

## 第 3 章 高精度运算

### 3.1 高精度运算的基本知识

#### 一、什么是高精度运算？

##### 1、C++的基本类型和范围

类型	定义	大小	范围
char	字符型	1 byte	-128~127
short(int)	短整型	2 byte	-32768~32767 ( $-2^{15} \sim 2^{15}-1$ )
int	整形	4 byte	-2147483648 ~ 2147483647 ( $-2^{31} \sim 2^{31}-1$ )
long	长整形	4 byte(32bit 系统) 或 8 byte(64bit 系统)	参考 int 和 long long 的范围
long long	长整形	8 byte	-9223372036854775808 ~ 9223372036854775807 ( $-2^{63} \sim 2^{63}-1$ )
float	实型	4 byte	-(10 的 38 次方)~10 的 38 次方
double	双精度	8 byte	-(10 的 308 次方)~10 的 308 次方

计算机在存储时，使用的最小单位是“位”(bit)，一位，是用来存储一个 0 或者一个 1。

1byte = 8 位

#### 2、什么是高精度运算

高精度运算，是指参与运算的数(加数，减数，因子……)范围大大超出了标准数据类型(整型，实型)能表示的范围的运算。

例如，求两个 200 位的数的和。这时，就要用到高精度算法了。

#### 二、高精度运算的基本思路？

基本思路：

(1) 由于字符数组可以输入 n 位，因此采用字符串（或字符数组）读入 2 个高精度的数；

(2) 由于加减乘运算都需要从右向左运算（包括进位），而且要进行整数运算；因此，为了方便，我们将 2 个字符数组逆序存入 2 个整数数组；这样既可以从左向右运算（运算和进位），又可以按照整数格式进行运算，比较方便；

(3) 将计算结果存入第 3 个数组，然后按照要求逆序输出结果，就可以实现高精度运

算。

注意：考虑高精度减法、乘法运算中结果为 0 的情况。

## 3.2 高精度运算的实现

### 一、课堂案例

#### 1268: 【基础】高精度加法

思路图解：

以：968+ 94 为例，数学运算的写法为

#### 加法的数学实现

		9	6	8
	+		9	4
做加法		9	15	12
进位得结果	1	0	6	2

在高精度运算中，为了方便模拟右对齐、从右向左做加法，以及从右向左进位的过程。

- ① 将 2 个数逆序、逐位存入 2 个整数数组，这样就能实现左对齐；
- ② 然后从左向右做加法以及进位。

#### 加法的高精度模拟实现

	下标	0	1	2	3	
		8	6	9		数组a 数组b 数组c
	+	4	9	0		
做加法		12	15	9		
进位得结果		2	6	0	1	
逆序输出结果		1	0	2	6	

```

s1 = "968"
s1[0] -> a[s1.size() - 1]
s1[1] -> a[s1.size() - 2]
s1[i] -> a[s1.size() - i - 1]
#include <bits/stdc++.h>
using namespace std;

/*
    第一步：用 string 读入高精度整数
    第二步：将两个高精度整数逆序存入 ab 两个整数数组
    第三步：从左向右，逐位求和，结果存入 c 数组
            从左向右，逐位进位
    第四步：逆序输出结果
*/

string s1,s2;//高精度整数
int a[250],b[250],c[500];
int i,j,len;

int main(){
    cin>>s1>>s2;
    //第二步：将两个高精度整数逆序存入 ab 两个整数数组
    for(i = 0;i < s1.size();i++){
        a[s1.size() - i - 1] = s1[i] - '0';
    }

```

```

for(i = 0; i < s2.size(); i++){
    b[s2.size() - i - 1] = s2[i] - '0';
}

//第三步： 从左向右，逐位求和，结果存入 c 数组
//          从左向右，逐位进位
//加法的次数，取决于两个整数的较长的字符串
len = s1.size();
if(s2.size() > s1.size()){
    len = s2.size();
}

//逐位相加
for(i = 0; i < len; i++){
    c[i] = a[i] + b[i];
}

//逐位进位
for(i = 0; i < len; i++){
    if(c[i] >= 10){
        c[i + 1] = c[i + 1] + c[i] / 10;
        c[i] = c[i] % 10;
    }
}

//第四步： 逆序输出结果
//两个不超过 len 位的整数做加法，结果可能是 len+1 位
if(c[len] != 0){
    len++;
}

//逆序输出结果
for(i = len - 1; i >= 0; i--){
    cout<<c[i];
}
}

```

### 1269: 【基础】高精度减法

思路图解：

如果  $a > b$ ，结果为正，用  $a - b$  运算；如果  $a < b$ ，结果为负，交换  $a$ 、 $b$  的值，再用  $a - b$  运算。

