第九章 高精度运算（2）

1. 高精度乘法

1.1 高精度数乘低精度数

高精度数乘低精度数的过程就是从高精度数的个位开始逐位数字与低精度数相乘， 相乘的结果模 10作为本位的结果，而后除以 10 的商作为下一位的进位
下面是高精度数乘低精度数的步骤：

1. 按照字符串形式输入高精度数，按整型输入低精度数
2. 将字符串逐位字符转换成数字并逆序存放在整型数组中
3. 从高精度数的个位开始 （即下标位置） 逐位数字与低精度数相乘，相乘的结果模 10作为 本位的结果， 而后除以 10 的商作为下一位的进位；
4. 重复第三步， 直到高精度数的最高位与低精度数乘完后结束；
5. 从后往前找结果数组中第一个非0元素的位置 （即去前导 0操作），从该位置开始逆序输 出数组中的元素，即为最终的乘法结果

- 乘法竖式 56789 * 8 的过程

```
   5 6 7 8 9
*          8
-----------
       7 2
     7 1
   6 3
 5 4
-----------
 4 5
-----------
 4 5 4 3 1 2
```

个位：9 * 8 = 72，结果为 2，进位为 7
十位：8 * 8 + 7 = 71，结果为 1，进位为 7
百位：7 * 8 + 7 = 63，结果为 3，进位为 6
千位：6 * 8 + 6 = 54，结果为 4，进位为 5
万位：5 * 8 + 5 = 45，结果为 5，进位为 4

- 高精度数乘低精度数的过程

1．按照字符串形式存储高精度数；

```
string s1 = “56789”;
int b = 8;
```

2．将字符串翻转存入整型数组中；

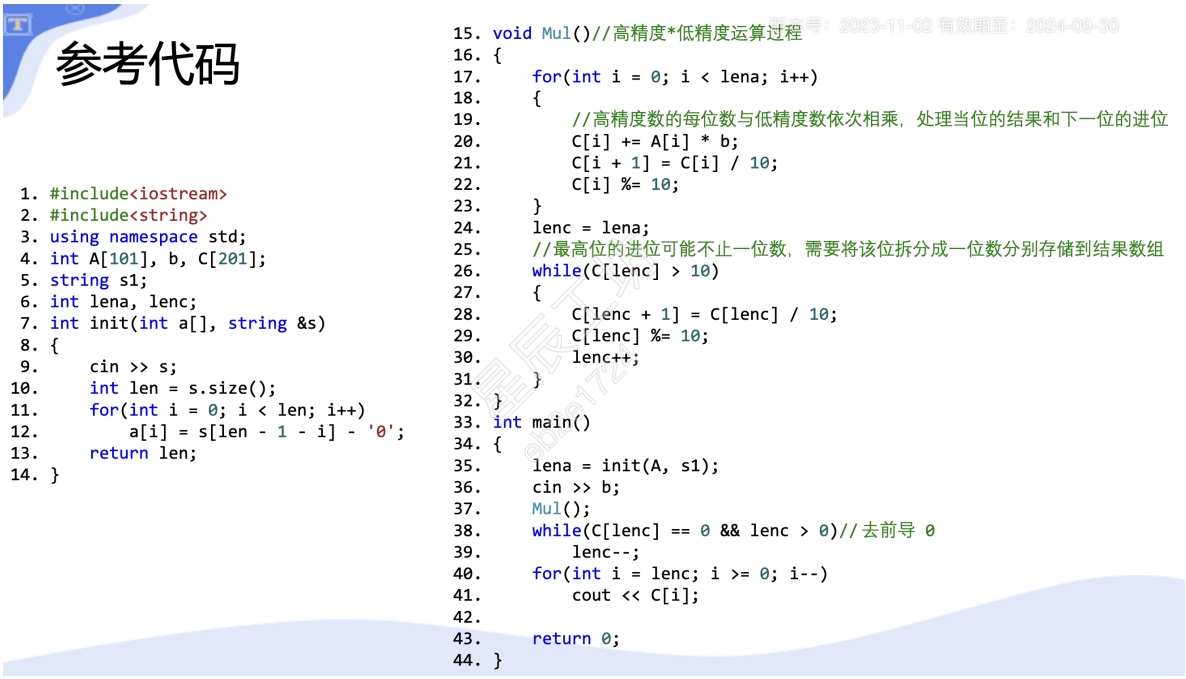
```
int A[100] = {9, 8, 7, 6, 5} , C[201] = { } (存储结果数组) ;
```

