



实验舱
青少年编程
走近科学 走进名校

实验舱蛟龙三班

Day11 高精度 乘除

zlj

2022.12

复习、高精度 加减法

$$\begin{array}{r} 5 3 3 \\ + 3 4 7 \\ \hline 8 7 10 \end{array}$$

$$\begin{array}{r} 5 3 3 \\ - 3 4 7 \\ \hline 2 - 1 - 6 \end{array}$$

一、高精度加法

思考：现有两个不超过100位的正整数，要求你将它们加起来的和输出。

【輸入】

39827384

3048503945903485

【输出】

3048503985730869

$$\begin{array}{rcccc} & 5 & 3 & 7 & \\ + & 5 & 4 & 3 & \\ \hline 1 & 0 & 8 & 0 & \end{array}$$

高精度加法

$$\begin{array}{r} 5 6 7 \\ + 6 2 8 9 \\ \hline 6 7 14 16 \end{array}$$

问题：

- 1、如何存数？
- 2、按位相加，结果如何存？
- 3、如何进位？
- 4、如何输出结果？

高精度加法

	5	6	7
+	6	2	8
9			

6	7	14	16
---	---	----	----

【算法分析】

1、输入并转换为int 数组

2、 `int a[500], b[500], c[500]`

3、按位相加：`c[i]=a[i]+b[i]`

4、如何进位？

`c[i+1]=c[i]/10;`

`c[i]=c[i]%10;`

5、按位输出结果。

```
5 string s1,s2;cin>>s1>>s2;
6 int len1=s1.size();int len2=s2.size();
7 for(int i=0; i<len1; i++) { //将字符转数字并反转
8     a[i]=s1[len1-i-1]-'0';
9 }
10 for(int i=0; i<len2; i++) {
11     b[i]=s2[len2-i-1]-'0';
12 }
13 int len=max(len1,len2);
14 for(int i=0; i<len; i++) {
15     ans[i]=a[i]+b[i]; //按位相加
16 }
```

```
17 for(int i=0; i<len; i++) {
18     ans[i+1]+=ans[i]/10; //处理进位
19     ans[i]%=10;
20 }
21 while(ans[len]){ //最后一位进位处理
22     ans[len+1]+=ans[len]/10;
23     ans[len]%=10;
24     len++;
25 }
26 for(int i=len-1; i>=0; i--){ //倒序输出
27     cout<<ans[i];
28 }
```

二、高精度减法

现有两个10000位的正整数a和b，要求你将它们做减法后输出。

【输入】

3048503945903485

39827384

【输出】

3048503906076101

$$\begin{array}{r} \\ 5 3 3 \\ - 3 4 7 \\ \hline 1 1 \\ 1 8 6 \end{array}$$

高精度减法

$$\begin{array}{r} 5 3 3 \\ - 3 4 7 \\ \hline 2 - 1 - 4 \end{array}$$

 1 1

【算法分析】

- 1、字符串转int 数组a, b, c
- 2、按位相减: $c[i] = a[i] - b[i]$
- 3、如何借位:
 如果 $c[i] < 0$:
 $c[i] = c[i] + 10$;
 $a[i+1] = a[i+1] - 1$;
- 4、按位输出
 考虑前置 0 如: 002;
- 5、考虑结果负数情况

字符串模拟减法

```
4 bool cmp(string x,string y) { // 比较大小规则
5     if(x.size()!=y.size())return x.size()<y.size();
6     return x<y;
7 }
8 int main() {
9     string s1,s2;cin>>s1>>s2;
10    int pd=0;
11    if(cmp(s1,s2)) {
12        swap(s1,s2);
13        pd=1; // 标记负数
14    }
15    int len1=s1.size();int len2=s2.size();
16    for(int i=0; i<len1; i++) {
17        a[i]=s1[len1-i-1]-'0';
18    }
19    for(int i=0; i<len2; i++) {
20        b[i]=s2[len2-i-1]-'0';
21    }
```

```
22  for(int i=0; i<len1; i++) {  
23      a[i]-=b[i]; //按位相减  
24  }  
25  for(int i=0; i<len1; i++) {  
26      if(a[i]<0) { //处理借位  
27          a[i+1]-=1;  
28          a[i]+=10;  
29      }  
30  }  
31  while(len1>1 && a[len1-1]==0) {  
32      len1--; //去前缀0  
33  }  
34  if(pd==1) {  
35      cout<<'-';  
36  } //倒序输出  
37  for(int i=len1-1; i>=0; i--) {  
38      cout<<a[i];
```

