信息学常见题目类型-模拟

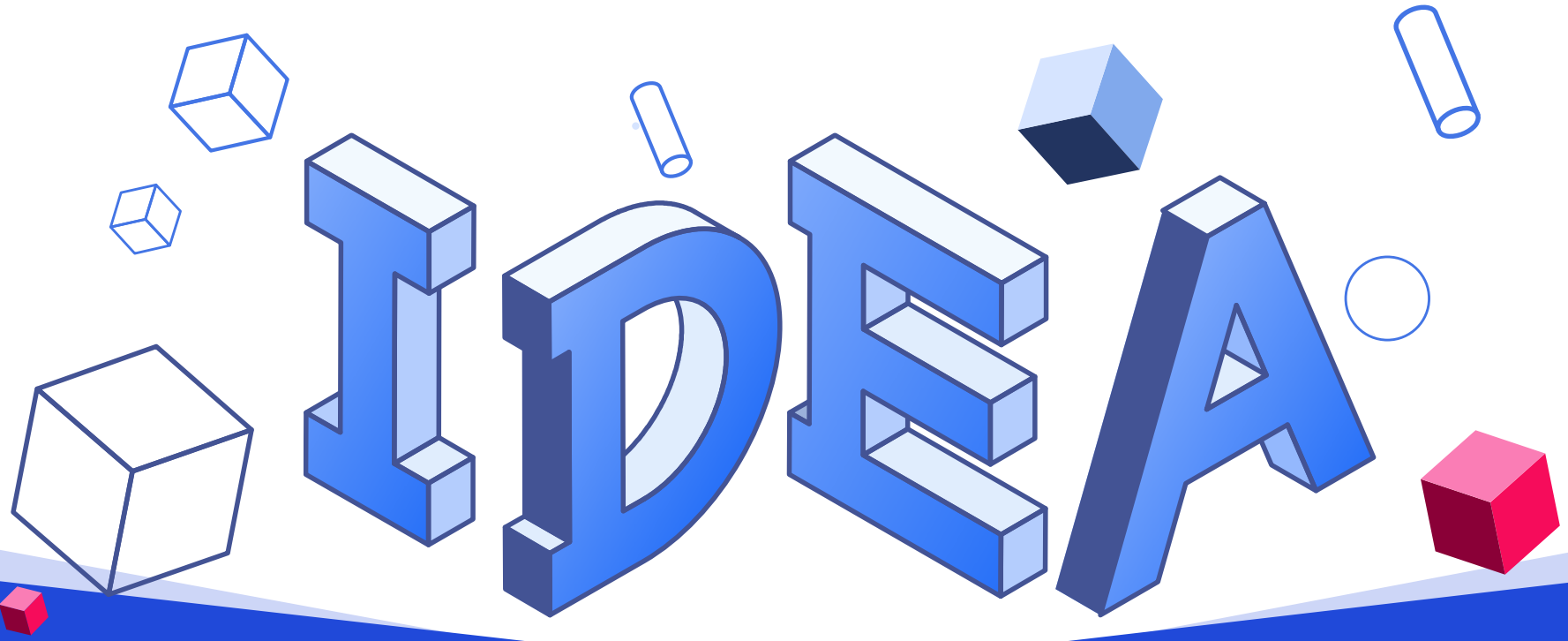
模拟题

是一类题目将解决的办法或者规则告诉你了，需要你编程实现的题目

大家已经做过一些了，例如：幻方、马鞍数等

这一类题目较为考验大家的代码基本功，许多选手都将模拟题作为自己代码能力提升的途径之一

接下来将会了解一个经典的模拟类型



信息学

高精度算法

西南大学附属中学校

信息奥赛教练组

高精度算法的作用

- **模拟人计算的过程，是一个解决竞赛问题的基本工具**
- **竞赛通常不会单独考察，往往是在某些题目里涉及到数值很大的时候，用高精度计算**

一般参与运算的数字位数很大，long long肯定存不下

高精度算法

高精度计算通用方法:

- 高精度计算时一般用一个数组来存储一个数,数组的一个元素对应于数的一位
- 由于数计算时可能要进位,因此为了方便,将数由低位到高位依次存在数组下标对应由低到高位置上,另外,我们申请数组大小时,一般考虑了最大的情况,
- 在很多情况下, 有富余,即高位有很多0,可能造成无效的运算和判断,因此,我们一般将数组的第0个下标对应位置来存储该数的位数.如数:3485，表达在数组a[10]上情况是:

下标	0	1	2	3	4	5	6	7	8	9
内容	4	5	8	4	3	0	0	0	0	0
说明	位数	个位	十位	百位	千位					

a[0]存储位数

字符串存储

```
#include <bits/stdc++.h>
using namespace std;
const int N=10001; //最多10000位
int main(){
  int a[N],b[N],i;
  string s1,s2;
  cin>>s1; //数s1
  cin>>s2; //数s2
  memset(a,0,sizeof(a)); //数组清0
  memset(b,0,sizeof(b));
  a[0]=s1.length(); //一般在下标0存储位数
  b[0]=s2.length();
  for(i=1;i<=a[0];i++)
    a[i]=s1[a[0]-i]-'0’;//将字符转为数字并倒序存储
  for(int i=1;i<=b[0];i++)
    b[i]=s2[b[0]-i]-'0';

  return 0;
}
```

memset(数组地址,赋的值,数组大小)

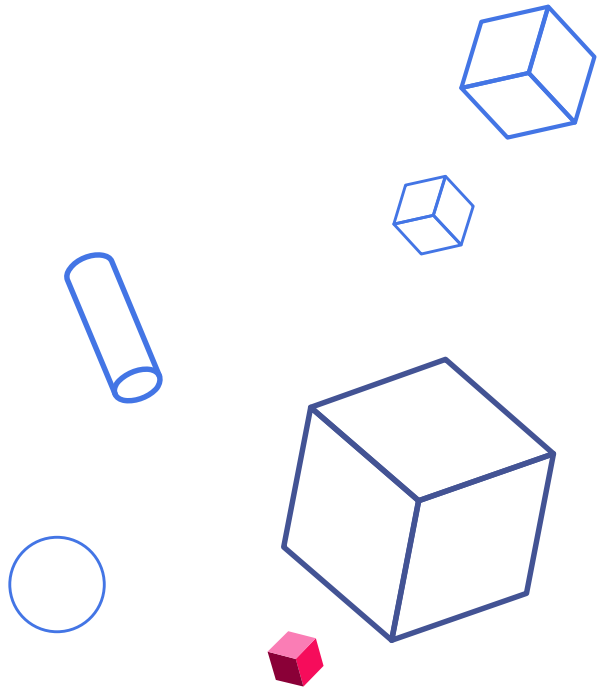
memset按字节赋值，常用的几个赋值：
0：赋值为0
-1：赋值为-1
127：接近int上限的正数
128：接近int下限的负数

模块化编程(函数)

```
#include <bits/stdc++.h>
using namespace std;
const int N=10001; //最多存10000位
int a[N],b[N];
string s1,s2;
int main(){
  int i;
  init();
  return 0;
}
```

```
void init(){
  cin>>s1; //数s1
  cin>>s2; //数s2
  memset(a,0,sizeof(a)); //数组清0
  memset(b,0,sizeof(b));
  a[0]=s1.length(); //一般在下标0存储位数
  b[0]=s2.length();
  for(int i=1;i<=a[0];i++)  //将字符转为数字并倒序存储
    a[i]=s1[a[0]-i]-'0';
  for(int i=1;i<=b[0];i++)
    b[i]=s2[b[0]-i]-'0';
}
```

