

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 $-2^{127} \sim 2^{127}-1$ ，如果使用了 unsigned __int128，则范围变成 $0 \sim 2^{128}$ ，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: $c[i]=a[i]+b[i];$

$\text{if } (c[i] \geq 10) \{ c[i] \% 10; ++c[i+1]; \}$

减法借位: $\text{if } (a[i] < b[i]) \{ --a[i+1]; a[i] += 10; \}$

$c[i]=a[i]-b[i];$

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 $2, 3, \dots, n$ ，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 $1 \leq n \leq 10^{18}$

【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】

```
1. #include <bits/stdc++.h>
2. #define ll long long
3. using namespace std;
4. int main(){
5.     ll N;
6.     int x=1e9+7;
7.     cin>>N;
8.     __int128 n=sqrt(N);
9.     ll ans=n*(n+1)*(2*n+1)/6%x;
10.    cout<<ans<<endl;
11.    return 0;
12.}
```

【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

```

1. #include<bits/stdc++.h>
2. using namespace std;
3. class Bigint{
4. private:
5.     int a[205];
6. public:
7.     Bigint(){
8.         memset(a,0,sizeof(a));
9.         a[0]=1;
10.    }
11.    friend ostream & operator<<(ostream &o,const Bigint & b);
12.    friend istream & operator >> (istream &in,Bigint & b);
13.    Bigint operator+ (Bigint r){
14.        Bigint c;
15.        c.a[0]=max(a[0],r.a[0]);
16.        int g=0;
17.        for(int i=1;i<=c.a[0];i++){
18.            c.a[i]=a[i]+r.a[i]+g;
19.            if(c.a[i]>9){g=1;c.a[i]-=10;}
20.            else g=0;
21.        }
22.        if(g==1){
23.            c.a[0]++;
24.            c.a[c.a[0]]=1;
25.        }
26.        return c;
27.    }
28. };

```

```

29. istream & operator >> (istream &in,Bigint & b){
30.     string s;
31.     in>>s;
32.     memset(b.a,0,sizeof(b.a));
33.     b.a[0]=s.size();
34.     for(int i=0,j=b.a[0];i<b.a[0];i++,j--)
35.         b.a[j]=s[i]-'0';
36.     for(int len=b.a[0],i=len;i>1;i--)
37.         if(b.a[i]==0) b.a[0]--; else break;
38.     return in;
39. }
40. ostream &operator << (ostream &o,const Bigint & b){
41.     for(int i=b.a[0];i>0;i--) o<<b.a[i];
42.     return o;
43. }
44. int main(){
45.     Bigint a,b;
46.     cin>>a>>b;
47.     cout<<a+b;
48.     return 0;
49. }

```

【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 ($2 < N \leq 10$ 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 ($2 < N \leq 10$ 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】

```
1. #include <bits/stdc++.h>
2. using namespace std;
3. string s;
4. int n;
5. struct Bigint{
6.     int a[100005],len;
7.     Bigint (){
8.         len=1;
9.         memset(a,0,sizeof(a));
10.    }
11. Bigint operator= (string s){
12.     len=s.size();
13.     bool flag=false;
14.     for(int i=1;i<=len;i++){
15.         if(s[len-i]>='0' && s[len-i]<='9'){
16.             a[i]=s[len-i]-'0';
17.         }else{
18.             a[i]=s[len-i]-'A'+10;
19.         }
20.     }
21.     return *this;
22. }
23. Bigint operator+ (Bigint x){
24.     Bigint ans;
25.     if(x.len>len) ans.len=x.len;
26.     else ans.len=len;
27.     for(int i=1;i<=ans.len;i++){
28.         ans.a[i]+=x.a[i]+a[i];
29.         ans.a[i+1]+=ans.a[i]/n;
30.         ans.a[i]%=n;
31.     }
```

```
32.     if(ans.a[ans.len+1]>0) ans.len++;
33.     return ans;
34. }
35. void print(){
36.     for(int i=len;i>=1;i--){
37.         cout<<a[i];
38.     }
39. }
40. };
41. Bigint a,b;
42. Bigint deal(Bigint x){
43.     for(int i=1;i<=x.len/2;i++){
44.         swap(x.a[i],x.a[x.len-i+1]);
45.     }
46.     return x;
47. }
48. bool check(Bigint x){
49.     for(int i=1;i<=x.len/2;i++){
50.         if(x.a[i]!=x.a[x.len-i+1]) return false;
51.     }
52.     return true;
53. }
54. int main(){
55.     cin>>n>>s;
56.     a=s;
57.     b=deal(a);
58.     for(int i=1;i<=30;i++){
59.         a=a+b;
60.         if(check(a)){
```

```
61.             cout<<i;
62.             return 0;
63.         }
64.         b=deal(a);
65.     }
66.     cout<<"Impossible";
67.     return 0;
68. }
```

