C++ BigInteger代碼模板

#include<iostream>

#include<cstring>

#include<string>

#include<cstdio>

const int maxn = 500;

using namespace std;

struct BigInteger{

int len, s[maxn];

BigInteger() {

memset(s, 0, sizeof(s));

len = 1;

}

BigInteger(int num) {

\*this = num;

}

BigInteger(const char\* num) {

\*this = num;

}

BigInteger operator = (int num) {

char s[maxn];

sprintf(s, "%d", num);

\*this = s;

return \*this;

}

BigInteger operator = (const char\* num) {

len = strlen(num);

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

s[i] = num[len - i - 1] - '0';

return \*this;

}

string str() const {

string res = "";

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

res = (char)(s[i] + '0') + res;

if(res == "")

res = "0";

return res;

}

BigInteger operator + (const BigInteger& b) const{

BigInteger c;

c.len = 0;

for(int i = 0, g = 0; g || i < max(len, b.len); i++) {

int x = g;

if(i < len)

x += s[i];

if(i < b.len) x += b.s[i];

c.s[c.len++] = x % 10;

g = x / 10;

}

return c;

}

void clean() {

while(len > 1 && !s[len-1])

len--;

}

BigInteger operator \* (const BigInteger& b) const{

BigInteger c; c.len = len + b.len;

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

for(int j = 0; j < b.len; j++)

c.s[i+j] += s[i] \* b.s[j];

for(int i = 0; i < c.len-1; i++){

c.s[i+1] += c.s[i] / 10;

c.s[i] %= 10;

}

c.clean();

return c;

}

BigInteger operator - (const BigInteger& b) const{

BigInteger c; c.len = 0;

for(int i = 0, g = 0; i < len; i++) {

int x = s[i] - g;

if(i < b.len)

x -= b.s[i];

if(x >= 0)

g = 0;

else {

g = 1;

x += 10;

}

c.s[c.len++] = x;

}

c.clean();

return c;

}

bool operator < (const BigInteger& b) const{

if(len != b.len) return len < b.len;

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

if(s[i] != b.s[i])

return s[i] < b.s[i];

return false;

}

bool operator > (const BigInteger& b) const{

return b < \*this;

}

bool operator <= (const BigInteger& b) const{

return (b > \*this);

}

bool operator >= (const BigInteger& b) const{

return (b < \*this);

}

bool operator == (const BigInteger& b) const{

return !(b < \*this) && !(\*this < b);

}

bool operator != (const BigInteger& b) const{

return !(\*this == b);

}

BigInteger operator += (const BigInteger& b) {

\*this = \*this + b;

return \*this;

}

BigInteger operator -= (const BigInteger& b) {

\*this = \*this - b;

return \*this;

}

BigInteger operator ++ (int) {

\*this = \*this + 1;

return \*this;

}

BigInteger operator -- (int) {

\*this = \*this - 1;

return \*this;

}

BigInteger operator ++ () {

\*this = \*this + 1;

return \*this;

}

BigInteger operator -- () {

\*this = \*this - 1;

return \*this;

}

};

istream& operator >> (istream &in, BigInteger& x) {

string s;

in >> s;

x = s.c\_str();

return in;

}

ostream& operator << (ostream &out, const BigInteger& x) {

out << x.str();

return out;

}