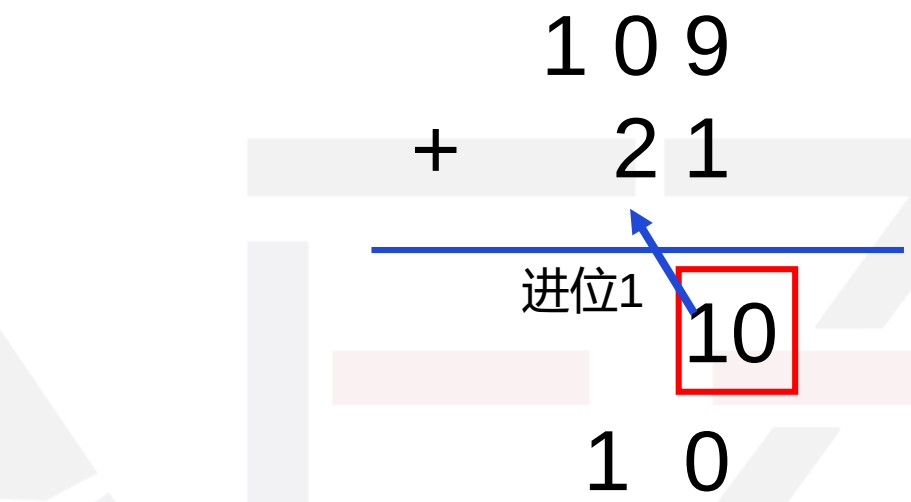
高精度加法



加法的过程

模拟列竖式加法过程

核心：
1.对位相加
2.高位进位

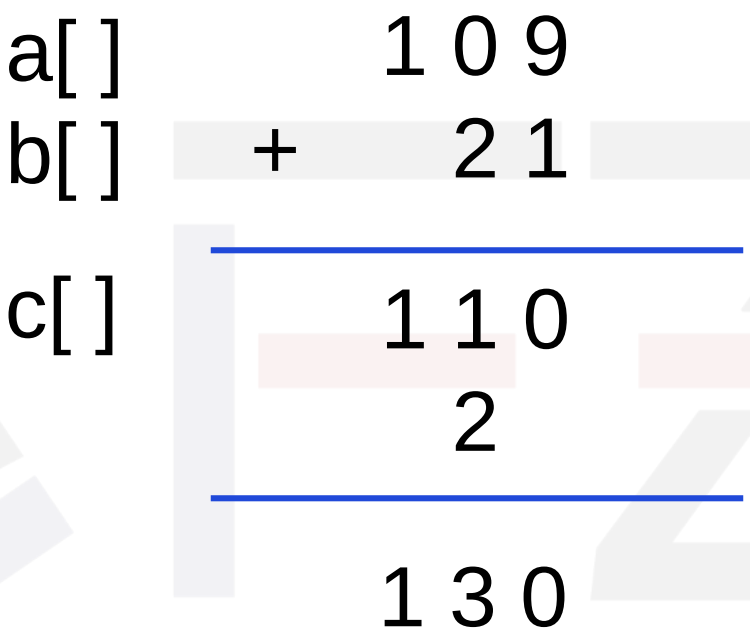


加法的过程

模拟列竖式加法过程

核心：
1.对位相加
2.进位

```
//对位相加
c[i]+=(a[i]+b[i]);
//进位
x=(a[i]+b[i])/10;
c[i]%=10;
c[i+1]+=x;
```



个位相加

c[1]	c[2]	c[3]
10	0	0

高位进位

c[1]	c[2]	c[3]
0	1	0

十位相加
无进位

c[1]	c[2]	c[3]
0	3	0

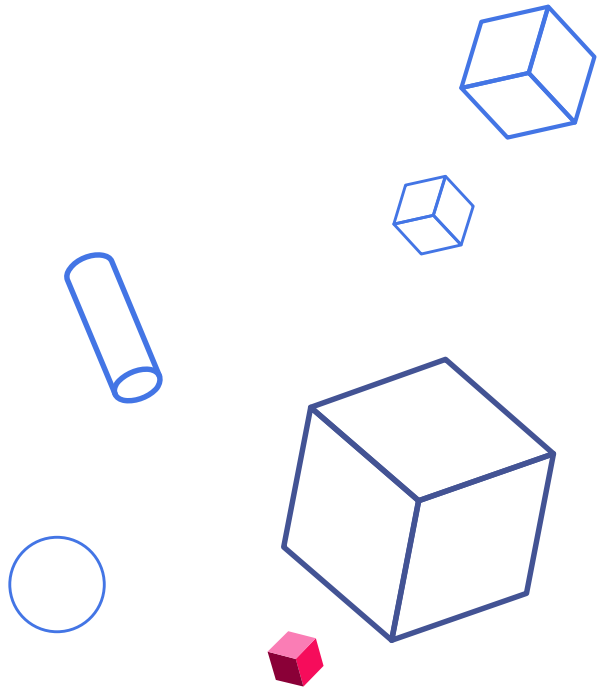
百位相加
无进位

c[1]	c[2]	c[3]
0	3	1

加法函数jia()

```
int c[N+1];
void jia() //计算c=a+b
  int i,k;
  k=max(a[0],b[0]);
  int x=0;
  for(i=1;i<=k;i++) {
     c[i]+=a[i]+b[i];
     x=c[i]/10;
     c[i+1]+=x;
     c[i]%=10;
  }
  if(c[k+1]>0) c[0]=k+1;  //修正和的位数（a+b最多只能的一个进位）
  else c[0]=k;
}
```

