第7章　结构体和文件

第78课　大整数

《信息学奥赛一本通·编程启蒙 C++版》

一、__int128

顾名思义， __int128 就是占用 128 字节的整数存储类型。由于是二进制，范围就是 -2^{127} ~ $2^{127}-1$，如果使用了 unsigned __int128，则范围变成 0 ~ 2^128，即约 39 位数！如果遇到 long long 变量溢出的情况，可以使用 __int128。

但是__int128 类型的变量，不能直接用 cin 和 cout 输出，需要用我们前面讲到过的“快读”和“快写”的方法进行输入和输出。

二、用结构体或类定义大数据

利用计算机进行数值计算，有时会遇到这样的问题：有些计算要求精度高，希望计算的数的位数可达几十位甚至几百位，虽然计算机的计算精度也算较高了但因受到硬件的限制，往往达不到实际问题所要求的精度。我们可以利用程序设计的方法去实现这样的高精度计算。介绍常用的几种高精度计算的方法。

高精度计算中需要处理好以下几个问题：

(1)数据的接收方法和存贮方法

数据的接收和存贮：当输入的数很长时，可采用字符串方式输入，这样可输入数字很长的数，利用字符串函数和操作运算，将每一位数取出，倒着存入数组中。

```
1. void init(int a[])  //传入一个数组
2. {
3.     string s;
4.     cin>>s;  //读入字符串 s
5.     len=s.length();//用 len 计算字符串 s 的位数
6.     for(i=1;i<=len;i++)
7.       a[i]=s[len-i]-'0'; //将数串 s 转换为数组 a，并倒序存储
8. }
```

(2) 高精度数位数的确定

位数的确定:接收时往往是用字符串的,所以它的位数就等于字符串的长度。

(3) 进位,借位处理

加法进位：c[i]=a[i]+b[i];

```
if (c[i]>=10) { c[i]%=10; ++c[i+1]; }
```

减法借位：

```
if (a[i]<b[i]) { --a[i+1]; a[i]+=10; }
c[i]=a[i]-b[i];
```

乘法进位：

```
c[i+j-1]= a[i]*b[j] + x + c[i+j-1];
x = c[i+j-1]/10;
c[i+j-1] %= 10;
```

【例 78.1】忽明忽暗

【题目描述】

走廊里有 n 盏灯，编号依次为,2,3,…,n，由学校电路控制中心管理。初始时，所有灯都是关闭的。某黑客入侵了学校电路控制中心，黑客想让灯忽明忽暗，进行了 n 轮操作。第 i 轮操作，会让所有编号为 i 的倍数的灯状态反转，也就是打开的变为关闭，关闭的变为打开。

现在黑客想知道，n 轮操作后，所有亮着的灯的编号之和为多少。因为答案很大，只需输出答案对 10^9+7 取模的结果。

【输入格式】

一个整数 n，表示灯的个数。对于 100% 的数据 $1 \leqslant n \leqslant 10^{18}$

【输出格式】

一个整数，表示亮着的灯的编号之和对 10^9+7 取模的结果

【样例输入】

```
20
```

【样例输出】

```
30
```

【代码实现】

```cpp
#include <bits/stdc++.h>
#define ll long long
using namespace std;
int main(){
    ll N;
    int x=1e9+7;
    cin>>N;
    __int128 n=sqrt(N);
    ll ans=n*(n+1)*(2*n+1)/6%x;
    cout<<ans<<endl;
    return 0;
}
```

