits[i - 1]--;

}

}

c.t = 0;

}

else

{

for (int i = 39; i > 0; i--)

{

if (c.digits[i] < 0)

{

c.digits[i] += 10;

c.digits[i - 1]--;

}

}

}

return c;

}

bool HugeInteger::isEqualTo(HugeInteger a)

{

bool k = 1;

for (int i = 0; i < 40; i++)

{

if (digits[i] != a.digits[i])

{

k = 0;

break;

}

}

return k;

}

bool HugeInteger::isNotEqualTo(HugeInteger a)

{

bool k = 1;

for (int i = 0; i < 40; i++)

{

if (digits[i] != a.digits[i])

{

k = 0;

break;

}

}

return !k;

}

bool HugeInteger::isGreaterThan(HugeInteger a)

{

bool k = 1;

for (int i = 0; i < 40; i++)

{

if (digits[i] < a.digits[i])

{

k = 0;

break;

}

else

if (digits[i] > a.digits[i])

break;

if (i == 39 && (digits[i] == a.digits[i]))

k = 0;

}

return k;

}

bool HugeInteger::isLessThan(HugeInteger a)

{

bool k = 1;

for (int i = 0; i < 40; i++)

{

if (digits[i] < a.digits[i])

{

k = 0;

break;

}

else

if (digits[i] > a.digits[i])

break;

if (i == 39 && (digits[i] == a.digits[i]))

k = 1;

}

return !k;

}

bool HugeInteger::isGreaterThanorEqualThan(HugeInteger a)

{

bool k = 1;

for (int i = 0; i < 40; i++)

{

if (digits[i] < a.digits[i])

{

k = 0;

break;

}

else

if (digits[i] > a.digits[i])

break;

}

return k;

}

bool HugeInteger::isLessThanorEqualThan(HugeInteger a)

{

bool k = 1;

for (int i = 0; i < 40; i++)

{

if (digits[i] < a.digits[i])

{

k = 0;

break;

}

else

if (digits[i] > a.digits[i])

break;

}

return !k;

}

bool HugeInteger::isZero(HugeInteger)

{

bool k = 1;

for (int i = 0; i < 40; i++)

{

if (!digits[i])

{

k = 0;

break;

}

}

return k;

}