高精度减法



模块化编程(函数)

```
#include <bits/stdc++.h>
using namespace std;
const int N=10001; //最多存10000位
int a[N],b[N];
string s1,s2;
int main(){
  int i;
  init();
  return 0;
}
```

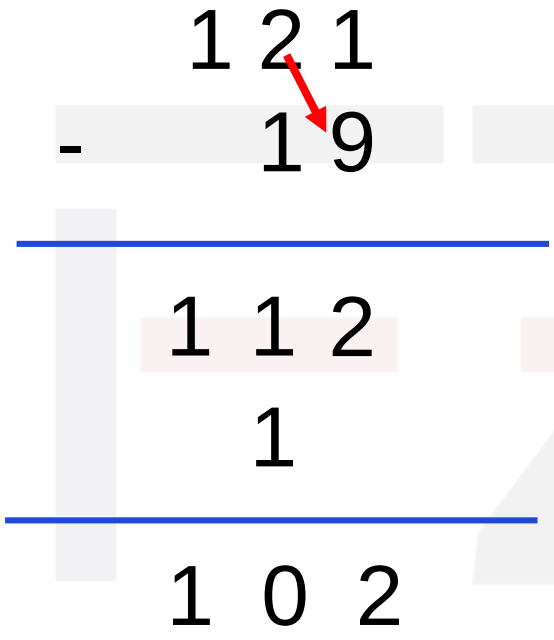
```
void init(){
  cin>>s1; //数s1
  cin>>s2; //数s2
  memset(a,0,sizeof(a)); //数组清0
  memset(b,0,sizeof(b));
  a[0]=s1.length(); //一般在下标0存储位数
  b[0]=s2.length();
  if(a[0]<b[0] || a[0]==b[0]&&s1<s2) {  //减后是负数
   cout<<“-”;
   swap(s1,s2),swap(a[0],b[0]);
  }
  for(int i=1;i<=a[0];i++)  //将字符转为数字并倒序存储
    a[i]=s1[a[0]-i]-'0';
  for(int i=1;i<=b[0];i++)
    b[i]=s2[b[0]-i]-'0';
}
```

减法的过程

模拟列竖式减法过程

核心：
1.对位相减
2.不够，高位借数

```
//对位相减
c[i]+=(a[i]-b[i]);
//高位借数
c[i+1]--;
c[i]+=10;
```



