

高精度

无论是32位还是64位整数，都有大小的范围限制。高精度算法主要解决的就是超过C++内置整型的运算。在求大整数运算的时候，由于输入数字太大，我们需要以字符串形式读入；输出同理。我们假定输入里没有前导零（前导零的意思是，一个非零数的最高位的那些零，例如0030这个数的左起的两个零。这些零在最终答案中不应该出现）。

本文主要考虑高精度四则运算。其中高精度乘法和除法，只考虑一个大数乘以或者除以一个较小数。

高精度加法

直接模拟人工加法，人工做加法的时候，是由低位向高位，逐位相加；某一位相加超过10，就要考虑进位，下一位要把进位加上。例题：https://blog.csdn.net/qq_46105170/article/details/113793376。看代码：

```
1  #include <algorithm>
2  #include <iostream>
3  #include <string>
4
5  using namespace std;
6
7  string add(auto& a, auto& b) {
8      string s;
9      s.reserve(max(a.size(), b.size()) + 1);
10
11     // t存储进位
12     for (int i = a.size() - 1, j = b.size() - 1, t = 0; i >= 0 || j >= 0 || t;) {
13         int sum = t;
14         if (i >= 0) sum += a[i--] - '0';
15         if (j >= 0) sum += b[j--] - '0';
16         s.push_back(sum % 10 + '0');
17         t = sum / 10;
18     }
19
20     reverse(s.begin(), s.end());
21     return s;
22 }
23
24 int main() {
25     string a, b;
26     getline(cin, a);
27     getline(cin, b);
28 }
```

```
29     cout << add(a, b) << endl;
30 }
```

高进度减法

一样，也要模拟人工做减法的流程。但与高精度加法不同的是，人工做减法的时候，我们总是用大数减去小数。所以在读入输入 a 和 b 之后，首先要判断大小。如果 $a < b$ ，那么我们要求的是 $b - a$ ，当然最后答案之前也要加上负号。此外还需要注意：

1. 与加法类似，做减法的时候也需要从最低位做起
2. 当前位做减法的时候要注意考虑之前的退位，当前位做减法的时候如果变为负数，也要记录退位，在做下一位减法的时候考虑进去
3. 最后答案可能会出现前导零，要注意去掉。

例题：https://blog.csdn.net/qq_46105170/article/details/113793404。

```
1  #include <algorithm>
2  #include <iostream>
3  #include <string>
4  using namespace std;
5
6  // 比较a和b这两个数哪个大。注意这里已经假定输入数里没有前导零了
7  bool cmp(auto& a, auto& b) {
8      if (a.size() != b.size()) return a.size() >= b.size();
9      return a >= b;
10 }
11
12 string sub(auto& a, auto& b) {
13     string c;
14     c.reserve(a.size());
15
16     // 从最低位开始做减法。t是退位
17     for (int i = 0, t = 0; i < a.size(); ++i) {
18         int x = a[a.size() - 1 - i] - '0';
19         int y = (i < b.size() ? b[b.size() - 1 - i] - '0' : 0);
20         // 当前位做减法要把退位考虑进去
21         int diff = x - y - t;
22
23         if (diff < 0) {
24             diff += 10;
25             t = 1;
26         } else
27             t = 0;
28
29         c += diff + '0';
```

```

30     }
31     // 去掉前导零
32     while (c.size() > 1 && c.back() == '0') c.pop_back();
33     // c是逆序的答案，还要翻转回来
34     reverse(c.begin(), c.end());
35     return c;
36 }
37
38 int main() {
39     string a, b;
40     getline(cin, a);
41     getline(cin, b);
42
43     cout << (cmp(a, b) ? sub(a, b) : "-" + sub(b, a)) << endl;
44 }