3．从低位到高位逐位数字与 b 相乘，相乘的结果模 10 作为本位的结果，而后除以 10的商作为下一位的进位。

$$\begin{array}{r} A_4\ A_3\ A_2\ A_1\ A_0 \\ \times\qquad\quad b \\ \hline C_5\ C_4\ C_3\ C_2\ C_1\ C_0 \end{array}$$

4．将结果数组逆序输出即为结果。

- 参考代码




```cpp
#include<iostream>
#include<string>
using namespace std;
int A[101], b, C[201];
string s1;
int lena, lenc;
int init(int a[], string &s)
{
    cin >> s;
    int len = s.size();
    for(int i = 0; i < len; i++)
        a[i] = s[len - 1 - i] - '0';
    return len;
}
void Mul()//高精度*低精度运算过程
{
    for(int i = 0; i < lena; i++)
    {
        //高精度数的每位数与低精度数依次相乘，处理当位的结果和下一位的进位
        C[i] += A[i] * b;
        C[i + 1] = C[i] / 10;
        C[i] %= 10;
    }
    lenc = lena;
    //最高位的进位可能不止一位数，需要将该位拆分成一位数分别存储到结果数组
    while(C[lenc] > 10)
    {
        C[lenc + 1] = C[lenc] / 10;
        C[lenc] %= 10;
        lenc++;
    }
}
int main()
{
    lena = init(A, s1);
    cin >> b;
    Mul();
    while(C[lenc] == 0 && lenc > 0)//去前导 0
        lenc--;
    for(int i = lenc; i >= 0; i--)
        cout << C[i];

    return 0;
}
```