个位相减

c[1]	c[2]	c[3]
-8	0	0

高位借位

c[1]	c[2]	c[3]
2	-1	0

十位相减
无借位

c[1]	c[2]	c[3]
2	0	0

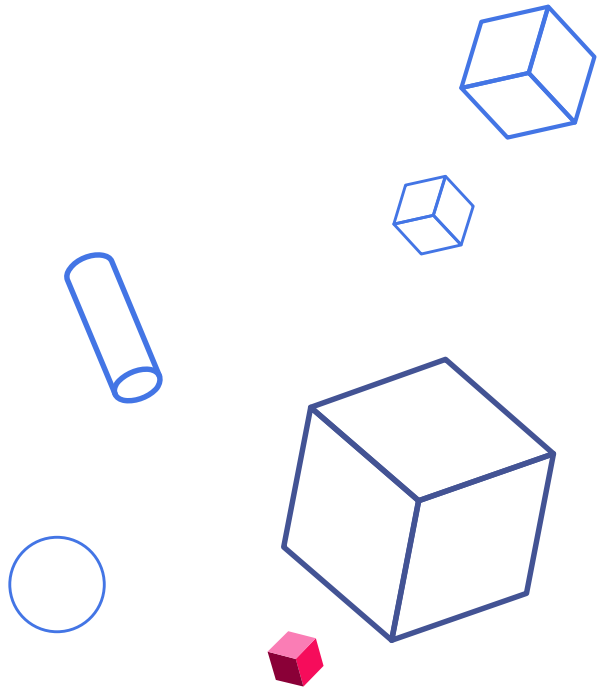
百位相减
无借位

c[1]	c[2]	c[3]
2	0	1

减法函数jian()

```
int jian(); {  //计算c=a-b
  int i,k;
  k=a[0];
  for(i=1;i<=k;i++){
     c[i]+=a[i]-b[i];
     if(c[i]<0){ c[i+1]--;c[i]+=10;} //若不够减则向上借一位
   }
    while(c[k]==0 && k>1) k--; //修正c的位数
    c[0]=k;
}
```

高精度乘法



乘法过程

模拟列竖式乘法过程

核心：
1. 按位相乘
2. 乘积相加，高位进位

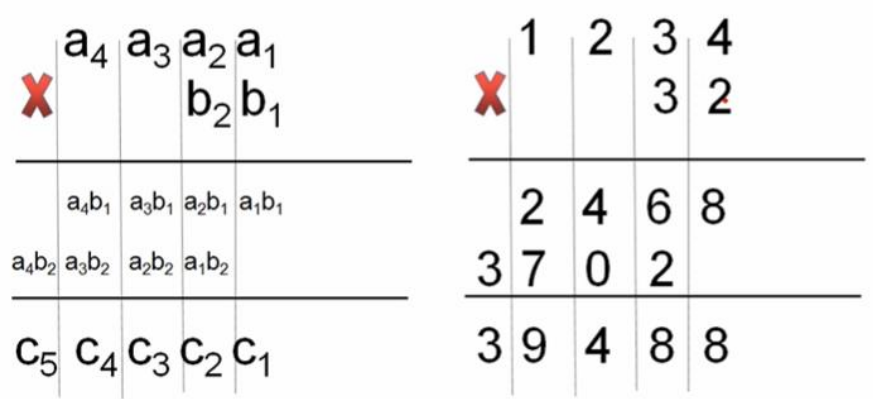
例如： 666 * 789 == 525474

```
     6   6   6
   *   7   8   9
  ----------------------    //从0开始数
     54  54  54       //拿b的 0位去乘 a的0 1 2位，存入c的0 1 2 位
   48  48  48         //拿b的 1位去乘 a的0 1 2位，存入c的1 2 3 位
 42  42  42           //拿b的 2位去乘 a的0 1 2位，存入c的2 3 4 位
----------------------
 42  90 144 102  54       //相加
----------------------
5 2   5   4   7   4       //进位
```

乘法过程

模拟列竖式乘法过程

核心：
1. 按位相乘
2. 乘积相加，高位进位



观察下标关系，得到：

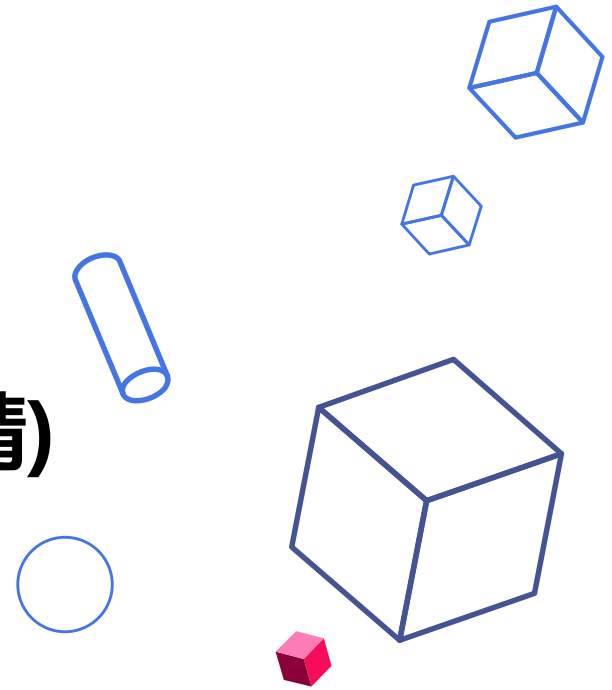
```
c[i+j-1]+= a[i]*b[j]+ x;
x=c[i+j-1]/10 ;
c[i+j-1]%=10
```

x：上一项的进位

乘法函数cheng()

```
void cheng(){
  for (i=1;i<=a[0];i++){
    x=0; //用于存放进位
   for (j=1;j<=b[0];j++) { //对乘数的每一位进行处理
     c[i+j-1]+=a[i]*b[j]+x;  //当前乘积+上次乘积进位+原数
     x=c[i+j-1]/10;
     c[i+j-1] %= 10;
   }
    c[i+b[0]]=x; //高位进位
 }
 c[0]=a[0]+b[0];
 int k=c[0];
 while (c[k]==0&&k>1)        //删除前导0
          k--;
 c[0]=k;
}
```

