乘法的难点在于要使用一个嵌套循环，除法的难点在于一个字使用符串比较方法的技巧，本次还是会将算法都写成函数，然后在main()函数中调用，原因是在第四篇我们要将整个大数运算的方法做成自己的一个库文件，可以供自己或他人使用。

大数乘法：   
由于乘法可以互换所以对于输入的数字没有限制条件，计算方法还是模仿手工算法，由被乘数的低位开始和乘数的每一位相乘并且要将大于９的十位数向前进一位，存在３个问题需要我们解决。

问题：   
１　我们要用多大的数组存储结果？   
２　要使用嵌套循环吗？   
３　如何在计算的过程中保证进位？

其实问题也很好解决，前两个问题都可以看出答案，最后一个问题和前两篇博客的进位问题很相似，所以简单的说明后再看注释的代码是很好懂的。   
１　二个数相乘最大的位数是两个乘数的位数之和。   
２　很明显由于乘法的特性使用嵌套循环很合适。   
３　在大数加减中执行完毕后再对存储结果的result数组进行一次进位，但在乘法中我们需要每执行一趟就要对数组进行进位的处理。

1 //#include"big.h" 2 #include<stdio.h> 3 #include<stdlib.h> 4 #include<string.h> 5 char \* bigmul(char \*m,int lena,char \*f,int lenb){ //乘法运算函数。 6 int i,j,k,lensum,tmp\_result,carry,num='0'; 7 lensum=lena+lenb; //确定结果数组的长度。 8 for(i=0;i<lena;i++){ //将ASCII码转为对应的数字存储。 9 m[i]=m[i]-num; 10 } 11 for(i=0;i<lenb;i++){ 12 f[i]=f[i]-num; 13 } 14 char \*result,final[BUFSIZ]; 15 result=(char\*)calloc(lensum,1); 16 for(i=0;i<lenb;i++){ //为被乘数作一趟乘法。 17 for(j=0;j<lena;j++){ 18 tmp\_result=f[lenb-i-1]\*m[lena-j-1]; 19 result[j+i]+=tmp\_result; 20 } 21 for(k=0;k<=j+i-1;k++){ //每作一趟乘法整理一次结果数组。 22 if(result[k]>9){ 23 carry=result[k]/10; 24 result[k]=result[k]%10; 25 result[k+1] += carry; 26 } 27 } 28 } 29 j=0; 30 if(result[lensum-1]!=0){ //去除前导零将结果整理到final数组中。 31 final[j]=result[lensum-1]+num; 32 j++; 33 } 34 for(i=lensum-2;i>=0;i--){ 35 final[j++]=result[i]+num; 36 } 37 result=final; //将指针指向final数组并返回该指针。 38 return result; 39 } 40 int main(){ //利用main测试方法，用puts打印结果。 41 int lena,lenb; 42 char \*result,sa[BUFSIZ],sb[BUFSIZ]; 43 scanf("%s",sa); 44 scanf("%s",sb); 45 lena=strlen(sa); 46 lenb=strlen(sb); 47 result=bigmul(sa,lena,sb,lenb); 48 puts(result); 49 50 }