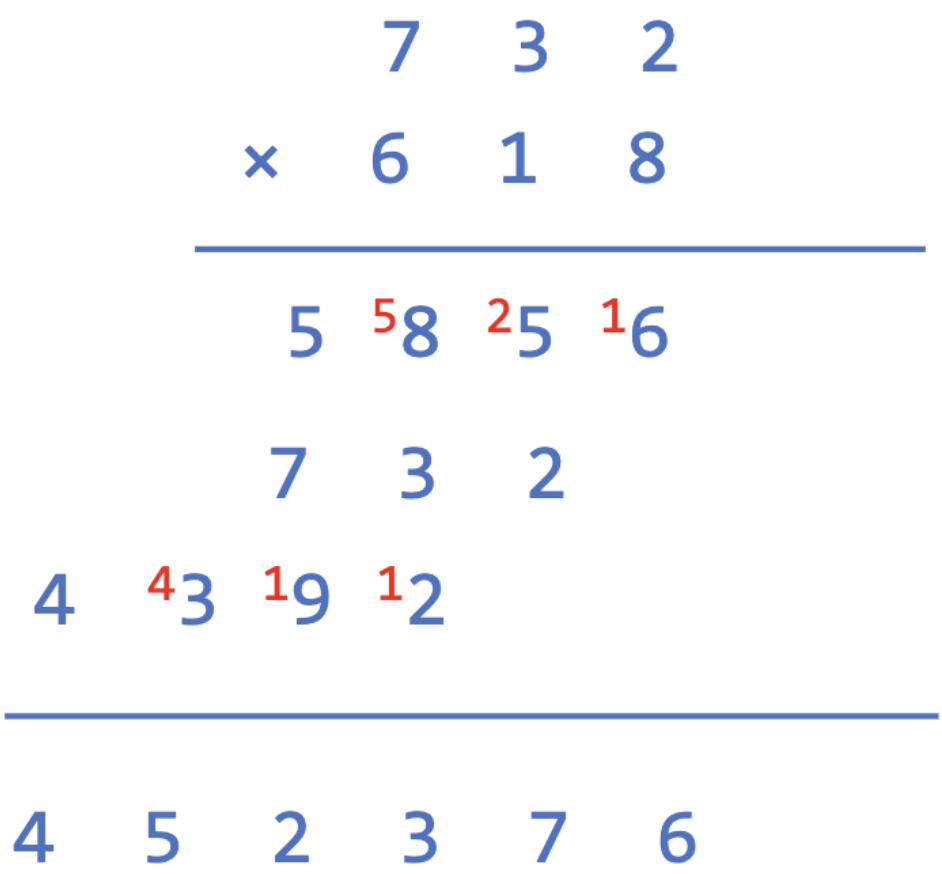
1.2 高精度数乘高精度数

两个高精度数 A*B 的运算就是利用列竖式乘法的过程，从个位开始用 A 的每一位数字分别乘以 B 的每一位数字，即 $A_{i(0...lenA-1)}$ * B_0、$A_{i(0...lenA-1)}$ * B_1、$A_{i(0...lenA-1)}$ * B_2……

具体步骤如下：

1. 将两个高精度数字以字符串的形式输入。
2. 将字符串翻转存入整型数组中，两数乘积的位数最大为两个因数的位数和；
3. 从个位开始分别计算 $A_{i(0...lenA-1)}$ * B_0，将结果存入 C_i 位置，$Ai_{(0...lenA-1)}$ * B_1，将结果累加存入 C_{i+1} 位置，$A_{i(0...lenA-1)}$ * B_2，将结果累加存入 C_{i+2} 位置……以此类推，同时处理每一步计算后的进位；
4. 重复步骤 3，直到所有位数相乘完毕；
5. 从后往前找结果数组中第一个非 0 元素的位置（即去前导 0 操作），从该位置开始逆序输出数组中的元素，即为最终的乘法结果。

- 乘法竖式732 * 618的过程：



用高精度数乘高精度数计算 732 * 618 的结果。

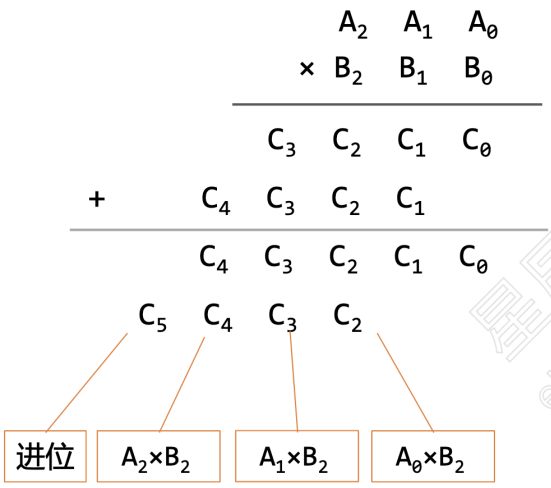
1. 将两个高精度数字以字符串的形式输入；

   string a = “732”, b = “618”;

2. 将字符串翻转存入整型数组中；

   int A[100] = {2, 3, 7}, B[100] = {8, 1, 6}, C[201] = { }（存储结果数组）；

3. 从个位开始分别计算 $A_{i(0...lenA-1)} * B_0$，将结果存入 C_i 位置，$Ai_{(0...lenA-1)} * B_1$，将结果累加存入 C_{i+1} 位置，$A_{i(0...lenA-1)} * B_2$，将结果累加存入 C_{i+2} 位置.....以此类推，同时处理每一步计算后的进位；