这里假设  $a > b$ ，则思路如下：

减法的数学实现

		借位
8	16	
9	6	8
-	0	9
	4	
做减法	8	7
	4	

### 减法的高精度模拟实现

	下标	0	1	2	3	
		8	6	9		数组a
-		4	9	0		数组b
		4	7	8		数组c
做减法，不够借位						
逆序输出结果		8	7	4		

```

#include <bits/stdc++.h>
using namespace std;

```

```

/*
高精度减法：
第一步：判断正负，如果 s1 比 s2 对应的整数小，结果为负，交换 s1 s2
第二步：将两个字符串，逆序存入 2 个整数输出测试
第三步：从左至右，逐位相减，不够借位
第四步：从右向左，逆序输出
*/

string s1,s2;
int a[250],b[250],c[250];
int i,len,p;
char f = '+'; //表示结果的正负

int main(){
    cin>>s1>>s2;
    //长的一定大，一样长字典码大的一定大
    // "123" "3" "123" "125"
    if(s1.size() < s2.size() || (s1.size() == s2.size() && s1 < s2)){
        f = '-';
        swap(s1,s2); //直接交换两个变量的值
    }

    //cout<<f<<" "<<s1<<" "<<s2;

    //将 s1 和 s2 逆序存入整数数组
    for(i = 0; i < s1.size(); i++){
        //0 -> s1[s1.size()-1]
        //1 -> s1[s1.size()-2]
        a[i] = s1[s1.size() - i - 1] - '0';
    }

    for(i = 0; i < s2.size(); i++){
        b[i] = s2[s2.size() - i - 1] - '0';
    }

    //逐位相减
    len = s1.size();

    for(i = 0; i < len; i++){
        //如果不够减，向右借 1，当 10 用
        if(a[i] < b[i]){
            a[i + 1] = a[i + 1] - 1;
            a[i] = a[i] + 10;
        }

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

    //判断是否要输出负号
    if(f == '-') cout<<f;

    //从右向左逐位输出，从第一个遇到的非 0 元素开始输出
    for(i = len - 1; i >= 0; i--){
        if(c[i] != 0){
            p = i;
            break;
        }
    }

    //逆序从第一个非 0 元素 输出每一位
    for(i = p; i >= 0; i--){
        cout<<c[i];
    }
}

```

注意：swap(a,b)：交换变量 ab 的值！

1286: 【基础】高精度乘单精度

高精度\*单精度的数学实现

$$\begin{array}{r}
 \begin{array}{r}
 \text{用单精度逐位乘高精度} \\
 \text{逐位进位}
 \end{array}
 \begin{array}{r}
 1 \quad 2 \quad 5 \\
 * \\
 \hline
 25 \quad 50 \quad 125 \\
 3 \quad 1 \quad 2 \quad 5
 \end{array}
 \end{array}$$

高精度模拟实现						
	下标	0	1	2	3	
		5	2	1		数组a
	*	25				整数b
逐位乘		125	50	25		数组c
逐位进位		5	2	1	3	
逆序输出结果		3	1	2	5	

1287: 【基础】高精度乘

乘的数学实现

$$\begin{array}{r}
 \begin{array}{r}
 \text{逐位相乘（进位）} \\
 \text{错位相加+}
 \end{array}
 \begin{array}{r}
 1 \quad 2 \quad 5 \\
 * \\
 \hline
 2 \quad 5 \\
 6 \quad 2 \quad 5 \\
 2 \quad 5 \quad 0 \\
 3 \quad 1 \quad 2 \quad 5
 \end{array}
 \end{array}$$

高精度模拟实现

	下标	0	1	2	3	
		5	2	1		数组a
	*	5	2			数组b
逐位乘（进位）		5	2	6		数组c
逐位乘（进位）			0	5	2	数组c
错位相加+		5	2	1	3	
逆序输出结果		3	1	2	5	

```

#include <bits/stdc++.h>
using namespace std;

string s1,s2;
int a[250],b[250],c[500];
int i,j,p;

int main(){
    cin>>s1>>s2;
    //逆序将 s1 和 s2 存入 ab 数组
    for(i = 0;i < s1.size();i++){
        a[i] = s1[s1.size() - i - 1] - '0';
    }

    for(i = 0;i < s2.size();i++){

```

```

        b[i] = s2[s2.size() - i - 1] - '0';
    }

    //循环 a 数组的每一位，用 a[i]去乘以 b 数组的每一位 b[j]
    //结果错位加到 c 数组的 c[i+j]这一位上
    for(i = 0; i < s1.size(); i++){
        for(j = 0; j < s2.size(); j++){
            c[i+j] = c[i+j] + a[i] * b[j];
            //进位
            if(c[i+j] >= 10){
                c[i+j+1] = c[i+j+1] + c[i+j] / 10;
                c[i+j] = c[i+j] % 10;
            }
        }
    }

    //逆序输出，逆序从第一个非 0 元素位置开始输出
    for(i = s1.size() + s2.size() - 1; i >= 0; i--){
        if(c[i] != 0){
            p = i;
            break;
        }
    }

    for(i = p; i >= 0; i--){
        cout<<c[i];
    }
}