```

高精度乘法

两个超大整数相乘的高效算法，是一个非常难的问题。假设两个整数的长度都是 n ，如果按照手工乘法的方法来做，那么就要做 $O(n^2)$ 次乘法和加法，从而时间复杂度是 $O(n^2)$ 。但是存在更快的算法，比如Karatsuba算法可以做到时间复杂度大约 $O(n^{1.5})$ ，还有更快的快速傅里叶变换的算法。事实上两个 n 位数相乘时间复杂度最低可以到哪里，是个未解决问题。我们只考虑一个 n 位数和一个较小数相乘的问题。

可以模拟人工的乘法，从大数的最低位开始，接着把那个小数与大数的各个位相乘。与加法类似，考虑进位的影响。比如考虑 1289×12 ，从大数的最低位开始：

先算 $9 \times 12 = 108$ ，所以个位是108模10，也就是8，进位是10；

再算 $8 \times 12 + 10 = 106$ ，所以十位是106模10也就是6，进位是10；

再算 $2 \times 12 + 10 = 34$ ，百位是4，进位3；

最后 $1 \times 12 + 3 = 15$ ，千位是5，进位1；

最后把进位填到最高位上去。

例题：https://blog.csdn.net/qg_46105170/article/details/113793743

```

1  #include <algorithm>
2  #include <iostream>
3  #include <string>
4  using namespace std;
5
6  string mult(string& a, int b) {
7      if (b == 0) return "0";
8

```

```

9   string res;
10
11  int t = 0;
12  for (int i = a.size() - 1; i >= 0; i--) {
13      int prod = (a[i] - '0') * b + t;
14      res.push_back(prod % 10 + '0');
15      t = prod / 10;
16  }
17
18  while (t) {
19      res.push_back(t % 10 + '0');
20      t /= 10;
21  }
22
23  reverse(res.begin(), res.end());
24  return res;
25 }
26
27 int main() {
28     string a;
29     int b;
30
31     cin >> a >> b;
32     cout << mult(a, b) << endl;
33 }

```

高精度除法

两个大整数除法是个更加难的问题。我们只考虑一个大整数除以另一个小整数，并且我们要求除出的商和余数。

由于四则运算里，只有人工除法是从高位做起的，所以我们在程序里处理被除数的时候，也是从高位做起。我们回忆竖式除法，我们的做法是，先考虑最高位，从高位开始除，把商写在上面，然后把余数写在下面，然后考虑下一位，下一位加上刚才得到的余数乘以10，成为新的被除数，再走相同的流程。

例如我们考虑 $1289/12$ ，先初始化余数 $r = 0$ ，然后从左向右考虑每一位

先考虑1，每次用余数乘以10加当前数作为新的被除数，现在得到 $0 \times 10 + 1 = 1$ ，其除以12商等于0，余1，令 $r = 1$ ；

接着考虑2，新被除数为 $1 \times 10 + 2 = 12$ ，除以12商等于1余0，令 $r = 0$ ；

接着考虑8，新被除数为 $0 \times 10 + 8 = 8$ ，除以12商等于0余8，令 $r = 8$ ；

再考虑9，新被除数为 $8 \times 10 + 9 = 89$ ，除以12商等于7余5，令 $r = 5$ ；

每一步得到的商连起来就是最终的商，即 $0107 = 107$ ，余数是5。

最后答案记得去掉前导零。

例题：https://blog.csdn.net/qq_46105170/article/details/113794074

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  // b是除数, r是余数
6  string div(string& a, int b, int& r) {
7      string res;
8      r = 0;
9
10     for (int i = 0; i < a.size(); i++) {
11         r = r * 10 + (a[i] - '0');
12         res.push_back(r / b + '0');
13         r %= b;
14     }
15
16     int i = 0;
17     while (i + 1 < res.size() && res[i] == '0') i++;
18     return res.substr(i);
19 }
20
21 int main() {
22     string a;
23     int b;
24     cin >> a >> b;
25
26     int r;
27     cout << div(a, b, r) << '\n' << r << endl;
28 }
```

作业

上面所有例题