取下一位 B_2，和 A 相乘

$A_i \times B_2$ 的乘积放在乘积数组 C[i+2] 处，最高位产生的进位在 C[lenA+2] 内，最后将 C 数组逆序输出（先排除前导 0 后输出），即为最终的乘法结果。

- 参考代码

```cpp
#include<iostream>
#include<string>
using namespace std;
int A[201], B[201], C[202];
string a, b;
int lenA, lenB, lenC;
int init(int a[], string &s)
{
    cin >> s;//读入字符串
    int len = s.size();//计算数字串位数
    for(int i = 0; i< len; i ++)
    {   //将数字串逐位转换为数字并逆序存放
        a [i] = s[len - 1- i] - '0';
    }
    return len ;// 返回数字串长度
}
void Mul() //高精度数乘高精度数运算过程
{
    for(int i=0; i <lenB; ++i)
    {
        int m =0;
        // A[i]*B[j] 的结果累加存放在C[i+j]
        for(int j = 0; j <lenA; ++j)
        {
            C[i+j] = A[j] + B[i] + m + C[i+j];
            m = C[i+j] /10;
            C[i+j] %=10;
        }
        C[lenA + i] = m;

    }
}
int main()
{
    int lena = init(A, a);//计算A数字串长度
    int lenb = init(B, b);//计算B数字串长度

    Mul();
    while(C[lenc] == 0 && lenc > 0) lenc --; //去前导0
    for(int i =lenc; i>=0; i--) cout << C[i];  // 逆序输出结果
    return 0;
}
```

2 高精度除法

2.1 高精度除以低精度的商和余数

高精度数除以低精度数的过程就是将高精度数逐位拼接到上一位除法得到的余数末尾，再与低精度数相除，记录每一次除法得到的商和余数
下面是高精度数除以低精度数的步骤：

1. 按照字符串形式输入高精度数
2. 将字符串逐位字符转换成数字并存放在整型数组中 （高精度既不需要个位对齐，也不需要处 理进位，所以无需逆序存储），两数相除的位数最大为较大数的位数减较小数的位数加 1
3. 使用高精度数的最高位 （即下标0位置） 数字与低精度数相除，相除的结果即为本位的结果， 记录余数
4. 逐位处理高精度数的每一位数字，将其拼接到上一位除法运算得到的余数末尾 （即余数 *10+ 该位数字），使用新数与低精度数相除，相除的结果即为本位的结果，记录余数

5. 从前往后找结果数组中第一个非0 元素的位置 （即去前导 操作），从该位置开始正序输出数 组中的元素，即为最终的除法结果，并输出最后一次记录的余数

- 参考代码

```cpp
#include<iostream>
#include<string>
using namespace std;
int A[201], b, C[202];
int lenA, lenC, x;//x为余数
string s1;
int init(int a[], string &s)
{
    cin >> s;
    int len = s.size();
    for(int i = 0; i < len; i++)
    {
        a[i] = s[i] - '0';
    }
    return len;
}
void Div()//除法运算过程
{
 //每一次将 a 当位数字拼接到上一位余数的末尾去除 b
    for(int i = 0; i < lenA; i++)
    {
        x = x * 10 + A[i];
        C[i] = x / b;//本位的计算结果
        x = x % b;   //本位的余数
    }
}
```

```
int main()
{
    lenA = init(A, s1);
    cin >> b;
    Div();
    lenC = 0;
    while(C[lenC] == 0 && lenC < lenA - 1)
        lenC++;
    for(int i = lenC; i < lenA; i++)
        cout << C[i];
    cout << endl << x << endl;
    return 0;
}
```

例题：阶乘之和

题目描述：
使用高精度计算S=1！+ 2！+ 3！+ 4！+...+n! (n <= 50) 其中！表示阶乘，定义为 n! nx n-1）×（n-2）×..× 1 例如：5！=5×4×3×2×1= 120。
输入格式:
输入一个正整数n（n<=50）
输出格式:
输出一个正整数S，表示计算结果
输入样例
3
输出样例
9

- 参考代码