```

#### 1271: 【基础】高精度整数除法

求 1 除以 8 的计算结果，模拟过程如下。

$$\begin{array}{r}
 0.125 \\
 8 \overline{) 10} \\
 \underline{8} \phantom{00} \\
 20 \\
 \underline{16} \phantom{00} \\
 40 \\
 \underline{40} \phantom{00} \\
 0
 \end{array}$$

```

#include <bits/stdc++.h>
using namespace std;

int main(){
    int a,b,i,n,t;
    cin>>a>>b>>n;
    cout<<a / b<<".";
    t = a % b;
    for(i = 1; i <= n; i++){
        t = t * 10;
        cout<<t / b;
        t = t % b;
    }
}

```

### 1280: 【基础】求 2 的 n 次方

a 整数数组，存放 2 的 n 次方

8	2	1						
0	1	2	3	4	5	6	7	8

k=3

a[k]

思路:

准备一个整数数组 a，存放 2 的 n 次方，a 数组默认存储一个 1，代表 2 的 0 次方！

循环 n 次，每次循环都要将 a 数组的每一位\*2，并进位，然后判断 a 数组\*2 后是否多出一位，如果多出一位，a 数组位数计数器 k++。

逆序输出 a 数组的 k 个数。

```
#include <bits/stdc++.h>
using namespace std;
```

```
int a[100] = {1};
//k 代表 a 数组元素的个数，代表了高精度的整数的位数
int i,j,k = 1,n;
int main(){
    cin>>n;
    //循环 n 次，每次都将 a 数组 * 2
    for(i = 1;i <= n;i++){
        //将 a 数组的每一位都 * 2
        for(j = 0;j < k;j++){
            a[j] = a[j] * 2;
        }

        //逐位进位
        for(j = 0;j < k;j++){
            if(a[j] >= 10){
                a[j+1]=a[j+1]+a[j]/10;
                a[j]=a[j]%10;
            }
        }

        //判断 a 数组是否多出一位
        if(a[k] != 0){
            k++;
        }
    }

    //逆序输出 a 数组的 k 个数
    for(i = k - 1;i >= 0;i--){
        cout<<a[i];
    }
}
```

### 1281: 【基础】求 2+2\*2+2\*2\*2+...+2\*2\*2\*... \*2

a 整数数组，存放 2 的 n 次方

6	1							
0	1	2	3	4	5	6	7	8

r 整数数组，存放总和

0	3							
0	1	2	3	4	5	6	7	8

```
#include <bits/stdc++.h>
using namespace std;
```

```
int a[100] = {1};
```



```
int r[1000];
//k 代表 a 数组元素的个数，代表了 2 的 n 次方高精度的整数的位数
//k2 代表了高精度的总和的位数
int i,j,k = 1,n,k2 = 1,len;
int main() {
    cin>>n;
    //循环 n 次，每次都将 a 数组 * 2
    for(i = 1; i <= n; i++) {
        //将 a 数组的每一位都 * 2
        for(j = 0; j < k; j++) {
            a[j] = a[j] * 2;
        }

        //逐位进位
        for(j = 0; j < k; j++) {
            if(a[j] >= 10) {
                a[j+1]=a[j+1]+a[j]/10;
                a[j]=a[j]%10;
            }
        }

        //判断 a 数组是否多出一位
        if(a[k] != 0) {
            k++;
        }

        //求出了 2 的 i 次方，结果为 k 位
        //将 k 位的 2 的 i 次方，加到 k2 位的总和 r 上
        len = k;
        if(k2 > k) len = k2;
        for(j = 0; j < len; j++) {
            r[j] = r[j] + a[j];
            //进位
            if(r[j] >= 10) {
                r[j+1]=r[j+1]+r[j]/10;
                r[j]=r[j]%10;
            }
        }
        //判断 r 数组是否多了 1 位
        if(r[k2]!=0) {
            k2++;
        }
    }

    //输出 r 数组的结果
    for(i = k2 - 1; i >= 0; i--) {
        cout<<r[i];
    }
}
```

## 二、作业

- 1285: 【基础】计算 N 的阶乘
- 1296: 【基础】求  $1!+2!+3!+4!+\dots+n!$
- 1409: 【基础】棋盘里的麦子?
- 1369: 【提高】Pell 数列

附加题:



1532: 【提高】小 X 与神牛

1368: 【提高】蜜蜂路线