高精度除法(高精除单精)



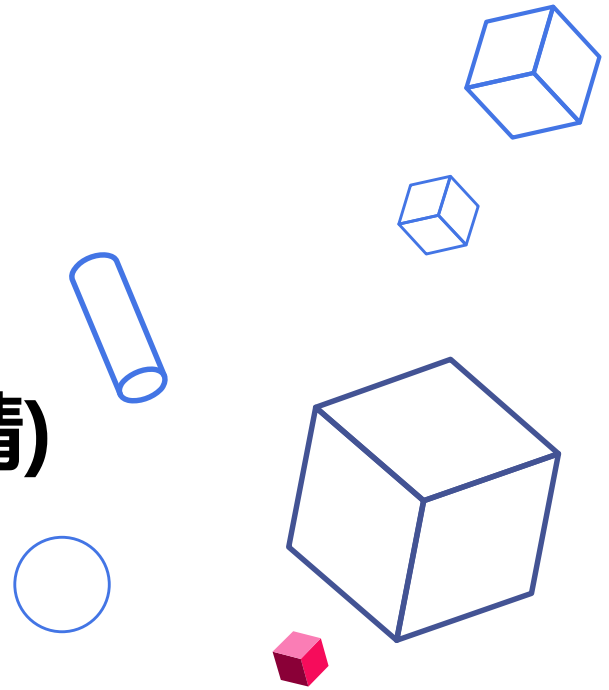
除法的过程

做除法时，每一次的商的值都在0~9，每次求得的余数连接以后的若干位得到新的被除数，继续做除法。因此在做高精度除法时，要涉及到乘法运算和减法运算以及移位处理。

核心：按位相除

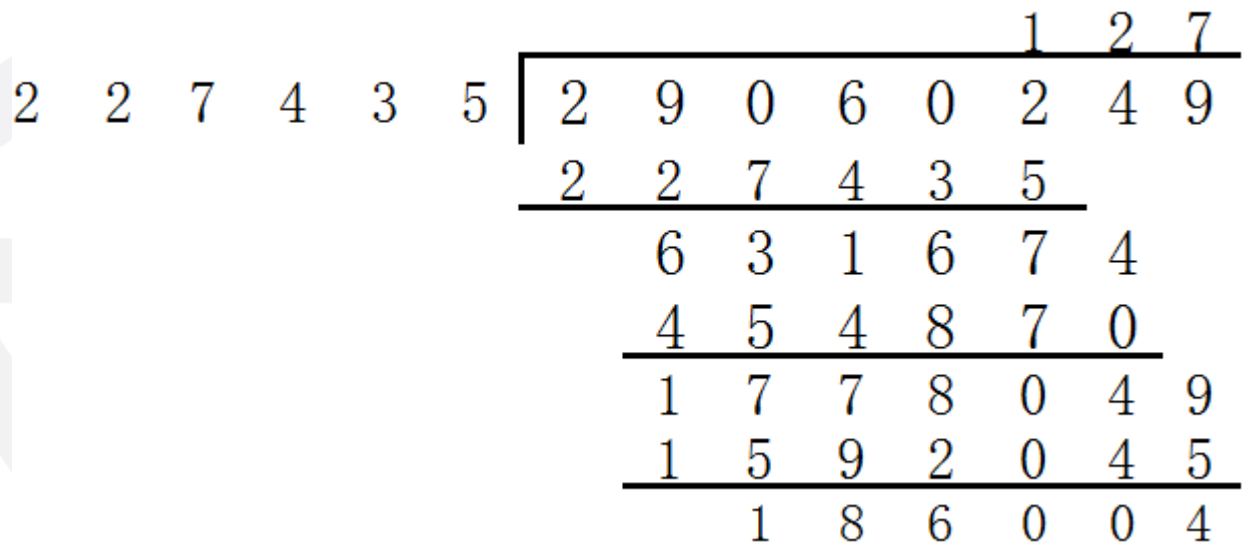
```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char str[100];
    int a[100],b,c[100];
    int lena,lenc;
    int x=0;
    int i;
    memset(a,0,sizeof(a));
    memset(c,0,sizeof(c));
    cin>>str;
    cin>>b;
    lena=strlen(str);
    for(i=0;i<lena;i++)
        a[i+1]=str[i]-'0';

    for(i=1;i<=lena;i++) {//按位相除
       c[i]=(x*10+a[i])/b;
        x=(x*10+a[i])%b;
    }

    lenc=1;
    while(c[lenc]==0&&lenc<lena)//删除前导0
        lenc++;
    for(i=lenc;i<=lena;i++)//倒序输出
        cout<<c[i];
    cout<<endl;

    return 0;
}
```

