**第一部分 第1.1题：长整数运算**

**实验报告**

题目：编制一个实现任意长的整型进行加法运算的演示程序。

班级：F1702128 姓名：赵伟基 学号：517021910883 完成日期：2018-11-2

1. **需求分析**

1.本实验要求输入和输出形式需按中国对于长整数的表示习惯，每四位一组，组间用逗号隔开，输入需能筛除错误字符，输出的运算结果不含非法字符。

2.本实验要求相加过程不能为长整数设定长度限定，不能破坏操作数列表。对加法操作进行程序设计。

3.本实验应对长整数类进行加法及减法运算符，赋值运算符重载，并对构造函数，复制构造函数等进行定义。

1. **概要设计**

对高精度长整数运算的保存，由于不常执行插入和删除操作，故本实验选择顺序存储：

对每个整数创建一个Longlongint对象，包含sign符号位及存储数字的num。考虑到整数运算都是从个位数开始，因此把大整数逆序存放。并用Longlongint类进行包装。

（一）class Longlongint

1. 数据对象：char sign符号位，char\*num 储存数字。
2. 基本操作：
3. Longlongint operator+（）：

长整数类的加法运算符重载；

初始条件：存在两个Longlongint类的对象；

操作结果：获得两个对象的和。

1. Longlongint operator-（）：

长整数类的减法运算符重载；

初始条件：存在两个Longlongint类的对象；

操作结果：获得两个对象的差。

1. Ostream &operator<<（）:

长整数类输出运算符重载；

初始条件：存在一个Longlongint类的对象；

操作结果：以规范格式输出该对象。

1. Longlongint（const char \*n=””）:

操作结果：构造函数，生成一个空的长整数型对象。

1. Longlongint（const Longlongint &）:

初始条件：已存在一个char\*类的对象或Longlongint类的对象；

操作结果：生成一个值与原有对象相等的长整数型的对象。

1. Longlongint &operator=(const Longlongint &);

初始条件：存在一个Longlongint类的对象；

操作结果：将该对象的值赋给另一个Longlongint类的对象。

1. ~Longlongint（）

初始条件：Longlongint类的对象已经建立

操作结果：析构函数，销毁该对象并释放空间。

（二）工具函数

（1）add（）

初始条件：存在两个char\*类的对象，即存在两个确定的数字字符串；

操作结果：返回有两个字符串相加结果的字符串；

（2）sub（）

初始条件：存在两个char\*类的对象，即存在两个确定的数字字符串；

操作结果：返回有两个字符串相减结果的字符串；

（3）greaterr（）

初始条件：存在两个char\*类的对象n1,n2，即存在两个确定的数字字符串；

操作结果：比较n1,n2字符串对应数字大小，若n1>n2返回true，否则返回false。

（三）main函数

{

构建Longlongint对象n1,n2；

为n1,n2赋值；

按规定格式输出n1+n2的值；

}

**三、 详细设计**

**（1）bignum.h**

#ifndef BIGNUM\_H\_INCLUDED

#define BIGNUM\_H\_INCLUDED

#include <iostream>

#include <cstring>

using namespace std;

class Longlongint

{

friend Longlongint operator+(const Longlongint&,const Longlongint&);

friend ostream &operator<<(ostream &,const Longlongint&);

friend Longlongint operator-(const Longlongint&,const Longlongint&);

private:

char sign; //符号

char \*num; //数字

public:

Longlongint(const char \*n="");

Longlongint(const Longlongint&);

Longlongint&operator=(const Longlongint&);

~Longlongint(){delete num;}

};

char \*add(const char \*s1,const char \*s2); //返回有两个字符串相加结果的字符串；

char \*sub(const char \*s1,const char \*s2); //返回有两个字符串相减结果的字符串；

bool greaterr(const char \*s1,const char \*s2); //s1>s2的逻辑值

#endif // BIGNUM\_H\_INCLUDED

**(2)bignum.cpp**

#include <iostream>

#include <cstring>

#include "bignum.h"

using namespace std;

Longlongint::Longlongint(const char \*n)

{

char \*temp;

switch (n[0]){

case '+':sign='+';break; //符号位定义

case '-':sign='-';break;

default:sign='+';

}

int len1;

if (n[0]=='+'||n[0]=='-')

len1=strlen(n)-1;

else

len1=strlen(n);

temp=new char[len1+1];

if (n[0]=='+'||n[0]=='-')

for(int i=0;i<len1;i++)

{

temp[i]=n[i+1]; //除符号

}

else

for(int i=0;i<len1;i++)

{

temp[i]=n[i];

}

temp[len1]='\0';

for(int i=0;i<len1;i++)

{

if(temp[i]==',')

for(int j=i;j<len1+1;j++) //消除','

temp[j]=temp[j+1];

}

if(strcmp(temp,"0")==0) temp=""; //特殊情况，若temp=0，直接附空字符串值

int len=strlen(temp);

num=new char[len+1];

for(int i=0;i<len;i++) num[len-i-1]=temp[i]; //用数组倒序存储数字

num[len]='\0';

delete temp;

}

Longlongint::Longlongint(const Longlongint &other)

{

sign=other.sign;

num=new char [strlen(other.num)+1];

strcpy(num,other.num); //拷贝构造

}

Longlongint&Longlongint::operator=(const Longlongint &right)

{

if(this==&right) return \*this;

delete num;

sign=right.sign;

num=new char[strlen(right.num)+1];

strcpy(num,right.num); //赋值 包括符号赋值及字符串赋值

return \*this;

}

ostream &operator<<(ostream &os,const Longlongint &obj)