【例 78.2】大整数加法

【题目描述】

求两个不超过 200 位的非负整数的和。

【输入格式】

有两行，每行是一个不超过 200 位的非负整数，可能有多余的前导 0

【输出格式】

一行，即相加后的结果。结果里不能有多余的前导 0，即如果结果是 342，那么就不能输出为 0342

【样例输入】

```
20
10
```

【样例输出】

```
30
```

```cpp
#include<bits/stdc++.h>
using namespace std;
class Bigint{
private:
    int a[205];
public:
    Bigint(){
        memset(a,0,sizeof(a));
        a[0]=1;
    }
   friend ostream & operator<<(ostream &o,const Bigint & b);
   friend istream & operator >> (istream &in,Bigint & b);
    Bigint operator+ (Bigint r){
        Bigint c;
        c.a[0]=max(a[0],r.a[0]);
        int g=0;
        for(int i=1;i<=c.a[0];i++){
            c.a[i]=a[i]+r.a[i]+g;
            if(c.a[i]>9){g=1;c.a[i]-=10;}
            else g=0;
        }
        if(g==1){
            c.a[0]++;
            c.a[c.a[0]]=1;
        }
        return c;
    }
};
```

```cpp
istream & operator >> (istream &in,Bigint & b){
    string s;
    in>>s;
    memset(b.a,0,sizeof(b.a));
    b.a[0]=s.size();
    for(int i=0,j=b.a[0];i<b.a[0];i++,j--)
       b.a[j]=s[i]-'0';
    for(int len=b.a[0],i=len;i>1;i--)
       if(b.a[i]==0) b.a[0]--; else break;
    return in;
}
ostream &operator << (ostream &o,const Bigint & b){
       for(int i=b.a[0];i>0;i--) o<<b.a[i];
       return o;
 }
int main(){
    Bigint a,b;
    cin>>a>>b;
    cout<<a+b;
    return 0;
}
```

【例 78.3】回文数(Noip1999)

【题目描述】

若一个数（首位不为零）从左向右读与从右向左读都是一样，我们就将其称之为回文数。例如：给定一个 10 进制数 56，将 56 加 65（即把 56 从右向左读），得到 121 是一个回文数。又如，对于 10 进制数 87，

STEP1： 87＋78= 165 STEP2： 165＋561= 726

STEP3： 726＋627＝1353 STEP4：1353+3531=4884

在这里的一步是指进行了一次 N 进制的加法,上例最少用了 4 步得到回文数 4884。

写一个程序，给定一个 N（2＜N＜＝10 或 N=16）进制数 M. 求最少经过几步可以得到回文数。如果在 30 步以内（包含 30 步）不可能得到回文数，则输出“Impossible” 。

【输入格式】

第 1 行，给定一个 N（2＜N≤10 或 N=16）表示进制；

第 2 行，一个 N 进制数 M

【输出格式】

最少几步。如果在 30 步以内（包含 30 步）不可能得到回文数，则输出“Impossible”

【样例输入】

9

87

【样例输出】

6

【代码实现】

```
#include <bits/stdc++.h>
using namespace std;
string s;
int n;
struct Bigint{
   int a[100005],len;
   Bigint (){
      len=1;
      memset(a,0,sizeof(a));
   }
Bigint operator= (string s){
    len=s.size();
    bool flag=false;
    for(int i=1;i<=len;i++){
       if(s[len-i]>='0' && s[len-i]<='9'){
          a[i]=s[len-i]-'0';
       }else{
          a[i]=s[len-i]-'A'+10;
       }
    }
    return *this;
}
Bigint operator+ (Bigint x){
    Bigint ans;
    if(x.len>len) ans.len=x.len;
    else ans.len=len;
    for(int i=1;i<=ans.len;i++){
       ans.a[i]+=x.a[i]+a[i];
       ans.a[i+1]+=ans.a[i]/n;
       ans.a[i]%=n;
    }
    if(ans.a[ans.len+1]>0) ans.len++;
    return ans;
 }
 void print(){
    for(int i=len;i>=1;i--){
      cout<<a[i];
    }
 }
};
Bigint a,b;
Bigint deal(Bigint x){
  for(int i=1;i<=x.len/2;i++){
    swap(x.a[i],x.a[x.len-i+1]);
  }
  return x;
}
bool check(Bigint x){
  for(int i=1;i<=x.len/2;i++){
    if(x.a[i]!=x.a[x.len-i+1]) return false;
  }
  return true;
}
int main(){
   cin>>n>>s;
   a=s;
   b=deal(a);
   for(int i=1;i<=30;i++){
      a=a+b;
      if(check(a)){
         cout<<i;
         return 0;
      }
      b=deal(a);
   }
   cout<<"Impossible";
   return 0;
}
```