高精度除法(高精除高精)



除法的过程

采用计算机做高精度除法时，模拟日常除法的步骤。



分析

但计算机不可能做“试商”，这时，我们可以采用减法来模拟"试商"的过程。
算法的步骤如下：

1、将除数移动和被除数对齐，位数不够时，补0，

2、利用被除数减去除数，一直减到被除数小于除数，减的次数，就是“试商”的结果。

3、重复上述步骤，一直到被除数和除数的位数相等为止。

核心：利用减法来模拟除法

结合下发的代码，理解高精除高精的过程

参考代码

```
#include<iostream>
#include<cstring>
using namespace std;
int a[100],b[100],c[100];
int compare(int a[],int b[])//比较a、b，若a>b为1；若a<b为-1；若a=b为0
{
    int i;
    if(a[0]>b[0])
        return 1;
    if(a[0]<b[0])
        return -1;
    for(i=a[0];i>0;i--)//从高位到低位比较
    {
        if(a[i]>b[i])
            return 1;
        if(a[i]<b[i])
            return -1;
    }
    return 0;
}
```

```
void subduction(int a[],int b[])//计算a=a-b
{
    int flag;
    int i;

    flag=compare(a,b);
    if(flag==0)//相等
    {
        a[0]=0;
        return;
    }
    if(flag==1)//大于
    {
        for(i=1;i<=a[0];i++)
        {
            if(a[i]<b[i])//若不够向上借位
            {
                a[i+1]--;
                a[i]+=10;
            }
            a[i]-=b[i];
        }
        while(a[0]>0&&a[a[0]]==0)//删除前导0
            a[0]--;
        return;
    }
}
```

```
int main(){
    char str1[100],str2[100];
    int i,j;
     memset(a,0,sizeof(a));
    memset(b,0,sizeof(b));
    memset(c,0,sizeof(c));
     cin>>str1>>str2;
    a[0]=strlen(str1);//a[0]存储串1的位数
    b[0]=strlen(str2);//b[0]存储串2的位数
    for(i=1;i<=a[0];i++)
        a[i]=str1[a[0]-i]-'0';
    for(i=1;i<=b[0];i++)
        b[i]=str2[b[0]-i]-'0';
    int temp[100];
    c[0]=a[0]-b[0]+1;
    for(i=c[0];i>0;i--){
        memset(temp,0,sizeof(temp));

        for(j=1;j<=b[0];j++)//从i开始的地方，复制数组b到数组temp
            temp[j+i-1]=b[j];
        temp[0]=b[0]+i-1;

        while(compare(a,temp)>=0)//用减法模拟
        {
            c[i]++;
            subduction(a,temp);
        }
    }
     while(c[0]>0&&c[c[0]]==0)//删除前导0
        c[0]--;

    cout<<"商为：";
    if(c[0]==0)//输出结果
        cout<<0<<endl;
    else{
for(i=c[0];i>0;i--)
            cout<<c[i];
        cout<<endl;
    }

    cout<<"余数为：";
    if(a[0]==0)//输出余数
        cout<<0<<endl;
    else
    {
        for(i=a[0];i>0;i--)
            cout<<a[i];
        cout<<endl;
    }

    return 0;
}
```

Thanks

For Your Watching