《很大的斐波那契数》

```
5      cin>>n;
6      int a=b=1,c; //小范围的求法
7      for(int i=3;i<=n;i++){
8          c=a+b;
9          a=b;
10         b=c;
11     }
12     cout<<c<<endl;
13     return 0;
```

```

3  int a[50005]= {1},b[50005]= {1},c[50005];
4  int main() {
5      int n;
6      cin>>n;
7      int len=1;
8      for(int i=3; i<=n; i++) {
9          int jw = 0;    //将a,b换成2个数组滚动
10         for(int j=0; j<len; j++) {
11             c[j]=a[j]+b[j]+jw;
12             jw=c[j]/10;
13             c[j]%=10;
14         }
15         if(jw>0) { //进位处理
16             c[len]=jw;
17             len++;
18         }
19         for(int j=0; j<len; j++) {
20             a[j]=b[j]; //交换a,b两数组
21             b[j]=c[j];
22         }
23
24         for(int i=len-1; i>=0; i--) {
25             cout<<c[i];
26         }

```

```

4 string bigadd(string sx,string sy) {
5     int len1=sx.size();int len2=sy.size();
6     memset(a,0,sizeof(a)); memset(b,0,sizeof(b));
7     for(int i=0; i<len1; i++) {
8         a[i]=sx[len1-i-1]-'0';
9     }
10    for(int i=0; i<len2; i++) {
11        b[i]=sy[len2-i-1]-'0';
12        a[i]+=b[i];
13    }
14    int len=max(len1,len2);
15    string ans="";
16    for(int i=0; i<len; i++) {
17        a[i+1]+=a[i]/10;
18        a[i]%=10;
19        ans=char(a[i]+'0')+ans;
20    }
21    if(a[len]>0){//最多一个进位
22        ans=char(a[len]+'0')+ans;
23    }
24    return ans;//返回一个大整数串

```

```

26 string s[50005];
27 int main() {
28     ios::sync_with_stdio(false);
29     int n;cin>>n;
30     s[1]="1";s[2]="1";
31     for(int i=3;i<=n;i++){
32         s[i]=bigadd(s[i-1],s[i-2]);
33     }
34     cout<<s[n];
35     return 0;

```

一、高精度 乘 单精：

$$\begin{array}{r} 523 \\ * 11 \\ \hline \end{array}$$

$$\begin{array}{r} 5523 \end{array}$$

一、高精度 乘 单精：

$$\begin{array}{r} 3 2 5 \\ * 11 \\ \hline 33 22 55 \\ 3 5 7 5 \end{array}$$

算法步骤：

- 1、大数用字符串读入并倒置
- 2、将字符转成数字
- 3、按位分别乘单精数 y
- 4、处理进位
- 5、返回大数并输出。

例：计算x的N次方（单精度）

$2^{100} =$
1267650600228229401496703205376

任意给定一个正整数N ($N \leq 100$)，计算2的n次方的值。

输入一个正整数N和X。

输出X的N次方的值。

样例输入

2 10

样例输出

1024

例： 计算x的N次方（单精度）

```
4 string bigs(string s,int x){
5     memset(a,0,sizeof(a));
6     int len=s.size();
7     for(int i=0;i<len;i++){
8         a[i]=s[len-i-1]-'0';
9         a[i]=a[i]*x;
10    }
11    //进位
12    for(int i=0;i<len;i++){
13        a[i+1]+=a[i]/10;
14        a[i]%=10;
15    }
16    while(a[len]){
17        a[len+1]+=a[len]/10;
18        a[len]%=10;
19        len++;
20    }
21    string ans=""; //倒序合成字符串
22    for(int i=len-1;i>=0;i--){
23        ans+=char(a[i]+'0');
24    }
25    return ans;
```

```
27 int main() {
28     int x,n;cin>>x>>n;
29     string ans="1"; //初始化串
30     for(int i=1;i<=n;i++){
31         ans=bigs(ans,x); //连续高精乘单精
32     }
33     for(auto i:ans){ //输出
34         cout<<i;
35     }
```