练 78.1 高精度乘法

【题目描述】

输入两个高精度正整数 M 和 N（M 和 N 均小于 100 位）。求这两个高精度数的积。

【输入格式】

输入两个高精度正整数 M 和 N

【输出格式】

求这两个高精度数的积

【样例输入】

```
36
3
```

【样例输出】

```
108
```

【代码实现】

```
#include<bits/stdc++.h>
using namespace std;
struct big_int{
    int a[1000],len;
    big_int operator= (string s){
        memset(a,0,sizeof(a));
        int l=s.size();
        len=l;
        for(int i=0;i<l;i++){
            a[l-i]=s[i]-48;
        }
        return *this;
    }
big_int operator* (big_int x){
    big_int ans;
    ans="0";
    for(int i=1;i<=len;i++){
        for(int j=1;j<=x.len;j++){
            ans.a[i+j-1]+=a[i]*x.a[j];
            ans.a[i+j]+=ans.a[i+j-1]/10;
            ans.a[i+j-1]%=10;
        }
    }
    ans.len=len+x.len;
    if(ans.a[ans.len]==0){
        ans.len--;
    }
    return ans;
 }
 void print(){
    for(int i=len;i>=1;i--){
        cout<<a[i];
    }
 }
};
big_int n,m,ans;
string s;
int main(){
    cin>>s;
    n=s;
    cin>>s;
    m=s;
    ans=n*m;
    ans.print();
    return 0;
}
```

练 78.2 大整数减法

【题目描述】

求两个大的正整数相减的差。

【输入格式】

共 2 行，第 1 行是被减数 a，第 2 行是减数 b(a > b)。每个大整数不超过 200 位，不会有多余的前导零

【输出格式】

一行，即所求的差

【样例输入】

```
99999999999999999999999999999999999999
999999999999
```

【样例输出】

```
99999999999999999999990000000000000000
```

【代码实现】

```
#include<bits/stdc++.h>
using namespace std;
using namespace std;
struct big_int{
    int a[1005];
    int len;
    big_int operator= (string s){
        memset(a,0,sizeof(a));
        int l=s.size();
        len=l;
        for(int i=0;i<l;i++){
            a[i]=s[l-i-1]-48;
    }
    return *this;
}
big_int operator+ (big_int &x){
    big_int ans;
    ans="0";
    ans.len=len;
    if(x.len>ans.len) ans.len=x.len;
    for(int i=0;i<ans.len;i++){
        ans.a[i]+=a[i]+x.a[i];
        ans.a[i+1]=ans.a[i]/10;
        ans.a[i]%=10;
    }
    if(ans.a[ans.len]>0){
        ans.len++;
    }
    return ans;
}
big_int operator- (big_int &x){
    big_int ans;
    ans="0";
    ans.len=len;
    if(x.len>ans.len) ans.len=x.len;
    for(int i=0;i<len;i++){
        ans.a[i]=a[i]-x.a[i];
        if(ans.a[i]<0){
            ans.a[i]+=10;
            a[i+1]--;
        }
    }
    while(ans.a[ans.len-1]==0 && ans.len>1){
        ans.len--;
    }
    return ans;
}
bool operator< (big_int &x){
    if(len<x.len) return true;
    else if(len>x.len) return false;
    else{
      for(int i=len-1;i>=0;i--){
          if(a[i]<x.a[i]) return true;
          else if(a[i]>x.a[i]) return false;
      }
      return false;
    }
}
bool operator>= (big_int &x){
    return !(*this<x);
}
void print(){
    for(int i=len-1;i>=0;i--)
        cout<<a[i];
    }
};
big_int n,m,a;
string s;
int main(){
    cin>>s;
    n=s;
    cin>>s;
    m=s;
    if(n>=m){
        a=n-m;
    }else{
        cout<<"-";
        a=m-n;
    }
    a.print();
    return 0;
}
```

谢谢！