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】

```
1. #include<bits/stdc++.h>
2. using namespace std;
3. struct big_int{
4.     int a[1000],len;
5.     big_int operator= (string s){
6.         memset(a,0,sizeof(a));
7.         int l=s.size();
8.         len=l;
9.         for(int i=0;i<l;i++){
10.            a[l-i]=s[i]-48;
11.        }
12.        return *this;
13.    }
14. big_int operator* (big_int x){
15.     big_int ans;
16.     ans="0";
17.     for(int i=1;i<=len;i++){
18.         for(int j=1;j<=x.len;j++){
19.             ans.a[i+j-1]+=a[i]*x.a[j];
20.             ans.a[i+1]+=ans.a[i+j-1]/10;
21.             ans.a[i+j-1]%=10;
22.         }
23.     }
```

```
24.     ans.len=len+x.len;
25.     if(ans.a[ans.len]==0){
26.         ans.len--;
27.     }
28.     return ans;
29. }
30. void print(){
31.     for(int i=len;i>=1;i--){
32.         cout<<a[i];
33.     }
34. }
35. };
36. big_int n,m,ans;
37. string s;
38. int main(){
39.     cin>>s;
40.     n=s;
41.     cin>>s;
42.     m=s;
43.     ans=n*m;
44.     ans.print();
45.     return 0;
46. }
```


【代码实现】

```
1. #include<bits/stdc++.h>
2. using namespace std;
3. using namespace std;
4. struct big_int{
5.     int a[1005];
6.     int len;
7.     big_int operator= (string s){
8.         memset(a,0,sizeof(a));
9.         int l=s.size();
10.        len=l;
11.        for(int i=0;i<l;i++){
12.            a[i]=s[l-i-1]-48;
13.        }
14.        return *this;
15.    }
16.    big_int operator+ (big_int &x){
17.        big_int ans;
18.        ans="0";
19.        ans.len=len;
20.        if(x.len>ans.len) ans.len=x.len;
```

```
21.        for(int i=0;i<ans.len;i++){
22.            ans.a[i]+=a[i]+x.a[i];
23.            ans.a[i+1]=ans.a[i]/10;
24.            ans.a[i]%=10;
25.        }
26.        if(ans.a[ans.len]>0){
27.            ans.len++;
28.        }
29.        return ans;
30.    }
31.    big_int operator- (big_int &x){
32.        big_int ans;
33.        ans="0";
34.        ans.len=len;
35.        if(x.len>ans.len) ans.len=x.len;
36.        for(int i=0;i<len;i++){
37.            ans.a[i]=a[i]-x.a[i];
38.            if(ans.a[i]<0){
39.                ans.a[i]+=10;
40.                a[i+1]--;
41.            }
42.        }
```

```
43.        while(ans.a[ans.len-1]==0 && ans.len>1){
44.            ans.len--;
45.        }
46.        return ans;
47.    }
48.    bool operator< (big_int &x){
49.        if(len<x.len) return true;
50.        else if(len>x.len) return false;
51.        else{
52.            for(int i=len-1;i>=0;i--){
53.                if(a[i]<x.a[i]) return true;
54.                else if(a[i]>x.a[i]) return false;
55.            }
56.            return false;
57.        }
58.    }
59.    bool operator>= (big_int &x){
60.        return !(*this<x);
61.    }
62.    void print(){
63.        for(int i=len-1;i>=0;i--){
64.            cout<<a[i];
65.        }
66.    }
67. };
```

```
68. big_int n,m,a;
69. string s;
70. int main(){
71.     cin>>s;
72.     n=s;
73.     cin>>s;
74.     m=s;
75.     if(n>=m){
76.         a=n-m;
77.     }else{
78.         cout<<"-";
79.         a=m-n;
80.     }
81.     a.print();
82.     return 0;
83. }
```

谢谢！

—