高精度乘法（双精度）

现有两个10000位的正整数a
和b，求出a*b的值并输出。

【输入】

3048503945903485

39827384

【输出】

121413937279013324033240

$$\begin{array}{r} 523 \\ * 47 \\ \hline 3661 \\ 2092 \\ \hline 24581 \end{array}$$

		5	2	3
	*		4	7
<hr/>				
		35	14	21
	20	8	12	
<hr/>				
	20	43	26	21
	²	⁴	²	²
2	4	5	8	1

【算法分析】

数组a和数组b分别置放被乘数和乘数，然后按位相乘及相加。

1. 二重循环，计算 $a[i]*b[j]$

2. 移位相加

$$c[i+j]=c[i+j]+a[i]*b[j]$$

3. 用一个循环专门处理进位。

$$c[i+1]=c[i+1]+c[i]/10;$$

$$c[i]=c[i]\%10;$$

		5	2	3
	*		4	7
		35	14	21
	20	8	12	
	20	43	26	21
2	2	4	2	2
		4	5	8
				1

高精度乘法：代码实现

```
15      td(s2,b); //将s1倒序存入 a 数组
16      for(int i=0;i<len1;i++){
17          for(int j=0;j<len2;j++){
18              ans[i+j]+=a[i]*b[j]; //错位相乘
19          }
20      }
21      int len=len1+len2; // 预估长度
22      for(int i=0;i<len;i++){//进位
23          ans[i+1]+=ans[i]/10;
24          ans[i]%=10;
25      }
26      while(ans[len]){
27          ans[len+1]+=ans[len]/10;
28          ans[len]%=10;
29          len++;
30      }
31      while(len>1&&ans[len-1]==0){//处理前置 0
32          len--;
33      }
```

例： 小Z的乘法：

小Z学会了乘法，但是计算机的世界里，如果数据够大，乘法将困难重重，好在学了高精度，请你帮小Z用程序解决三个数相乘的难题。

输入格式：

三行，输入三个整数 a, b, c ($0 \leq a, b, c \leq 10^{1000}$)。

输出格式：

一行，三个数的乘积。

输入样例：

1
2
3

输出样例：

6

三、高精度 除 单精：

$$\begin{array}{r}
 19 \overline{) \begin{array}{r} 1067 \\ 2019 \\ 1171 \\ 1144 \end{array}} \\
 \underline{2019} \\
 1171 \\
 \underline{1144} \\
 27
 \end{array}$$

		0	1	0	6
19)	2	0	1	7
		1	9		
			1	1	7
			1	1	4
					3

高精度除法（单精度）

【算法分析】

数组a置放被除数，然后采用竖式除法的方法。

```
yu=0;
for(int i=0;i<len;i++)
{
    c[i]=(yu*10+a[i])/b;
    yu=(yu*10+a[i])%b;
}
```

程序结果

c[]={0, 1, 0, 6}

yu=3

	0	1	0	6
19	2	0	1	7
	1	9		
		1	1	7
		1	1	4
				3

例4：高精度除法（单精度）

现有两个正整数a和b，a的位数小于10000，b的值小于100000，计算 a/b ，将结果输出，保留小数点20位。

输入

40933903092948

93872

输出

436060839.15276556735800802667

如何保留10位小数？

```
6 |  
7 |  
8 |  
9 |  
10 |  
11 |  
12 |  
13 |  
    cout<<a/b<<'.';  
    int yu=a%b;  
    // 保留小数位数  
    for(int i=0;i<n;i++){  
        cout<<(yu*10)/b;  
        yu=(yu*10)%b;  
    }  
    return 0;
```

练习： 请将高精除单精改写成 函数的形式

例5:大整数的因子

【描述】

已知正整数 k 满足 $2 \leq k \leq 9$ ，现给出长度最大为30位的十进制非负整数 c ，求所有能整除 c 的 k 。

【输入】

一个非负整数 c ， c 的位数 ≤ 30 。

【输出】

若存在满足 $c \% k == 0$ 的 k ，从小到大输出所有这样的 k ，相邻两个数之间用单个空格隔开；若没有这样的 k ，则输出“none”。

【样例输入】

30

样例输出

2 3 5 6

小结：

1、高精度乘法

2、高精度除单精