{

char \*temp;

int dounum=strlen(obj.num)/4-1; //输出增加的','数目

int k=0,pause=0;

if(strlen(obj.num)==0) os<<0;

else{

if(obj.sign=='-')

os<<obj.sign;

temp=new char[dounum+strlen(obj.num)+1];

for(int j=0;j<strlen(obj.num);j++)

{

temp[j+k]=obj.num[j];

pause++;

if(pause==4)//每四位加一个‘，’

{

k++;

temp[j+k]=',';

pause=0;

}

}

temp[dounum+strlen(obj.num)+1]='\0';

if(temp[strlen(temp)-1]==',') temp[strlen(temp)-1]='\0';//消除首位‘，’

for(int i=strlen(temp)-1;i>=0;--i) os<<temp[i];

delete temp; //输出增加'，'

}

return os;

}

Longlongint operator+(const Longlongint&n1,const Longlongint &n2)

{

Longlongint n;

delete n.num;

if(n1.sign==n2.sign){

n.sign=n1.sign;

n.num=add(n1.num,n2.num); //若符号相同，直接将n1数字与n2数字相加

}

else if(greaterr(n1.num,n2.num)){

n.sign=n1.sign;

n.num=sub(n1.num,n2.num); //若符号不同 则用绝对值大的数减去绝对值小的数

}

else{

n.sign=n2.sign;

n.num=sub(n2.num,n1.num);

}

return n;

}

Longlongint operator-(const Longlongint&n1,const Longlongint &n2)

{

Longlongint n(n2);

if(n.sign=='+') n.sign='-';

else n.sign='+';

return n1+n; //改变减数符号 利用加法完成

}

char \*add(const char\*s1,const char\*s2)//绝对值相加函数

{

int len1=strlen(s1);

int len2=strlen(s2);

int minLen=(len1>len2?len2:len1);

int len=(len1>len2?len1:len2)+1; //相加后字符串的长度

int carry=0,result; //carry模拟进位

int i=0;

char \*num=new char [len+1];

for( i=0;i<minLen;i++) //对都存在的位数，模拟手动相加进位

{

result=s1[i]-'0'+s2[i]-'0'+carry;

num[i]=result%10+'0'; //取本位结果

carry=result/10; //进位

}

while(i<len1){

result=s1[i]-'0'+carry; //对S1或S2都存在的位数，直接补充至和上

num[i]=result%10+'0';

carry=result/10;

++i;

}

while(i<len2){

result=s2[i]-'0'+carry;

num[i]=result%10+'0';

carry=result/10;

++i;

}

if(carry!=0) num[i++]=carry+'0';//最高进位

num[i]='\0';

return num;

}

char \*sub(const char\*s1,const char\*s2)

{

if(!strcmp(s1,s2)) return "";

int len1=strlen(s1);

int len2=strlen(s2);

int minus=0; //借位记录

int i=0;

char \*tmp=new char [len1+1];

for( i=0;i<len2;++i)

{

tmp[i]=s1[i]-s2[i]-minus;

if(tmp[i]<0){tmp[i]+=10;minus=1;}

else {minus=0;}

tmp[i]+='0';

}

while (i<len1)

{

tmp[i]=s1[i]-'0'-minus;

if(tmp[i]<0){tmp[i]+=10;minus=1;}

else minus=0;

tmp[i]+='0';

++i;

}

do{

--i;

}while(i>=0&&tmp[i]=='0');

tmp[i+1]='\0';

return tmp;

}

bool greaterr(const char \*s1,const char \*s2)

{

int len1=strlen(s1),len2=strlen(s2);

if(len1>len2) return true;

else if(len1<len2) return false; //s1长度长于s2长度 则s1>s2；

for (int i=len1-1;i>=0;--i)

{

if(s1[i]>s2[i]) return true; //s1高位大 返回true

else if(s1[i]<s2[i]) return false;

}

return false;

}

**(3)main.cpp**

#include <iostream>

#include "bignum.h"

#include <cstring>

using namespace std;

int main()

{

Longlongint a,b;

char a1[100],b1[100];

cin>>a1>>b1;

a=a1;

b=b1;

cout<<a+b;

return 0;

}

* 1. **调试分析**

1. 算法复杂度分析
2. 时间复杂度

O（n）：Longlongint(const char \*n)；Longlongint(const Longlongint &other)；

operator=()；operator<<（）； add(); sub(); greater()。

~ Longlongint();

O（1）：operator+();operator-()。

算法复杂度为O(n)的函数需对数字字符串进行遍历。

1. 空间复杂度

程序中每个函数都需要使用num，复杂度为O(n)。

1. 由于对长整数初期构思不足，采取顺序结构，没有采用链表，导致算法效率并不高。
2. 本实验对长整数类进行了很好的包装，对=，+，-和<<都采用了运算符重载。数据安全性得到保证。
   1. **用户手册**

1.本程序使用的 Code::Blocks 16.01 IDE，程序以项目（project）方式组织，







1. 点击项目中的main.cpp->build and run 依次输入两个长整数即可。如需使用减法，在mian函数中将a+b更换为a-b即可。
   1. **测试结果**

（1）0;0;输出 0

（2）-2345,6789;-7654,3211; 输出-1,0000,0000

（3）-9999,9999;1,0000,0000,0000; 输出 9999,0000,0001

（4）1,0001,0001;-1,0001,0001; 输出 0

（5）1,0001,0001;-1,0001,0000; 输出 1

（6）-9999,9999,9999;-9999,9999,9999; 输出-1,9999,9999,9998

（7）1,0000,9999,9999;-1,0001,0001; 输出9999,9998,9998

* 1. **附录**

源程序文件名清单：

1. main.cpp //主函数
2. bignum.h //长整数类头文件
3. bignum.cpp//长整数类实现文件